

Fig. 6. The successive erosion of the image in the top left by a diamond shaped structuring element. Pixels which are white in the i th erosion are white pixels in the input image which have a 4-distance of greater than i pixels to the black background.

formulation employing the first operator is equal to that formulation employing operator on the negated variables. An example is DeMorgan's law, illustrating the duality of union and intersection,

$$(A \cup B)^c = A^c \cap B^c.$$

Here the negation of a set A is its complement,

$$A^c = \{x \in E^n \mid x \notin A\}.$$

In morphology, negation of a set is considered in a geometrical sense: that of reversing the orientation of the set with respect to its coordinate axes. Such reversing is called reflection.

Definition 24: Let $B \subseteq E^n$. The reflection of B is denoted by \tilde{B} and is defined by

$$\tilde{B} = \{x \mid \text{for some } b \in B, x = -b\}.$$

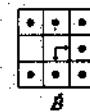
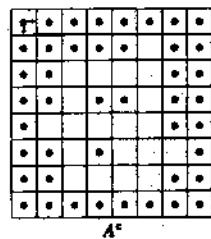
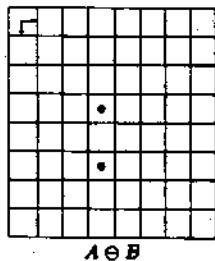
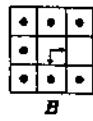
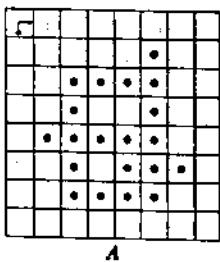
The reflection occurs about the origin. Matheron [18] refers to \tilde{B} as "the symmetrical set of B with respect to the origin." Serra [31] refers to \tilde{B} as " B transpose."

As given in Theorem 25, the duality of dilation and erosion employs both logical and geometric negation because of the different roles of the image and the structuring element in an expression employing these morphological operators.

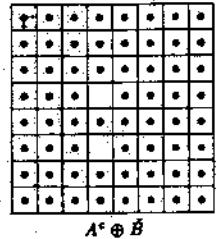
Erosion Dilation Duality Theorem 25: $(A \ominus B)^c = A^c \oplus \tilde{B}$.

Proof: $x \in (A \ominus B)^c$ if and only if $x \notin A \ominus B$. $x \notin A \ominus B$ if and only if there exists $b \in B$ such that $x + b \notin A$. There exists $b \in B$ such that $x + b \in A^c$ if and only if there exists $b \in B$ such that $x \in (A^c)_{-b}$. There exists $b \in B$ such that $x \in (A^c)_{-b}$ if and only if $x \in \bigcup_{b \in B} (A^c)_{-b}$. Now, $x \in \bigcup_{b \in B} (A^c)_{-b}$ if and only if $x \in \bigcup_{b \in B} (A^c)_b$; and $x \in \bigcup_{b \in B} (A^c)_b$ if and only if $x \in A^c \oplus \tilde{B}$.

Example: This illustrates an instance of the relationship $(A \ominus B)^c = A^c \oplus \tilde{B}$.



A^c



$A^c \oplus B$

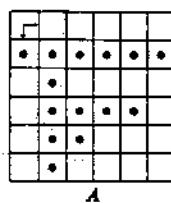
The difference between the dilation and erosion transformations is illuminated in the algebraic properties of the erosion as contrasted with the dilation. First, the erosion of the intersection of two images is equal to the intersection of their erosions. This contrasts with Proposition 14 where the relationship is one of containment.

Proposition 26:

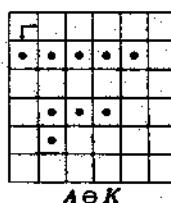
$$(A \cap B) \ominus K = (A \ominus K) \cap (B \ominus K).$$

Proof: $x \in (A \cap B) \ominus K$ if and only if for every $k \in K$, $x + k \in A \cap B$. $x + k \in A \cap B$ if and only if $x + k \in A$ and $x + k \in B$. $x + k \in A$ for every $k \in K$ if and only if $x \in A \ominus K$. $x + k \in A$ for every $k \in K$ if and only if $x \in B \ominus K$. $x + k \in A$ for every $k \in K$ and $x + k \in B$ for every $k \in K$ if and only if $x \in (A \ominus K) \cap (B \ominus K)$.

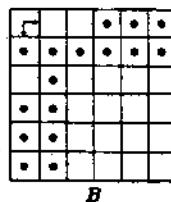
Example: This illustrates an instance of the relationship $(A \cap B) \ominus K = (A \ominus K) \cap (B \ominus K)$.



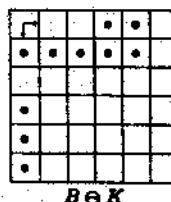
A



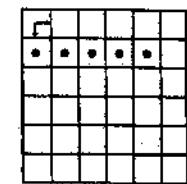
$A \ominus K$



B



$B \ominus K$



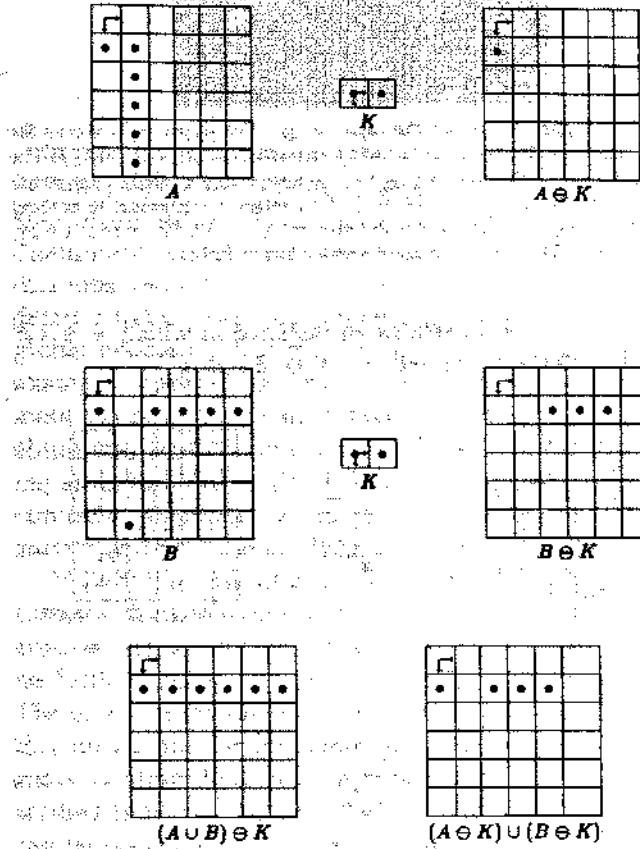
$(A \cap B) \ominus K = (A \ominus K) \cap (B \ominus K)$

On the other hand, whereas the dilation of the unions of two images is equal to the union of their dilations (Proposition 15), for the erosion transformation the relationship is one of containment.

Proposition 27: $(A \cup B) \ominus K \supseteq (A \ominus K) \cup (B \ominus K)$.

Proof: Let $x \in (A \ominus K) \cup (B \ominus K)$. Then $x \in A \ominus K$ or $x \in B \ominus K$. If $x \in A \ominus K$ then since $A \cup B \supseteq A$, $x \in (A \cup B) \ominus K$. If $x \in B \ominus K$ then since $A \cup B \supseteq B$, $x \in (A \cup B) \ominus K$.

Example: This illustrates an instance in which $(A \cup B) \ominus K$ strictly contains $(A \ominus K) \cup (B \ominus K)$.

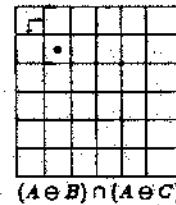
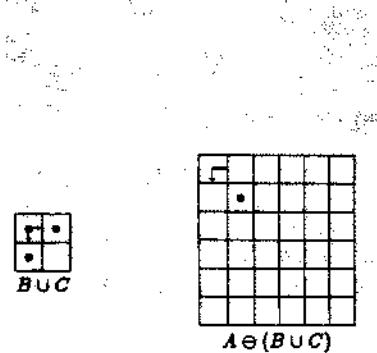
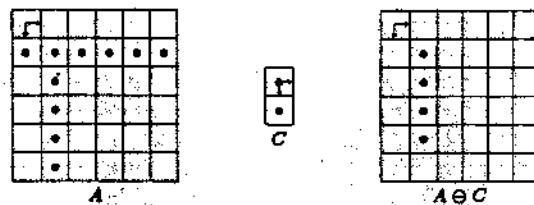
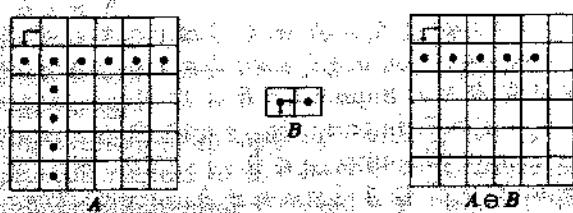


But erosion is not commutative, $A \ominus B \neq B \ominus A$. Hence the behavior of $A \ominus (B \cup C)$ indicated in the equality of Proposition 28 is different than the behavior of $(A \cup B) \ominus C$ as indicated in Proposition 27.

Proposition 28: $A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C)$

Proof: $x \in A \ominus (B \cup C)$ if and only if $x + y \in A$ for every $y \in B \cup C$. $x + y \in A$ for every $y \in B \cup C$ if and only if $x + y \in A$ for every $y \in B$ and $x + y \in A$ for every $y \in C$. $x + y \in A$ for every $y \in B$ if and only if $x \in A \ominus B$. $x + y \in A$ for every $y \in C$ if and only if $x \in A \ominus C$. $x + y \in A$ for every $y \in B$ and $x + y \in A$ for every $y \in C$ if and only if $x \in (A \ominus B) \cap (A \ominus C)$.

Example: This illustrates an instance of the relationship $A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C)$.



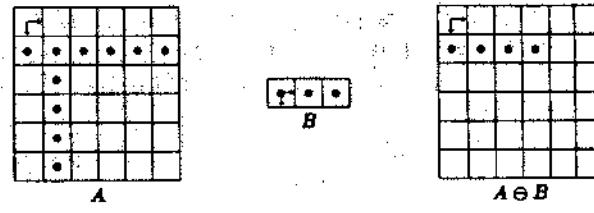
The practical utility of Proposition 28 is that it indicates how to compute erosions with structuring elements which can only be decomposed as the union of individual structuring elements.

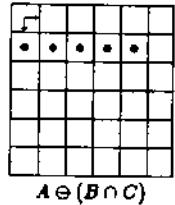
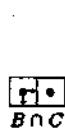
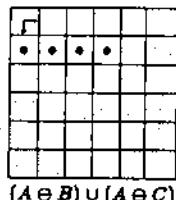
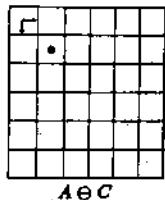
Although structuring elements can be decomposed through union into simpler structuring elements to simplify the erosion transformation, structuring elements cannot be decomposed through intersection and maintain an equality. Rather, the intersection decomposition leads to a containment relationship.

Proposition 29: $A \ominus (B \cap C) \supseteq (A \ominus B) \cup (A \ominus C)$.

Proof: Let $x \in (A \ominus B) \cup (A \ominus C)$. Then $x \in A \ominus B$ or $x \in A \ominus C$. If $x \in A \ominus B$, then $x + b \in A$ for every $b \in B$. If $x \in A \ominus C$, then $x + b \in A$ for every $b \in C$. Hence, $x + b \in A$ for every $b \in B \cap C$. Now by definition of erosion, $x \in A \ominus (B \cap C)$.

Example: This illustrates an instance in which $A \ominus (B \cap C)$ strictly contains $(A \ominus B) \cup (A \ominus C)$.





Finally, with respect to structuring element decomposition, a chain rule for erosion holds when the structuring element is decomposable through dilation,

$$A \ominus (B \oplus C) = (A \ominus B) \oplus C.$$

This relation is as important as the chain rule relation for dilation because it permits a large erosion to be computed by two successive smaller erosions.

Proposition 30: $(A \ominus B) \oplus C = A \ominus (B \oplus C)$.

Proof: Let $x \in (A \ominus B) \oplus C$. Then for every $c \in C$, $x + c \in A \ominus B$. But $x + c \in A \ominus B$ implies $x + c + b \in A$ for every $b \in B$. But $x + b + c \in A$ for every $b \in B$ and $c \in C$ implies $x + d \in A$ for every $d \in B \oplus C$.

Let $x \in A \ominus (B \oplus C)$. Then $x + d \in A$ for every $d \in B \oplus C$. Hence $x + b + c \in A$ for every $b \in B$ and $c \in C$. Now $(x + c) + b \in A$ for every $b \in B$ implies $x + c \in A \ominus B$. But $x + c \in A \ominus B$ for every $c \in C$ implies $x \in (A \ominus B) \oplus C$.

Corollary 31 extends this result to structuring elements decomposed as the dilation of K structuring elements.

Corollary 31: $A \ominus (B_1 \oplus \dots \oplus B_K) = (\dots (A \ominus B_1) \oplus \dots \oplus B_K)$.

It is immediately apparent from the corollary that because dilation is commutative, the order in which successive erosions are applied is immaterial.

Fig. 7 illustrates the utility of the chain rule for erosions.

Reversing the position of dilation and erosion in Proposition 30 does not lead to an equality as in Proposition 30 but leads to a containment relation as given in Proposition 32. In some sense this indicates that when performing erosion and dilation, performing erosion first is more severe than performing dilation first.

Proposition 32: $A \oplus (B \ominus C) \subseteq (A \oplus B) \ominus C$.

Proof: Let $x \in A \oplus (B \ominus C)$. Then for some $a \in A$ and $B \ominus C$, $x = a + y$. But $y \in B \ominus C$ implies $y + c \in B$ for every $c \in C$. Now $y + c \in B$ and $a \in A$ implies $y + c + a \in A \oplus B$. Finally $y + c + a \in A \oplus B$ for every $c \in C$ implies $x = y + a \in (A \oplus B) \ominus C$.

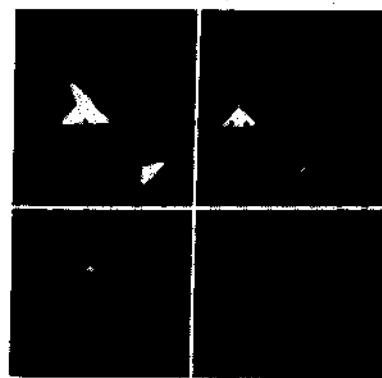
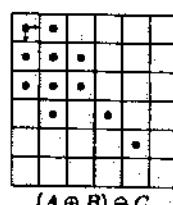
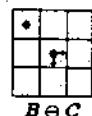
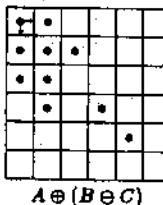
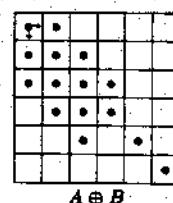
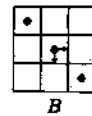
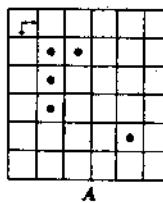


Fig. 7: The upper left shows the input image. The upper right shows the input image eroded by the structuring element $\{(0,0), (0, -14)\}$. The lower left shows the eroded image of the upper right eroded by the structuring element $\{(0,0), (-14,0)\}$. This result is equivalent to eroding the input image by the structuring element $\{(0,0), (0, -14), (-14,0), (-14, -14)\}$ which is shown in the lower right.

Example: This illustrates an instance in which $A \oplus (B \ominus C)$ is strictly contained in $(A \oplus B) \ominus C$.



Although dilation and erosion are dual, this does not imply that we can freely perform cancellation on morphological equalities. For example, if $A = B \ominus C$, then dilating both sides of the expression by C results in $A \oplus C = B \ominus C \oplus C \neq B$. However, a containment relationship is maintained, as indicated in Proposition 33.

Proposition 33: $A \subseteq B \ominus C$ if and only if $B \supseteq A \oplus C$.

Proof: Suppose $A \subseteq B \ominus C$. Let $x \in A \oplus C$. Then there exists $a \in A$ and $c \in C$ such that $x = a + c$. But $a \in A$ and $A \subseteq B \ominus C$ implies $a \in B \ominus C$. Hence, for every $c' \in C$, $a + c' \in B$. In particular, $c \in C$. Thus $a + c \in B$. But $x = a + c$. Therefore, $x \in B$.

Suppose $A \oplus C \subseteq B$. Let $x \in A$. Let $c \in C$. Then $x +$

$c \in A \oplus C$. But $A \oplus C \subseteq B$ so that $x + c \subseteq B$. Finally $x + c \subseteq B$ for any $c \in C$ implies $x \in B \ominus C$.

The containment is maintained for chained erosions, as follows.

Corollary 34:

$$A \subseteq (\cdots (B \ominus C_1) \ominus \cdots) \ominus C_N \text{ if and only if } (\cdots (A \oplus C_1) \oplus \cdots) \oplus C_N \subseteq B.$$

III. OPENINGS AND CLOSINGS

In practice, dilations and erosions are usually employed in pairs, either dilation of an image followed by the erosion of the dilated result, or image erosion followed by dilation. In either case, the result of iteratively applied dilations and erosions is an elimination of specific image detail smaller than the structuring element without the global geometric distortion of unsuppressed features. For example, opening an image with a disk structuring element smooths the contour, breaks narrow isthmuses, and eliminates small islands and sharp peaks or capes. Closing an image with a disk structuring element smooths the contours, fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps on the contours.

Of particular significance is the fact that image transformations employing iteratively applied dilations and erosions are idempotent, that is, their reapplication effects no further changes to the previously transformed result. The practical importance of idempotent transformations is that they comprise complete and closed stages of image analysis algorithms because shapes can be naturally described in terms of under what structuring elements they can be opened or can be closed and yet remain the same. Their functionality corresponds closely to the specification of a signal by its bandwidth. Morphologically filtering an image by an opening or closing operation corresponds to the ideal nonrealizable bandpass filters of conventional linear filtering. Once an image is ideal bandpassed filtered, further ideal bandpass filtering does not alter the result.

This property motivates the importance for having definitions of opening and closing, concepts first studied by Matheron [17], [18] who was interested in axiomatizing the concept of size. Both Matheron's [18] definitions and Serra's [31] definitions for opening and closing are identical to the ones given here, but their formulas appear different because they use the symbol \ominus to mean Minkowski subtraction rather than erosion.

Definition 35: The opening of image B by structuring element K is denoted by $B \circ K$ and is defined as $B \circ K = (B \ominus K) \oplus K$.

Definition 36: The closing of image B by structuring element K is denoted by $B \bullet K$ and is defined by $B \bullet K = (B \oplus K) \ominus K$.

If B is unchanged by opening it with K , we say that B is open with respect to K , while if B is unchanged by closing it with K , then B is closed with respect to K .

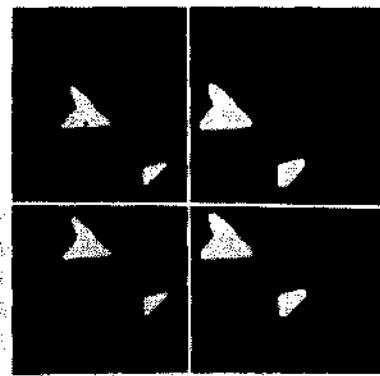


Fig. 8. The upper left shows the input image. In the upper right, the input image is dilated by a structuring element consisting of a 5×5 square. In the lower left, the dilated image is eroded by a 5×5 square structuring element. It is the closing of the input image. In the lower right, the closed image is dilated by a 5×5 square structuring element. It is the same as the initially dilated image shown in the upper right.

We approach the issue of idempotency of opening and closing by first discussing a class of sets which are unaltered by erosion followed by dilation with a given structuring element K . This class consists of all sets which can be expressed as some set dilated by K .

Proposition 37: $A \oplus K = (A \oplus K) \circ K = (A \bullet K) \oplus K$.

Proof: Let $G = A \oplus K$, $H = G \ominus K$, and $I = H \oplus K$. Now, by Proposition 33, $G = A \oplus K$ implies $A \subseteq G \ominus K = H$; $H = G \ominus K$ implies $G \supseteq H \oplus K = I$. But $A \subseteq H$ implies $A \oplus K \subseteq H \oplus K$. Since $G = A \oplus K$ and $I = H \oplus K$, $G \subseteq I$. Finally, $G \supseteq I$ and $G \subseteq I$ imply $G = I$. Hence,

$$\begin{aligned} A \oplus K &= H \oplus K = (G \ominus K) \oplus K \\ &= ((A \oplus K) \ominus K) \oplus K = (A \bullet K) \oplus K. \end{aligned}$$

Proposition 37 is illustrated in Fig. 8. The idempotency of closing follows immediately as given by Theorem 38.

Theorem 38: $(A \bullet K) \bullet K = A \bullet K$.

Proof:

$$\begin{aligned} A \oplus K &= (A \bullet K) \oplus K \\ (A \oplus K) \ominus K &= ((A \bullet K) \oplus K) \ominus K \\ A \bullet K &= ((A \bullet K) \bullet K). \end{aligned}$$

Similarly, images eroded by K are unaltered by further dilation and erosion by K .

Proposition 39: $A \ominus K = (A \circ K) \ominus K = (A \bullet K) \bullet K$.

Proof: Let $G = A \ominus K$, $H = G \oplus K$, and $I = H \ominus K$. Now, $G = A \ominus K$ implies $A \supseteq G \oplus K = H$; $H = G \oplus K$ implies $G \subseteq H \ominus K = I$. But $A \supseteq H$ implies $A \ominus K \supseteq H \ominus K$ so that $G \supseteq I$. Finally, $G \subseteq I$ and $G \supseteq I$ imply $G = I = H \ominus K = (G \oplus K) \ominus K$. Since $G = A \ominus K$, $A \ominus K = ((A \ominus K) \oplus K) \ominus K = (A \circ K) \ominus K$. The idempotency of opening follows immediately as given by Theorem 40.

Theorem 40: $A \circ K = (A \circ K) \circ K$.

Proof:

$$A \ominus K = (A \circ K) \oplus K$$

$$(A \ominus K) \oplus K = ((A \circ K) \oplus K) \oplus K$$

$$A \circ K = (A \ominus K) \circ K.$$

As with chained dilations and erosions, we can extend these results for chained openings and closings.

Openings and closings have other properties. For example, it follows immediately from the increasing property of dilation (Proposition 12) and the increasing property of erosion (Proposition 22) that both opening and closing are increasing.

It follows immediately from the translation invariance of dilation (Proposition 6) and the translation invariance of erosion (Proposition 21) that both opening and closing are translation invariant. Unlike dilation and erosion, opening and closing are invariant to translations of the structuring elements. That is, $A \circ (B)_x = A \circ B$ and $A \bullet (B)_x = A \bullet B$. This also follows directly from Propositions 6 and 21. As stated in Proposition 41, the opening transformation is antiextensive, i.e., the opening of A by structuring element B is necessarily contained in A , regardless of whether or not the origin belongs to B .

Antiextensivity of Opening Proposition 41: $A \circ B \subseteq A$.

Proof: Let $x \in A \circ B$. Then $x \in (A \ominus B) \oplus B$. Hence there exist $u \in A \ominus B$ and $v \in B$ such that $x = u + v$. Now $u \in A \ominus B$ implies $u + b \in A$ for every $b \in B$. In particular, $v \in B$. Thus $u + v \in A$. But $x = u + v$ so that $x \in A$.

Example: Illustrates how opening can produce a result which is strictly contained in the original.

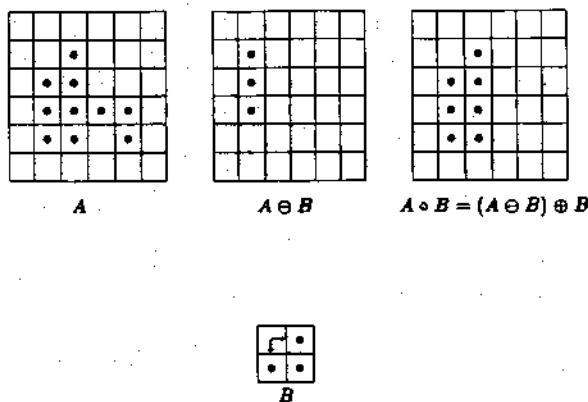


Fig. 9 illustrates an opening by a structuring element which does not include its origin.

The closing transformation is extensive, i.e., the closing of A by structuring element B contains A regardless of whether or not B contains its origin.

Extensivity of Closing Proposition 42: $A \subseteq A \bullet B$.

Proof: Let $a \in A$. Let $b \in B$. Then $a + b \in A \oplus B$. But $a + b \in A \bullet B$ for every $b \in B$ implies $a \in (A \oplus B) \ominus B$.

Example: This illustrates an instance of the relationship $A \subseteq A \bullet B$.

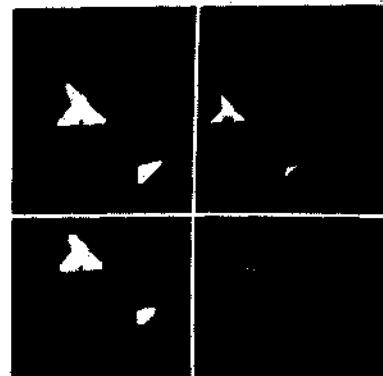


Fig. 9. The upper left shows the input image. In the upper right, the input image is eroded by the box boundary structuring element shown in the lower right. Notice that because the box is big enough to surround the hole and still be inside the blob with the hole, the eroded image has one white point whose position is in the hole. The lower left shows the image of the upper right dilated by the box boundary structuring element. This is the opening of the input image by the box boundary structuring element.

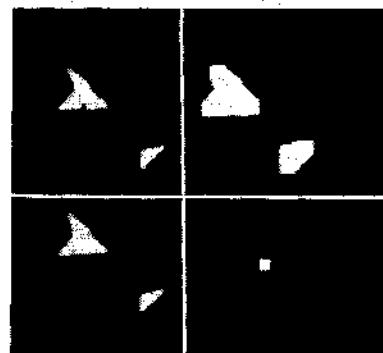


Fig. 10. The upper left shows the input image. In the upper right, the input image is dilated by a structuring element consisting of an 11×11 square shown in lower right. In the lower left, the dilated image is eroded by a 11×11 square structuring element. This results in the closing of the input image.

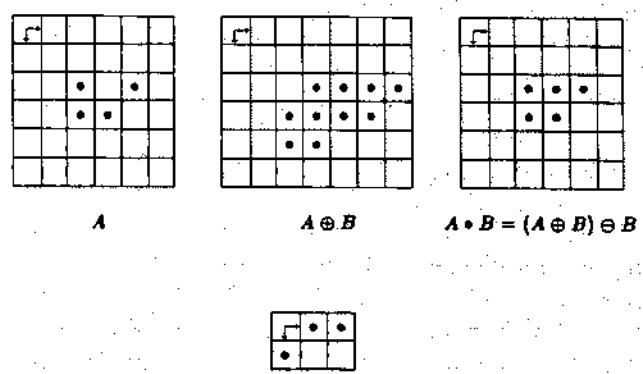


Fig. 10 illustrates a closing by a square structuring element.

Openings and closings, like erosion and dilations, are dual transformations. The complement of the closing of A by B is the opening of A^c by \bar{B} . This is illustrated in Fig. 11.

Duality of Opening and Closing Theorem 43: $(A \bullet B)^c = A^c \circ \bar{B}$.

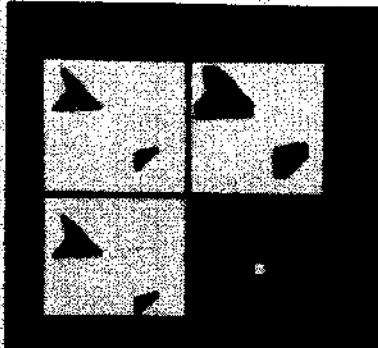


Fig. 11. The upper left shows the input image which is the complement of the input image of Fig. 10. In the upper right, the input image is eroded by a structuring consisting of an 11×11 square (shown in lower right). In the lower left, the eroded image is dilated by the 11×11 square structuring element. This results in the opening of the input image. Comparing Fig. 11 to Fig. 10, the opening of the complement is the complement of the closing for the symmetric square structuring element.

Proof:

$$\begin{aligned} (A \bullet B)^c &= [(A \oplus B) \ominus B]^c \\ &= (A \oplus B)^c \ominus B \\ &= (A^c \ominus B) \oplus B \\ &= A^c \circ B. \end{aligned}$$

Proposition 44 gives a geometric characterization to the opening operation. The opening of A by B is the union of all translations of B that are contained in A .

Proposition 44:

$$\begin{aligned} A \circ B &= \{x \in A \mid \text{for some } y, x \in B_y \subseteq A\} \\ &= \bigcup_{\{y \mid B_y \subseteq A\}} B_y. \end{aligned}$$

Proof: Suppose $x \in A \circ B$, then $x \in (A \ominus B) \oplus B$. Hence there exists a $y \in A \ominus B$ and $b \in B$ such that $x = y + b$. Since $y \in A \ominus B$, $y + b \in A$ with $b \in B$ implies $x \in B_y$.

Suppose $x \in A$ and for some $y, x \in B_y \subseteq A$, then for every $z \in B$, $z + y \in A$ and there must exist some $b \in B$ such that $x = b + y$. But $y + z \in A$ for every $z \in B$ implies by definition of erosion that $y \in A \ominus B$. And $x = b + y$ implies by definition of dilation that $x \in (A \ominus B) \oplus B = A \circ B$.

By the duality of opening and closing, it is immediate that the closing of A by B is the complement of the union of all translations of B that are contained in A^c . That is, $A \bullet B = (A^c \circ B) = [\bigcup_{\{y \mid B_y \subseteq A^c\}} B_y]^c$. Proposition 45 gives another geometric characterization to the closing operation.

Proposition 45:

$$\begin{aligned} A \bullet B &= \{x \in E^N \mid x \in B_y \text{ implies } B_y \cap A \neq \emptyset\} \\ &= \bigcap_{\{(y \mid B_y \cap A \neq \emptyset)\}} B_y. \end{aligned}$$

Proof: By Theorem 43, $A \bullet B = (A^c \circ B)^c$.

By Proposition 44, $A^c \circ B = \{x \in E^N \mid \text{for some } y, x \in B_y \subseteq A^c\}$. Hence,

$$\begin{aligned} A \bullet B &= (A^c \circ B)^c = \{x \in E^N \mid \text{for some } y, x \in B_y \text{ and } B_y \subseteq A^c\} \\ &= \{x \in E^N \mid \text{for some } y, x \in B_y \text{ and } B_y \cap A = \emptyset\} \\ &= \{x \in E^N \mid x \in B_y \text{ implies } B_y \cap A \neq \emptyset\}. \end{aligned}$$

Propositions 44 and 45 immediately imply that $A \circ B_x = A \circ B_y$ and $A \bullet B_x = A \bullet B_y$ for any x and y in E^N . Hence the origin of the structuring element makes no difference in the results of an opening or closing.

Also,

$$\left[\bigcup_{\{y \mid B_y \subseteq A^c\}} B_y \right]^c = \left[\bigcup_{\{y \mid B_y \cap A = \emptyset\}} B_y \right].$$

By DeMorgan's Law,

$$\left[\bigcup_{\{y \mid B_y \cap A^c \neq \emptyset\}} B_y \right]^c = \bigcap_{\{y \mid B_y \cap A^c \neq \emptyset\}} B_y^c.$$

IV. GRAY SCALE MORPHOLOGY

The binary morphological operations of dilation, erosion, opening, and closing are all naturally extended to gray scale imagery by the use of a min or max operation. Nakagawa and Rosenfeld [22] first discussed the use of neighborhood min and max operators. The general extensions, due to Sternberg [33], [35], keep all the relationships discussed in Sections II and III. Peleg and Rosenfeld [24] use gray scale morphology to generalize the medial axis transform to gray scale imaging. Peleg, Naor, Hartley, and Avnir [23] use gray scale morphology to measure changes in texture properties as a function of resolution. Werman and Peleg [38] use gray scale morphology for texture feature extraction. Favre, Muggli, Stucki, and Bonderet [5] use gray scale morphology for the detection of platelet thrombosis detection in cross sections of blood vessels. Coleman and Sampson [2] use gray scale morphology on range data imagery to help mate a robot gripper to an object.

We will develop the extension in the following way. First we introduce the concept of the top surface of a set and the related concept of the umbra of a surface. Then gray scale dilation will be defined as the surface of the dilation of the umbras. From this definition we will proceed to the representation which indicates that gray scale dilation can be computed in terms of a maximum operation and a set of addition operations. A similar plan is followed for erosion which can be evaluated in terms of a minimum operation and a set of subtraction operations.

Of course, having a definition and a means of evaluating the defined operations does not imply that the properties of gray scale dilation and erosion are the same as binary dilation and erosion. To establish that the relationships are identical, we explore some of the relationships between the umbra and surface operation. Our explana-

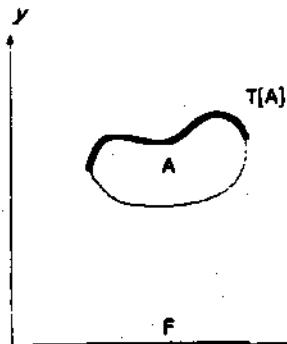


Fig. 12. The concept of top or top surface of a set.

tion shows that umbra and surface operations are essentially inverses of each other. Then we illustrate how the umbra operation is a homomorphism from the gray scale morphology to the binary morphology. Having the homomorphism in hand, all the interesting relationships follow by appropriately unwrapping and wrapping the involved sets or functions.

A. Gray Scale Dilation and Erosion

We begin with the concepts of surface of a set and the umbra of a surface. Suppose a set A in Euclidean N -space is given. We adopt the convention that the first $(N - 1)$ coordinates of the N -tuples of A constitute the spatial domain of A and the N th coordinate is for the surface. For gray scale imagery, $N = 3$. The top or top surface of A is a function defined on the projection of A onto its first $(N - 1)$ coordinates. For each $(N - 1)$ -tuple x , the top surface of A at x is the highest value y such that the N -tuple $(x, y) \in A$. This is illustrated in Fig. 12. If the space we work in is Euclidean, we can express this using the concept of supremum. If the space is discrete, we use the more familiar concept of maximum. Since we have suppressed the underlying space in what follows, we use maximum throughout. The careful reader will want to translate maximum to supremum under the appropriate circumstances.

Definition 46: Let $A \subseteq E^N$ and $F = \{x \in E^{N-1} \mid \text{for some } y \in E, (x, y) \in A\}$. The top or top surface of A , denoted by $T[A]: F \rightarrow E$, is defined by

$$T[A](x) = \max \{y \mid (x, y) \in A\}.$$

Definition 47: A set $A \subseteq E^{N-1} \times E$ is an umbra if and only if $(x, y) \in A$ implies that $(x, z) \in A$ for every $z \leq y$.

For any function f defined on some subset F of Euclidean $(N - 1)$ -space the umbra of f is a set consisting of the surface f and everything below the surface.

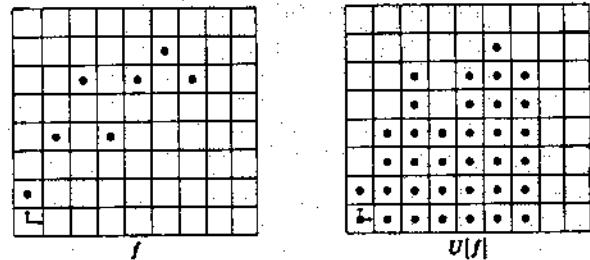
Definition 48: Let $F \subseteq E^{N-1}$ and $f: F \rightarrow E$. The umbra of f , denoted by $U[f]$, $U[f] \subseteq F \times E$, is defined by

$$U[f] = \{(x, y) \in F \times E \mid y \leq f(x)\}.$$

Obviously, the umbra of f is an umbra.

Example: This illustrates a discretized one-dimensional function f defined as a domain consisting of seven

successive column positions and a finite portion of its umbra which lies on or below the function f . The actual umbra has infinite extent below f . The reader should note that because the gray scale morphology so closely involves functions defined on the real line or plane, our example illustrations use the ordinary (x, y) coordinate frame instead of the row column coordinate frame employed in the examples of the binary morphology.

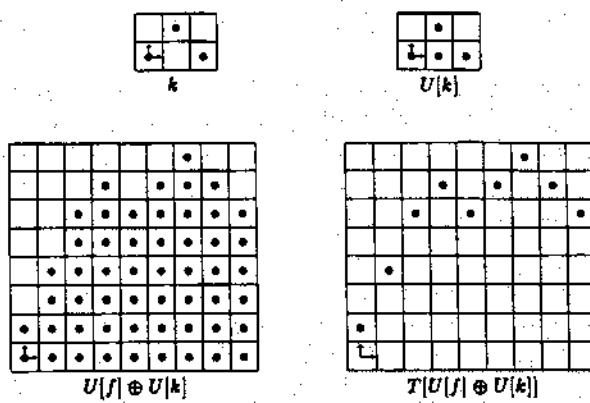


Having defined the operations of taking a top surface of a set and the umbra of a surface, we can define gray scale dilation. The gray scale dilation of two functions is defined as the surface of the dilation of their umbras.

Definition 49: Let $F, K \subseteq E^{N-1}$ and $f: F \rightarrow E$ and $k: K \rightarrow E$. The dilation of f by k is denoted by $f \oplus k$, $f \oplus k: F \oplus K \rightarrow E$, and is defined by

$$f \oplus k = T[U[f] \oplus U[k]].$$

Example: This illustrates a second discretized one-dimensional function k defined on a domain consisting of three successive column positions and a finite portion of its umbra which lies on or below the function k . The dilation of the umbras of f (from the previous example) and k are shown and the surface of the dilation of the umbras of f and k are shown.



The definition of gray scale dilation tells us conceptually how to compute the gray scale dilation, but this conceptual way is not a reasonable way to compute it in hardware. The following theorem establishes that gray scale dilation can be accomplished by taking the maximum of a set of sums. Hence, gray scale dilation has the same complexity as convolution. However, instead of doing the summation of products as in convolution, a maximum of sums is performed.

Proposition 50: Let $f: F \rightarrow E$ and $k: K \rightarrow E$. Then $f \oplus$

$k: F \oplus K \rightarrow E$ can be computed by

$$(f \oplus k)(x) = \max_{\substack{z \in K \\ x - z \in F}} \{ f(x - z) + k(z) \}.$$

Proof. Suppose $z = (f \oplus k)(x)$. Then $z = T[U[f] \oplus U[k]](x)$. By definition of surface,

$$z = \max \{ y | (x, y) \in [U[f] \oplus U[k]] \}.$$

By definition of dilation,

$$z = \max \{ a + b | \text{for some } u \in K \text{ satisfying } x - u \in F, (x - u, a) \in U[f] \text{ and } (u, b) \in U[k] \}.$$

By definition of umbra, the largest a such that $(x - u, a) \in U[f]$ is $a = f(x - u)$. Likewise, the largest b such that $(u, b) \in U[k]$ is $b = k(u)$. Hence

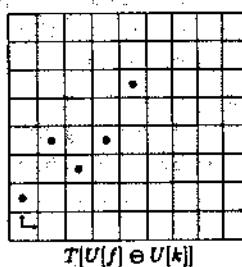
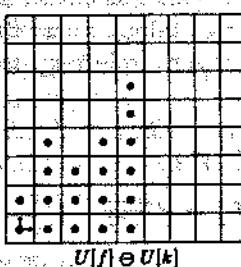
$$\begin{aligned} z &= \max \{ f(x - u) + k(u) | u \in K, (x - u) \in F \} \\ &= \max_{\substack{u \in K \\ (x - u) \in F}} \{ f(x - u) + k(u) \}. \end{aligned}$$

The definition for gray scale erosion proceeds in a similar way to the definition of gray scale dilation. The gray scale erosion of one function by another is the surface of the binary erosions of the umbra of one with the umbra of the other.

Definition 51: Let $F \subseteq E^{N-1}$ and $K \subseteq E^{N-1}$. Let $f: F \rightarrow E$ and $k: K \rightarrow E$. The erosion of f by k is denoted by $f \ominus k, f \ominus k: F \ominus K \rightarrow E$, and is defined by

$$f \ominus k = T[U[f] \ominus U[k]].$$

Example: Using the same function f and k of the previous example, illustrated here is the erosion of f by k by taking the surface of the erosion of the umbra of f by the umbra of k .



Evaluating a gray scale erosion is accomplished by taking the minimum of a set of differences. Hence its complexity is the same as dilation. Its form is like correlation with the summation of correlation replaced by the minimum operation and the product of correlation replaced by a subtraction operation. If the underlying space is Euclidean, substitute infimum for minimum.

Proposition 52: Let $f: F \rightarrow E$ and $k: K \rightarrow E$. Then $f \ominus k: F \ominus K \rightarrow E$ can be computed by $(f \ominus k)(x) = \min_{z \in K} \{ f(x + z) - k(z) \}$.

Proof. Suppose $z = (f \ominus k)(x)$. Then, $z = T[U[f] \ominus U[k]](x)$. By definition of surface, $z = \max \{ y | (x, y) \in U[f] \ominus U[k] \}$. By definition of erosion

$$\begin{aligned} z &= \max \{ y | \text{for every } (u, v) \\ &\in U[k], (x, y) + (u, v) \in U[f] \}. \end{aligned}$$

By definition of umbra,

$$\begin{aligned} z &= \max \{ y | \text{for every } u \in K, v \leq k(u), y \\ &+ v \leq f(x + u) \} \\ &= \max \{ y | \text{for every } u \in K, v \\ &\leq k(u), y \leq f(x + u) - v \}. \end{aligned}$$

But $y \leq f(x + u) - v$ for every $v \leq k(u)$ implies $y \leq f(x + u) - k(u)$. Hence,

$$z = \max \{ y | \text{for every } u \in K, y \leq f(x + u) - k(u) \}.$$

But $y \leq f(x + u) - k(u)$ for every $u \in K$ implies

$$y \leq \min_{u \in K} [f(x + u) - k(u)].$$

Now,

$$\begin{aligned} z &= \max \left\{ y | y \leq \min_{u \in K} [f(x + u) - k(u)] \right\} \\ &= \min_{u \in K} [f(x + u) - k(u)]. \end{aligned}$$

Fig. 13 illustrates an example of gray scale dilation and erosion.

The basic relationship between the surface and umbra operations is that they are, in a certain sense, inverses of each other. More precisely, the surface operation will always undo the umbra operation. That is, the surface operation is an inverse to the umbra operation as given in the next proposition.

Proposition 53: Let $F \subseteq E^{N-1}$ and $f: F \rightarrow E$. Then $T[U[f]] = f$.

Proof. Let $y \in T[U[f]](x)$. Then $y = \max \{ z | (x, z) \in U[f] \}$. Now $(x, z) \in U[f]$ implies $z \leq f(x)$. Also, $(x, f(x)) \in U[f]$. Thus y cannot get larger than $f(x)$ and since $(x, f(x)) \in U[f]$, y can get as large as $f(x)$. Thus $y = f(x)$.

Corollary 54: $U[T[U[f]]] = U[f]$.

However the umbra operation is not an inverse to the surface operation. Without any constraints on the set A , the strongest statement which can be made is that the umbra of the surface of A contains A . This is illustrated in Fig. 14.

Proposition 55: Let $A \subseteq E^N$. Then $A \subseteq U[T[A]]$.

Proof. Let $x \in E^{N-1}$ and $y \in E$. Suppose, $(x, y) \in A$. Let $z = T[A](x) = \max \{ v | (x, v) \in A \}$. Hence, $z \geq y$. But by definition of the umbra operation $z = T[A](x)$ implies $(x, w) \in U[T[A]]$ for all $w \leq z$. In particular, $y \leq z$. Hence, $(x, y) \in U[T[A]]$.

When the set A is an umbra, then the umbra of the surface of A is itself A . In this case the umbra operation is an inverse to the surface operation.

Proposition 56: If A is an umbra, then $A = U[T[A]]$.

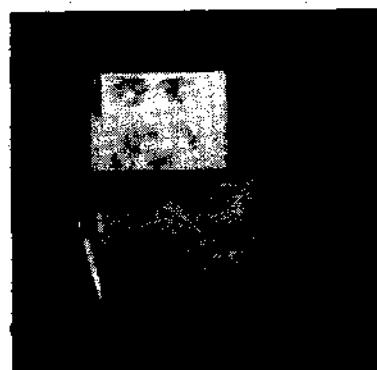
Proof. By Proposition 54, $A \subseteq U[T[A]]$. So we just need to show that $A \supseteq U[T[A]]$. Suppose $(x, y) \in$



(a)



(b)



(c)

Fig. 13. A woman's face in the image form and in a perspective projection surface plot form. This image is morphologically processed with a paraboloid structuring element given by $6(8 - r^2 - c^2)$, $-2 \leq r \leq 2$, $-2 \leq c \leq 2$. (b) The erosion of the girl's face in image form and perspective projection surface plot form. (c) The dilation of the girl's face in image form and perspective projection surface plot form.

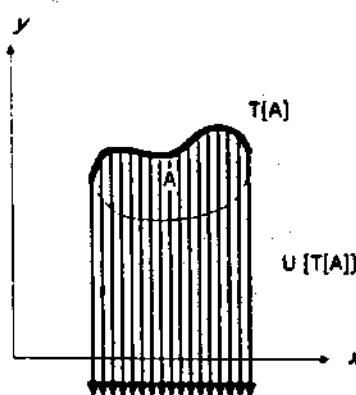


Fig. 14. The umbra of the top surface of a set.

$U[T[A]]$ and A is an umbra. By definition of the umbra operation $(x, y) \in U[T[A]]$ implies $y \leq T[A](x)$ and there exists some z such that $(x, z) \in A$. Now if there exists some z such that $(x, z) \in A$, $(x, T[A](x)) \in A$. Since A is an umbra, $(x, T[A](x)) \in A$ implies that $(x, w) \in A$ for every $w \leq T[A](x)$. In particular $y \leq T[A](x)$. Hence $(x, y) \in A$.

Having established that the surface operation is always an inverse to the umbra operation and that the umbra operation is the inverse to the surface operation when the set being operated on itself is an umbra, we are almost ready to develop the umbra homomorphism theorem. First we need to establish that the dilation of one umbra by another is an umbra and that the erosion of one umbra by another is also an umbra.

Proposition 57: Suppose A and B are umbras. Then $A \oplus B$ and $A \ominus B$ are umbras.

Proof: Suppose $(x, y) \in A \oplus B$. Let $w \leq y$. We need to demonstrate that $(x, w) \in A \oplus B$. By definition of dilation, $(x, y) \in A \oplus B$ implies that there exists $(u, v) \in B$ such that $(x - u, y - v) \in A$. Now $w \leq y$ implies $w - v \leq y - v$. And A is an umbra so that $(x - u, y - v) \in A$ and $w - v \leq y - v$ implies $(x - u, w - v) \in A$. But by definition of dilation, $(x - u, w - v) \in A$ and $(u, v) \in B$ implies $(x, w) \in A \oplus B$ which means that $A \oplus B$ is an umbra.

Suppose $(x, y) \in A \ominus B$. Let $w \leq y$. We need to demonstrate that $(x, w) \in A \ominus B$. By definition of erosion, $(x, y) \in A \ominus B$ implies that for every $(u, v) \in B$, $(x, y) + (u, v) = (x + u, y + v) \in A$. Now $w \leq y$ implies $w + v \leq y + v$. Since A is an umbra and $(x + u, y + v) \in A$ and $w + v \leq y + v$, then $(x + u, w + v) \in A$. But by definition of erosion if $(x, w) + (u, v) \in A$ for every $(u, v) \in B$ then $(x, w) \in A \ominus B$. Hence $A \ominus B$ is an umbra.

Now we are ready for the umbra homomorphism theorem which states that the operation of taking an umbra is a homomorphism from the gray scale morphology to the binary morphology.

Umbra Homomorphism Theorem 58: Let $F, K \subseteq E^{N-1}$ and $f: F \rightarrow E$ and $k: K \rightarrow E$. Then

$$1) U[f \oplus k] = U[f] \oplus U[k]$$

and

$$2) U[f \ominus k] = U[f] \ominus U[k]$$

Proof: 1) $f \oplus k = T[U[f] \oplus U[k]]$ so that $U[f \oplus k] = U[T[U[f] \oplus U[k]]]$. But $U[f] \oplus U[k]$ is an umbra and for sets which are umbras the umbra operation undoes the surface operation. Hence $U[f \oplus k] = U[T[U[f] \oplus U[k]]] = U[f] \oplus U[k]$.

2) $f \ominus k = T[U[f] \ominus U[k]]$ so that $U[f \ominus k] = U[T[U[f] \ominus U[k]]]$. But $U[f] \ominus U[k]$ is an umbra and for sets which are umbras, the umbra operation undoes the surface operation. Hence,

$$U[f \ominus k] = U[T[U[f] \ominus U[k]]] = U[f] \ominus U[k].$$

To illustrate how the umbra homomorphism property is

used to prove relationships by first wrapping the relationship by re-expressing it in terms of umbra and surface operations and then transforming it through the umbra homomorphism property and finally by unwrapping it using the definitions of gray scale dilation and erosion, we state and prove the commutivity and associativity of gray scale dilation and the chain rule for gray scale erosion.

Proposition 59: $f \oplus k = k \oplus f$.

Proof:

$$\begin{aligned} f \oplus k &= T[U[f] \oplus U[k]] \\ &= T[U[k] \oplus U[f]] \\ &= k \oplus f. \end{aligned}$$

Proposition 60: $k_1 \oplus (k_2 \oplus k_3) = (k_1 \oplus k_2) \oplus k_3$.

Proof:

$$\begin{aligned} k_1 \oplus (k_2 \oplus k_3) &= T[U[k_1] \oplus U[k_2 \oplus k_3]] \\ &= T[U[k_1] \oplus (U[k_2] \oplus U[k_3])] \\ &= T[(U[k_1] \oplus U[k_2]) \oplus U[k_3]] \\ &= T[U[k_1 \oplus k_2] \oplus U[k_3]] \\ &= (k_1 \oplus k_2) \oplus k_3. \end{aligned}$$

Proposition 61: $(f \ominus k_1) \ominus k_2 = f \ominus (k_1 \oplus k_2)$.

Proof:

$$\begin{aligned} (f \ominus k_1) \ominus k_2 &= T[U[f \ominus k_1] \ominus U[k_2]] \\ &= T[(U[f] \ominus U[k_1]) \ominus U[k_2]] \\ &= T[U[f] \ominus (U[k_1] \oplus U[k_2])] \\ &= T[U[f] \ominus U[k_1 \oplus k_2]] \\ &= f \ominus (k_1 \oplus k_2). \end{aligned}$$

Gray scale opening and closing are defined in an analogous way to opening and closing in the binary morphology and they have similar properties.

Definition 62: Let $f: F \rightarrow E$ and $k: K \rightarrow E$. The gray scale opening of f by structuring element k is denoted by $f \circ k$ and is defined by $f \circ k = (f \ominus k) \oplus k$.

Definition 63: Let $f: F \rightarrow E$ and $k: K \rightarrow E$. The gray scale closing of f by structuring element k is denoted by $f \bullet k$ and is defined by $f \bullet k = (f \oplus k) \ominus k$.

Fig. 15 shows an example of gray scale opening and closing.

To prove the idempotency of gray scale opening and closing, we need the following property relating functions to their umbrae.

Proposition 64: Let $f: F \rightarrow E$ and $g: G \rightarrow E$. Suppose $F \subseteq G$. Then $f \leq g$ if and only if $U[f] \subseteq U[g]$.

Proof: Suppose $f \leq g$. Let $(x, y) \in U[f]$. Then by definition of umbra, $y \leq f(x)$. But $x \in F$ and $F \subseteq G$ so that $x \in G$. By supposition, $f(x) \leq g(x)$. Hence, $y \leq g(x)$. Now by definition of umbra, $(x, y) \in U[g]$.

Suppose $U[f] \subseteq U[g]$. Let $y = f(x)$. Certainly, $(x,$

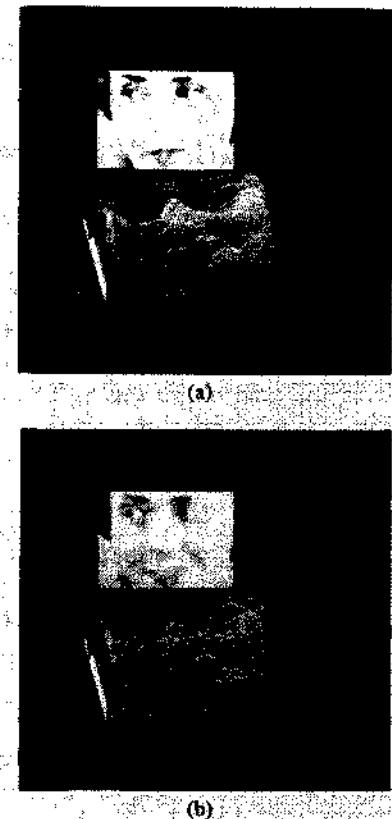


Fig. 15. The gray scale opening and closing operation. (a) The gray scale opening of the girl's face in image form and in perspective projection surface plot form. The structuring element is the paraboloid described in Fig. 13. (b) The gray scale closing of the girl's face in image form and in perspective projection surface plot form.

$y) \in U[f]$. But $U[f] \subseteq U[g]$ so that $(x, y) \in U[g]$. Now by definition of umbra, $y \leq g(x)$.

Having this property, the analog to Proposition 33 follows.

Proposition 65: $g \leq f \ominus k$ if and only if $f \geq g \oplus k$.

Proof: $g \leq f \ominus k$ if and only if $U[g] \subseteq U[f \ominus k]$. But $U[f \ominus k] = U[f] \ominus U[k]$. Now $U[g] \subseteq U[f] \ominus U[k]$ if and only if $U[f] \supseteq U[g] \oplus U[k]$. But $U[g] \oplus U[k] = U[g \oplus k]$. Finally, $U[f] \supseteq U[g \oplus k]$ if and only if $f \geq g \oplus k$.

Another property which is immediately obvious is that if one set is contained in a second, then the surface of the first will be no higher at each point than the surface of the second.

Proposition 66: Let $A \subseteq E^{N-1} \times E$ and $D \subseteq E^{N-1} \times E$. Then $A \subseteq D$ implies $T[A](x) \leq T[D](x)$.

Proof: Let $x \in E^{N-1}$ be given. Then, since $A \subseteq D$,

$$T[A](x) = \max_{(x,z) \in A} z \leq \max_{(x,z) \in D} z = T[D](x).$$

From this fact, it quickly follows that the gray scale opening of a function must be no larger than the function at each point in their common domain. This is the gray scale analog to the antiextensivity property of the binary morphology opening.

Proposition 67: $(f \circ k)(x) \leq f(x)$ for every $x \in F \circ K$.

Proof:

$$\begin{aligned} f \circ k = (f \oplus k) \ominus k &= T[U(f \oplus k) \ominus U[k]] \\ &= T[(U[f] \oplus U[k]) \ominus U[k]]. \end{aligned}$$

But $(U[f] \oplus U[k]) \ominus U[k] \subseteq U[f]$, hence by Proposition 63,

$$T[(U[f] \oplus U[k]) \ominus U[k])(x) \leq T[U[f]](x)$$

for every $x \in (F \oplus K) \ominus K$. Since $T[U[f]] = f$, $T[(U[f] \oplus U[k]) \ominus U[k]](x) \leq f(x)$.

Likewise, the gray scale closing of a function must be no smaller than the function at each point in their common domain. This is the gray scale analog to the extensivity property of the binary morphology closing.

Proposition 68: $f(x) \leq (f \bullet k)(x)$ for every $x \in F$.

Proof:

$$\begin{aligned} f \bullet k = (f \oplus k) \ominus k &= T[U[f \oplus k] \ominus U[k]] \\ &= T[(U[f] \oplus U[k]) \ominus U[k]]. \end{aligned}$$

But $(U[f] \oplus U[k]) \ominus U[k] \supseteq U[f]$; hence $T[(U[f] \oplus U[k]) \ominus U[k]](x) \geq T[U[f]](x)$ for every $x \in F$. Since $T[U[f]] = f$, $f(x) \leq T[(U[f] \oplus U[k]) \ominus U[k]](x)$.

Now the idempotency property of opening and closing can be proved by the umbra homomorphism theorem.

Proposition 69: $(f \circ k) \circ k = f \circ k$.

Proof:

$$\begin{aligned} (f \circ k) \circ k &= T[(U[f \circ k] \ominus U[k]) \oplus U[k]] \\ &= T[((((U[f] \ominus U[k]) \oplus U[k]) \ominus U[k]) \\ &\quad \oplus U[k])] \\ &= T[(U[f] \circ U[k]) \ominus U[k]] \\ &= T[U[f] \circ U[k]] \\ &= T[(U[f] \ominus U[k]) \oplus U[k]] \\ &= T[U[f \ominus k] \oplus U[k]] \\ &= T[U[(f \ominus k) \oplus k]] \\ &= T[U[f \circ k]] \\ &= f \circ k. \end{aligned}$$

Proposition 70: $(f \bullet k) \bullet k = f \bullet k$.

Proof:

$$\begin{aligned} (f \bullet k) \bullet k &= T[(U[f \bullet k] \oplus U[k]) \ominus U[k]] \\ &= T[((((U[f] \oplus U[k]) \ominus U[k]) \oplus U[k]) \\ &\quad \ominus U[k])] \\ &= T[(U[f] \bullet U[k]) \bullet U[k]] \\ &= T[U[f] \bullet U[k]] \\ &= T[(U[f] \oplus U[k]) \ominus U[k]] \\ &= T[U[(f \oplus k) \ominus k]] \\ &= T[U[f \bullet k]] \\ &= f \bullet k. \end{aligned}$$

There is a geometric interpretation to the gray scale opening and to the gray scale closing in the same manner that there is a geometric meaning to the binary morphological opening and closing (Propositions 44 and 45). To obtain the opening of f by a paraboloid structuring element, for example, take the paraboloid, apex up, and slide it under all the surface of f pushing it hard up against the surface. The apex of the paraboloid may not be able to touch all points of f . For example, if f has a spike narrower than the paraboloid, the top of the apex may only reach as far as the mouth of the spike. The opening is the surface of the highest points reached by any part of the paraboloid as it slides under all the surface of f . The formal statement of this is given in Proposition 71.

Proposition 71:

$$f \circ k = T\left[\bigcup_{\{y | U[k], y \in U[f]\}} U[k]_y\right].$$

Proof:

$$\begin{aligned} f \circ k &= T[U[f \ominus k] \oplus U[k]] \\ &= T[(U[f] \ominus U[k]) \oplus U[k]] \\ &= T[U[f] \circ U[k]] \\ &= T\left[\bigcup_{\{z | (U[k])_z \subseteq U[f]\}} (U[k])_z\right]. \end{aligned}$$

We have not mentioned the duality relationship between gray scale dilation and erosion. We need this in order to give the geometric interpretation to closing. The duality relationship is analogous to the relationship given in Theorem 25. Before stating and proving it, we need the definition of gray scale reflection.

Definition 72: Let $f: F \rightarrow E$. The reflection of f is denoted by $\check{f}: \check{F} \rightarrow E$, and is defined by $\check{f}(x) = f(-x)$.

Gray Scale Dilation Erosion Duality Theorem 73: Let $f: F \rightarrow E$ and $k: K \rightarrow E$. Let $x \in (F \oplus K) \cap (F \ominus \check{K})$ be given. Then $-(f \oplus k)(x) = ((-\check{f}) \ominus \check{k})(x)$.

Proof:

$$\begin{aligned} -(f \oplus k)(x) &= -\max_{\substack{z \in K \\ x-z \in F}} [f(x-z) + k(z)] \\ &= \min_{\substack{z \in K \\ x-z \in F}} [-f(x-z) - k(z)] \\ &= \min_{\substack{z \in K \\ x+z \in F}} [-f(x+z) - \check{k}(z)] \\ &= ((-\check{f}) \ominus \check{k})(x). \end{aligned}$$

It follows immediately from the gray scale dilation and erosion duality that there is a gray scale opening and closing duality.

Gray Scale Opening and Closing Duality Theorem 74: $-(f \circ k) = (-\check{f}) \bullet \check{k}$.

Proof.

$$\begin{aligned}
 -(f \circ k) &= -((f \ominus k) \oplus k) \\
 &= ((-f) \oplus k) \ominus k \\
 &= (-f) \bullet k.
 \end{aligned}$$

Having the gray-scale opening and closing duality, we immediately have $f \bullet k = -((-f) \circ k)$. In essence, this means that we can think of closing like opening. To close f with a paraboloid structuring element, we take the reflection of the paraboloid in the sense of Definition 72, turn it upside down (apex down), and slide it all over the top of the surface of f . The closing is the surface of all the lowest points reached by the sliding paraboloid.

V. SUMMARY

We have developed the basic relationships in binary morphology and have then developed the extensions of these relationships in gray scale morphology. We have shown that morphological openings are increasing, antiextensive, translation invariant, and idempotent. We have shown that morphological closings are increasing, extensive, translation invariant, and idempotent. For further algebraic depth on opening and closings, see [18] or [31].

We intend to publish two follow-on tutorials to the present one. The first will discuss a variety of topics including sieves, sampling, morphologic topography, thickenings, thinnings, boundaries, skeletons, connectivity, convexity, morphologic derivative estimation, and bounding derivatives by gray scale morphologic openings and closings. The second will be on application where we will discuss morphologic solutions to a variety of industrial vision problems.

REFERENCES

- [1] K. E. Batcher, "Design of a massively parallel processor," *IEEE Trans. Comput.*, vol. C-29, pp. 836-840, Sept. 1980.
- [2] E. N. Coleman and R. E. Sampson, "Acquisition of randomly oriented workpieces through structure mating," in *Proc. Computer Vision Pattern Recognition Conf.*, San Francisco, CA, June 19-23, pp. 350-357.
- [3] M. Duff, "Parallel processors for digital image processing," in *Advances in Digital Image Processing*, P. Stucki, Ed. New York: Plenum, 1979, pp. 265-279.
- [4] M. J. B. Duff, D. M. Watson, T. M. Fountain, and G. K. Shaw, "A cellular logic array for image processing," *Pattern Recognition*, vol. 5, 1973.
- [5] A. Favre, R. Muggli, A. Stucki, and P. Bonderet, "Application of morphologic filters in the assessment of platelet thrombus deposition in cross sections of blood vessels," in *Proc. 4th Scandinavian Conf. Image Analysis*, Trondheim, Norway, June 17-20, 1985, pp. 629-640.
- [6] F. Gerritsen and L. G. Aardema, "Design and use of DIP-1: A fast flexible and dynamically microprogrammable image processor," *Pattern Recognition*, vol. 14, pp. 319-330, 1981.
- [7] F. A. Gerritsen and P. W. Verbeek, "Implementation of cellular logic operators using 3×3 convolution and table lookup hardware," *Comput. Vision, Graphics, Image Processing*, vol. 27, pp. 115-123, 1984.
- [8] M. J. E. Golay, "Hexagonal parallel pattern transformations," *IEEE Trans. Comput.*, vol. C-18, pp. 733-740, 1969.
- [9] D. Graham and P. E. Norgren, "The Diff3 analyzer: A parallel/serial Golay image processor," in *Real Time Medical Image Processing*, M. Onoe, K. Preston, and A. Rosenfeld, Eds. London: Plenum, 1980, pp. 163-182.
- [10] H. Hadwiger, *Vorlesungen über Inhalt, Oberfläche und Isoperimetrie*. Berlin: Springer, 1957.
- [11] R. M. Haralick, "A reconfigurable systolic network in computer vision," *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Miami Beach, FL, Nov. 18-20, 1985, pp. 507-515.
- [12] M. J. Kimmel, R. S. Jaffe, J. R. Manderville, and M. A. Lavin, "MITE: Morphic image transform engine, an architecture for reconfigurable pipelines of neighborhood processors," in *Proc. IEEE Comput. Soc. Workshop Computer Architecture for Pattern Analysis and Image Database Management*, Miami Beach, FL, Nov. 18-20, 1985, pp. 493-500.
- [13] R. A. Kirsch, L. Cahn, C. Ray, and G. H. Urban, "Experiments in processing pictorial information with a digital computer," in *Proc. Eastern Joint Comput. Conf.*, 1957, pp. 221-229.
- [14] J. C. Klein and J. Serra, "The texture analyzer," *J. Microscopy*, vol. 95, pp. 349-356, 1977.
- [15] B. Kruse, "Design and implementation of a picture processor," *Science and Technology dissertation*, Univ. Linköping, Linköping, Sweden, Rep. 13, 1977.
- [16] P. F. Leonard, "Pipeline architectures for real-time machine vision," *IEEE Comput. Soc. Workshop Computer Architecture for Pattern Analysis and Image Database Management*, Miami Beach, FL, Nov. 18-20, 1985, pp. 502-505.
- [17] G. Matheron, *Éléments Pour une Théorie des Milieux Poreux*. Paris: Masson, 1965.
- [18] —, *Random Sets and Integral Geometry*. New York: Wiley, 1975.
- [19] D. L. McCuberry and R. M. Lougheed, "Morphological image analysis using a raster pipeline processor," in *IEEE Comput. Soc. Workshop Computer Architecture for Pattern Analysis and Image Database Management*, Miami Beach, FL, Nov. 18-20, 1985, pp. 444-452.
- [20] H. Minkowski, "Volumen und Oberfläche," *Math. Ann.*, vol. 57, pp. 447-495, 1903.
- [21] G. A. Moore, "Automatic scanning and computer processes for the quantitative analysis of micrographs and equivalent subjects," in *Pictorial Pattern Recognition*, G. C. Cheng et al., Eds. Washington, DC: Thompson, 1968, pp. 275-326.
- [22] Y. Nakagawa and A. Rosenfeld, "A note on the use of local min and max operations in digital picture processing," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 632-635, Aug. 1978.
- [23] S. Peleg, J. Naor, R. Hartley, and D. Avnir, "Multiple resolution texture analysis and classification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 518-523, 1984.
- [24] S. Peleg and A. Rosenfeld, "A min max medial axis transformation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, pp. 206-210, 1981.
- [25] J. L. Potter, "MPP architecture and programming," in *Multicomputers and Image Processing*, K. Preston and L. Uhr, Eds. New York: Academic, 1982, pp. 275-290.
- [26] —, "Image processing on the massively parallel processor," *Computer*, vol. 16, no. 1, pp. 62-67, Jan. 1983.
- [27] W. K. Pratt, "A pipeline architecture for image processing and analysis," in *Proc. IEEE Comput. Soc. Workshop Computer Architecture for Pattern Analysis and Image Database Management*, Miami Beach, FL, Nov. 18-20, 1985, pp. 516-520.
- [28] K. Preston, Jr., "Machine techniques for automatic identification of binucleate lymphocyte," in *Proc. Fourth Int. Conf. Medical Electronics*, Washington, DC, July 1961.
- [29] —, "Application of cellular automata to biomedical image processing," in *Computer Techniques in Biomedicine and Medicine*. Philadelphia, PA: Auerbach, 1973.
- [30] J. Serra, "Stereology and structuring elements," *J. Microscopy*, pp. 93-103, 1972.
- [31] —, *Image Analysis and Mathematical Morphology*. London: Academic, 1982.
- [32] S. R. Sternberg, "Parallel architectures for image processing," in *Proc. IEEE COMPSAC*, Chicago, IL, 1979.
- [33] —, "Cellular computers and biomedical image processing," in *Biomedical Images and Computers*, J. Sklansky and J. C. Bisconte, Eds. Berlin: Springer-Verlag, 1982, pp. 294-319 (also presented at United States-France Seminar on Biomedical Image Processing, St. Pierre de Chartreuse, France, May 27-31, 1980).
- [34] —, "Pipeline architectures for image processing," in *Multicomputers and Image Processing*, K. Preston and L. Uhr, Eds. New York: Academic, 1982, pp. 291-305.

- [35] —, "Esoteric iterative algorithms," in *Proc. Second Int. Conf. Image Analysis and Processing*, Selva di Fasano, Brindisi, Italy, Nov. 15-18, 1982.
- [36] —, "Biomedical image processing," *Computer*, vol. 16, no. 1, pp. 22-34, Jan. 1983.
- [37] S. H. Unger, "A computer oriented to spatial problems," *Proc. IRE*, vol. 46, pp. 1744-1750, 1958.
- [38] M. Werman and S. Peleg, "Min-max operators in texture analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 730-733, Nov. 1985.
- [39] S. Wilson, "The Pixie-5000—A systolic array processor," in *Proc. IEEE Comput. Soc. Workshop Computer Architecture for Pattern Analysis and Image Database Management*, Miami Beach, FL, Nov. 18-20, 1985, pp. 477-483.



Robert M. Haralick (S'62-M'69-SM'76-F'84) was born in Brooklyn, NY, on September 30, 1943. He received the B.A. degree in mathematics from the University of Kansas, Lawrence, in 1964, the B.S. degree in electrical engineering in 1966, the M.S. degree in electrical engineering in 1967, and the Ph.D. degree from the University of Kansas in 1969.

He has worked with Autonetics and IBM. In 1965 he worked for the Center for Research, University of Kansas, as a Research Engineer and in 1969 he joined the faculty of the Department of Electrical Engineering there where he last served as a Professor from 1975 to 1978. In 1979 he joined the faculty of the Department of Electrical Engineering at Virginia Polytechnic Institute and State University where he was a Professor and Director of the Spatial Data Analysis Laboratory. From 1984 to 1986 he served as Vice President of Research at Machine Vision International, Ann Arbor, MI. He now holds the Boeing Clairmont Egertson chaired professorship in the Department of Electrical Engineering, University of Washington, Seattle. He has done research in pattern recognition, multi-image processing, remote sensing, texture analysis, data compression, clustering, artificial intelligence, and general systems theory, and has published over 180 papers. He is responsible for the development of GIPSY (General Image Processing System), a multi-image processing package which runs on a minicomputer system.

Dr. Haralick is a member of the Association for Computing Machinery, Sigma Xi, the Pattern Recognition Society, and the Society for General Systems Research.



Stanley R. Sternberg received the Bachelor of Science degree in electrical engineering from Drexel Institute of Technology, Philadelphia, PA, in 1962, the Master of Science degree in computer sciences, and the Ph.D. degree in industrial engineering, both from the University of Michigan, Ann Arbor, in 1965 and 1971, respectively.

He is Chief Technical Officer of Machine Vision International Corporation, Ann Arbor. In 1981 he founded the corporation under the name CytoSystems Corporation to produce and market

video rate processors for industrial inspection, robot vision, and biomedical image analysis. He is the holder of 20 U.S. patents for his developments of novel methods of infrared imaging, image processing and laser line-of-sight data transmissions. He is the recipient of two Civil Service Outstanding Service Awards for his contributions as Manager of quality control and reliability for a major United States Air Force program. He is a founder of the Computer and Image Processing Research Network of the University of Michigan and a director of the Automated Vision Association of the Robotic Industries Association. From 1974 to 1981, he was Senior Research Scientist and head of the Department of Computer Design at the Environmental Research Institute of Michigan (ERIM) where he directed and conducted research and development of cellular computers for image processing. He designed and built a highly parallel real-time image processor, the Cytocomputer, investigated high level languages for parallel image processing, and applied these concepts to biomedical and industrial image analysis. He is Adjunct Associate Professor of Electrical and Computer Engineering at the University of Michigan where he teaches cellular computers and image processing on a part-time basis.

Dr. Sternberg is a member of Tau Beta Pi, Sigma Xi, and Eta Kappa Nu.



Xinhua Zhuang graduated from Peking University, Peking, China, in 1963, after a four-year undergraduate program and a two-year graduate program in mathematics.

Before 1983 he served as a Senior Research Engineer in the Computing Technique Institute, Hangzhou, China. He was a Visiting Scholar of Electrical Engineering at the Virginia Polytechnic Institute and State University, Blacksburg, from 1983 to 1984, a Visiting Scientist of Electrical and Computer Engineering at the University of Michigan, Ann Arbor, granted by Machine Vision International, Ann Arbor, from 1984 to 1985. He was selected as a consultant to the Advisory Group for Aerospace Research and Development, NATO, in 1985. From 1985 to 1986 he was a Visiting Research Professor with the Coordinated Science Laboratory and a Visiting Professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. From 1986 to 1987 he was a Visiting Scientist in the Department of Electrical Engineering, University of Washington, Seattle. His professional interests lie in applied mathematics, image processing, computer and robotic vision, artificial intelligence, and computer architecture. He is a contributor (with E. Ostevoord and R. M. Haralick) of the book *Image Recovery: Theory and Application*, edited by Henry Stark; Editor (with R. M. Haralick) of the book *Consistent Labeling Problems in Pattern Recognition*; and author (with T. S. Huang and R. M. Haralick) of the planned book titled *Image Time Sequence Motion Analysis*.

Reprinted from *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume PAMI-2, Number 2, March 1980, pages 111-126. Copyright © 1980 by The Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

Transformations of shapes are well understood [1], [2], [3]. However, the direct matching of two shapes is not well understood. In this paper we propose a structural model of shape which characterizes a shape by its primitives and their interrelationships. This model is used to define a matching procedure that uses a tree search with look-ahead to find mappings from a prototype shape to a candidate shape.

Shape description and recognition is an important problem in scene analysis. Our approach to shape description is a formal model of a shape consisting of a set of primitives, their properties, and their interrelationships. The primitives are the simple parts and intrusions of the shape which can be derived through the graph-theoretic clustering procedure described in [31]. The interrelationships are two ternary relations on the primitives: the intrusion relation which relates two simple parts that join to the intrusion they surround and the protrusion relation which relates two intrusions to the protrusion between them. Using this model, a shape matching procedure that uses a tree search with look-ahead to find mappings from a prototype shape to a candidate shape has been developed. An experimental Snobol4 implementation has been used to test the program on hand-printed character data with favorable results.

Abstract—Shape description and recognition is an important and interesting problem in scene analysis. Our approach to shape description is a formal model of a shape consisting of a set of primitives, their properties, and their interrelationships. The primitives are the simple parts and intrusions of the shape which can be derived through the graph-theoretic clustering procedure described in [31]. The interrelationships are two ternary relations on the primitives: the intrusion relation which relates two simple parts that join to the intrusion they surround and the protrusion relation which relates two intrusions to the protrusion between them. Using this model, a shape matching procedure that uses a tree search with look-ahead to find mappings from a prototype shape to a candidate shape has been developed. An experimental Snobol4 implementation has been used to test the program on hand-printed character data with favorable results.

Index Terms—Decomposition, matching, relational description, relaxation, shape description, shape recognition, tree search.

I. INTRODUCTION

SHAPE description and characterization is an important problem in scene analysis. There have been many different analytic methods used in shape analysis including transformations and decompositions of the boundary of the shape, and partitions of the interior of the shape into simple pieces.

Manuscript received October 30, 1978; revised April 13, 1979. This work was supported by the National Science Foundation under Grant MCS77-23945.

The author is with the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061.

Shape descriptions have included numeric values, feature vectors, character strings, trees, and graphs. The major problem seems to be the lack of a mathematical model for characterizing a shape. Our approach to shape description is to construct a topological model of a shape which consists of a set of primitives, their properties, and their interrelationships. The primitives we use are the simple parts (near-convex pieces) and intrusions of the shape which can be derived through a decomposition procedure described in a previous paper [31]. The interrelationships are two ternary relations on the primitives: the intrusion relation and the protrusion relation. The intrusion relation is a set of triples of the form (s_1, i, s_2) where the two simple parts s_1 and s_2 touch or nearly touch and form part of the boundary of intrusion i . The protrusion relation is a set of triples of the form (i_1, s, i_2) where simple part s protrudes between the two intrusions i_1 and i_2 . These two relations seem to satisfactorily characterize a shape in a manner that agrees with human intuition on familiar shapes such as hand-printed characters. Our topological approach to shape matching is that one shape matches another if there is a mapping from the primitives of the first shape to the primitives of the second shape that preserves the interrelationships. In this approach, the metric and statistical properties of the shape and its parts are suppressed.

In this paper, we define a formal shape model and a shape matching procedure that uses a tree search with look-ahead to find mappings from a prototype shape to a candidate shape.

Section II discusses related work, Section III describes a general structural model for shape and defines our particular relational model, Section IV discusses shape matching, and Section V describes an experimental shape matching system.

II. RELATED WORK

There have been several different approaches to the shape recognition problem. (For a complete survey, see Pavlidis [26].) On the statistical side, shapes have been described by such features as area, perimeter, moments, and coefficients of Fourier series. Description by moments includes the work of Alt [1] and Hu [19] among others. Description using coefficients of Fourier series includes the work of Zahn and Roskies [36], Richard and Hemami [28], Persoon and Fu [27], and Granlund [13]. Agrawala and Kulkarni [2] present a sequential one-pass algorithm for extracting simple shape features such as perimeter, area, and moments. All of these approaches work with a discrete representation of the boundary of the shape.

Much of the recent work has been structural in nature. Blum [4] suggests the application of the medial axis transformation which transforms a shape into a line drawing representing its "skeleton." The skeleton can be used to derive properties of the shape and, together with information about the distance of the boundary from the skeleton, allows reconstruction of the shape. A second structural approach involves the decomposition of the boundary of the shape into a sequence of line segments. Work in this area includes the Freeman chain code [11], the analysis of convex blobs by computing dominant points [20], [23], [29], [30], and Davis' work with angles, sides, and symmetry [6]-[8].

Also in the structural domain is the work on syntactic shape recognition. This includes Pavlidis and Ali's syntactic analysis of shapes [24] using regular expressions over the terminal symbols QUAD (arcs that can be approximated by quadratic curves), TRUS (sharp protrusions or intrusions), LINE (long linear segments), and BREAK (short segments). Also using the syntactic approach are Horowitz [18] and Lozano-Perez [21], both of whom parse piecewise linear approximations of one-dimensional waveforms to determine the structure of the peaks and valleys and Fu and Lu [12] who encode line patterns as strings and use error-correcting parsers to determine the distance from an input pattern to grammars describing each of the possible pattern classes. A clustering algorithm then determines which class the input pattern belongs to.

Another recent effort in syntactic shape recognition is the work of You and Fu [35] on attributed shape grammars. In an attributed shape grammar, each curve segment (primitive or nonterminal) is described by four attributes: the vector spanning its endpoints, its length, its angular change, and a symmetry measure. Grammar rules specify the generation of curve segments from smaller curve segments connected by angle primitives and are of the context free form $N \rightarrow (XA)^*X$ where N is a nonterminal, X can be a primitive curve segment or a nonterminal, and A is an angle primitive. In this case, the description of the curve segment N can be obtained directly from the descriptions of the curve segments composing N and the angles between them. Two recognition algorithms have

been developed which accomplish parsing and primitive extraction simultaneously, using high-level knowledge to drive low-level processes.

Syntactic pattern recognition has the main advantage of having well-defined, often linear, parsing algorithms. The main disadvantage of this approach to shape recognition is that the shape is usually first encoded as a one-dimensional string of symbols which is the input to the parser. The You and Fu work is an exception in that the parser directs the low-level processes that extract and encode primitives. However, the parsing is still inherently a one-dimensional process due to the nature of context free grammars. While the attributed shape grammar and associated recognition algorithms seem to be more powerful than previous shape grammars, they typically handle only adjacency relationships on the boundary of the shape.

It is not clear to us that all the information in a two-dimensional shape can be preserved in a one-dimensional representation. In particular, the intrusion and protrusion relations defined in this paper cannot be expressed naturally by one-dimensional grammar rules since they capture nonlinear information about the shape.

The approaches that are most related to our work are those that attempt to decompose the whole shape into two-dimensional parts. This includes the work of Pavlidis [25], Feng and Pavlidis [10], Maruyama [22], and Eden [9]. In the Feng and Pavlidis work, a shape is decomposed into convex parts, T-shaped parts, and spirals and described by a labeled graph indicating connections between pairs of parts. Maruyama suggests a decomposition of shapes into angularly simple regions where each such region has at least one interior point that can "see" its entire boundary. Eden decomposes script characters into meaningful primitive strokes.

In [31] we presented an algorithm for the decomposition of a two-dimensional shape into simple parts. The input to this algorithm is an ordered sequence of (x, y) coordinates representing the vertices of a polygonal approximation to the boundary of a shape. From these points the interior line segment relation LI is computed. LI is a binary relation consisting of all pairs of vertices such that the straight line segment joining them lies wholly within the shape. A graph-theoretic clustering algorithm on the relation LI yields a set of clusters, each consisting of a subset of the original set of vertices. These clusters are the *interior clusters* or *simple parts* of the shape.

The decomposition algorithm has three important properties.

- 1) The algorithm is independent of the starting point on the boundary of the shape.
- 2) The relation LI is invariant under linear transformations and perspective transformations; thus, a transformation of a shape yields the same decomposition as the original shape.
- 3) The clusters produced by the graph-theoretic clustering algorithm can be controlled by numerical parameters; they are not necessarily convex or even disjoint. Thus, the definition of simple part remains flexible.

Fig. 1(a) shows the decomposition of a hand-printed letter E by this method.

In [31] we defined the exterior line segment relation LE,

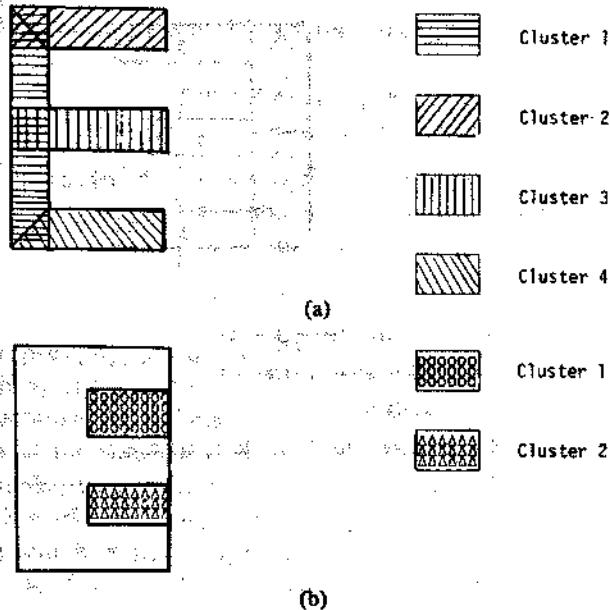


Fig. 1. Illustrates the decomposition of a shape into simple parts and intrusions. (a) Shows the simple parts or interior clusters of a hand-printed letter E. (b) Shows the intrusions or exterior clusters of the hand-printed letter E.

consisting of all pairs of vertices such that the straight line joining them lies exterior to the shape, and suggested that LE would be useful in obtaining a structural description of the shape. The clusters of the LE relation, obtained with the same graph-theoretic clustering algorithm, correspond to *intrusions* into the boundary of the shape. Fig. 1(b) shows these *exterior clusters* for the letter E of Fig. 1(a).

In Section III we will describe a structural model for a shape based on the simple parts, the intrusions, their properties, and their interrelationships.

III. THE MODEL

A structural description of a shape must include a set of shape primitives, their properties, and their interrelationships. In this section we will define a general, formal, structural model for the description of two-dimensional shapes. We will then define a specific structural description that we have been using in our shape matching experiments.

The General Model and Some Examples

We define a shape description D to be a 5-tuple $D = (N, T, G, P, R)$ where N is the *name* of the shape being described, T is the *type* of the shape, G is an attribute-value table containing the *global attributes* of the shape, P is a set of *shape primitives* obtained from a structural analysis of the shape, and R is a set $\{R_1, \dots, R_K\}$ of *relations* on P. For each k, there is a positive integer N_k and a possibly singleton label set L_k such that $R_k \subseteq P^{N_k} \times L_k$, $k=1, \dots, K$. Thus, each R_k is a labeled N_k -ary relation. Haralick and Kartus [14] call R_k an "arrangement."

The name N and the type T are identifying information. The name might identify a region in an image being analyzed or it might be a character string that the experimenter wants to associate with a particular input file. The type identifies a class that the shape belongs to. For instance, if zip codes

are being analyzed, the type might be "numeral." The purpose of the type is to limit the number of stored prototypes that are considered for a match. (See Section V.) Of course in the worst case, where the origin of the shape is completely unknown, no type information can be input.

A structural description of a shape includes a table of attribute-value pairs pertaining to the shape as a whole, a set of primitives that the shape can be broken down into, and a set of relations showing the interrelationships among the primitives. While the primitives P and the relations R define the structure of complex shapes, we feel that it is also important to record any global attributes that are available. If the shape is associated with a region of an image, then it might be convenient to consider intensity or texture information as global attributes of the shape. The global attributes may also include statistical descriptors such as circularity, elongation, convexity, crenation (notched boundary), and so on [17]. For simpler shapes, the global attributes may be more important than the primitives and relations, which may be missing altogether. The primitives and relations will differ with each particular shape model. We will illustrate with a few examples borrowed from the literature.

Shaw [32] defined a picture description language (PDL) in which pictures could be defined hierarchically in terms of primitives. Each primitive in PDL is a two-dimensional entity having two connection points, a *head* and a *tail*. The operators +, -, X, and * indicate the head-to-tail, head-to-head, tail-to-tail, and head-to-head with tail-to-tail connection of two primitives. For example, the expression "A + B" indicates that the head of A is to be connected to the tail of B. The resultant entity has its tail at the tail of A and its head at the head of B and can be connected to other primitives or higher entities. PDL sentences are expressions consisting of primitives, operators, and parentheses indicating the order of applying the operators. For example, "A * (B + C)" indicates that the head of B is connected to the tail of C to form a new entity. Then the new entity is connected to A, head-to-head and tail-to-tail.

We will give an example of a simple shape description using Shaw's operators, but not using the PDL expression concept. The primitives of our example shape are rectangles one unit long and one-half unit wide of various orientations. Fig. 2 illustrates a set of such primitives. The primitives can be connected head-to-tail, head-to-head, and tail-to-tail. (Since they are not curved, the head-to-head with tail-to-tail connection is not possible.) We will assume that for each pair of primitives, the method for joining them in each of the possible connections has been specified. We can describe certain simple shapes such as block letters in terms of these primitives and their pairwise connections.

A description for a block letter would be a 5-tuple $D_1 = (N, T, G, P, R)$ where N is the name of the letter, T is the type "block letter," G contains global attributes of the letter such as the number of primitives, P is the subset of the primitives of Fig. 2 that the letter is composed of, and R contains a single labeled binary connection relation $C \subseteq \{(p_1, p_2, O) | p_1, p_2 \in S, O \in \{+, -, X\}\}$. Fig. 3 illustrates this type of description for the block letter E.

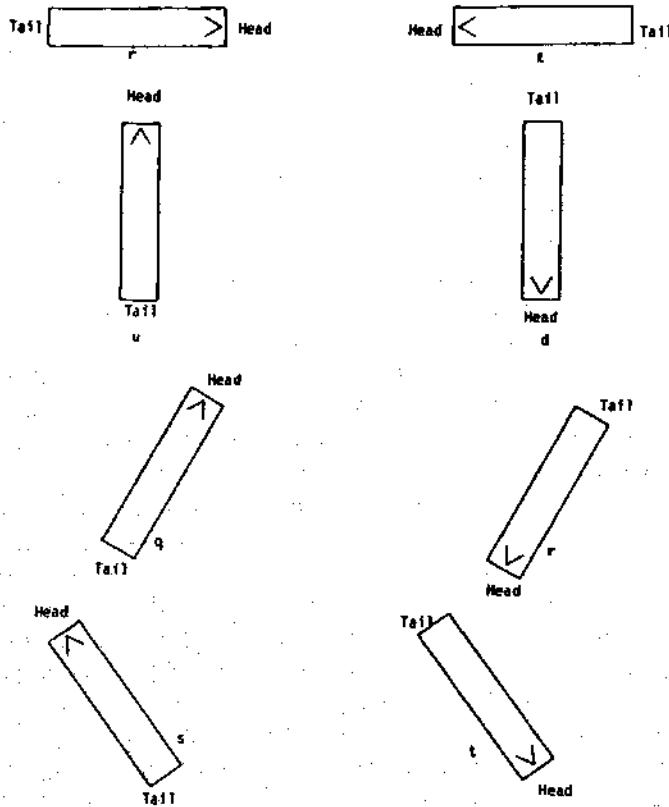


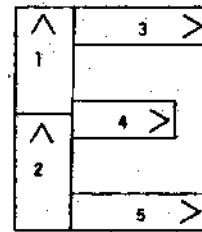
Fig. 2. Illustrates a set of rectangular shape primitives.

Another approach to shape description is in terms of an ordered sequence of line segments comprising the boundary of the shape. The relative lengths of each line segment and the relative angles between successive segments complete the model. A description for a block letter in this system would be a 5-tuple $D_2 = (N, T, G, P, R)$ where N, T, and G are as above, P is a set of line segments whose attributes include length, and R contains the labeled binary relation $J = \{(\ell_1, \ell_2, a) | \ell_1, \ell_2 \in P, \ell_1 \text{ connects to } \ell_2 \text{ and } a \text{ is the angle between } \ell_1 \text{ and } \ell_2\}$. Fig. 4 illustrates this type of description for the block letter E.

Both description D1 and description D2 are size invariant since all lengths specified are also relative. In description D1, the primitives are defined according to their orientation. Thus, the structural descriptions are not rotation invariant. This sort of description can work very well in applications like character recognition where the text is assumed to be right-side up, but descriptions using orientations cannot handle a general shape recognition problem. Description D2 is rotation invariant since the only orientation information is the set of relative angles between adjacent pairs of line segments. A description like D2 which concentrates on the boundary of the shape is useful in shape matching problems where the boundary is of primary importance. Map data is an example of this class of shapes.

A Structural Model Based on Simple Parts and Intrusions

One aspect of shape is its size and orientation invariance. This is the aspect that we attempt to capture with the structural model presented here. Let $S = \{s_1, \dots, s_n\}$ be the set of simple parts of a shape, and let $I = \{i_1, \dots, i_m\}$ be the set

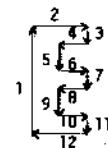


```

D1 = ('E', block letter, G, P, (C))
G = ((number-of-primitives, 5))
P = {1, 2, 3, 4, 5}
C = ((2, 1, +)(1, 3, +)(1, 4, x)(2, 4, +)(2, 5, x))
1 = u
2 = v
3 = r
4 = r
5 = r

```

Fig. 3. Illustrates a shape description for a block letter E. The primitives are directed, oriented rectangles, and the relation C consists of triples of the form (primitive 1, primitive 2, operator) where the operator determines the way in which the two primitives connect. The idea of directed primitives and the operators + (head-to-tail connection) and x (tail-to-tail) connection are from the work of Shaw [32].



```

D2 = ('E', block letter, G, P, (J))
G = ((number-of-segments, 12))
P = {(1, 2, 3, 4, ..., 12)}
J = {((1, 2, -90), (2, 3, -90), (3, 4, -90),
      (4, 5, 90), (5, 6, 90), (6, 7, -90),
      (7, 8, 90), (8, 9, 90), (9, 10, 90),
      (10, 11, -90), (11, 12, -90), (12, 1, -90))
      }

```

1 = 4 units	4 = 1 unit	7 = 1 unit	10 = 1 unit
2 = 2 units	5 = 1/2 unit	8 = 1 unit	11 = 1 unit
3 = 1 unit	6 = 1 unit	9 = 1/4 unit	12 = 2 units

Fig. 4. Illustrates a second shape description for a block letter E. The primitives, which are line segments, have one attribute—their relative length. The relation J consists of triples of the form (segment1, segment2, angle) where segment1 and segment2 join and angle is the angle between them. For the purpose of calculating the angles, the line segments are considered to be directed clockwise.

of intrusions. S and I are assumed to be obtained from the shape by a decomposition procedure such as the one described in [31]. We will treat each simple part and each intrusion as a two-dimensional region having area and a closed boundary. Let $P = S \cup I$. P consists of the simple parts and intrusions. Let $\delta: P \times P \rightarrow [0, \infty)$ be a distance function which gives a relative measure of the distance between two simple parts, two intrusions, or a simple part and an intrusion with the property that if p_1 and p_2 touch or overlap, then $\delta(p_1, p_2) = 0$. Let Δ be a nonnegative real number. Define

the Boolean function $p: P^3 \rightarrow \{\text{true, false}\}$ by

$$p(p_1, p_2, p_3) = \begin{cases} \text{true if } \delta(p_1, p_2) \leq \Delta, \\ \quad \delta(p_2, p_3) \leq \Delta, \text{ and} \\ \text{it is possible to draw a straight line from} \\ \text{a boundary point of } p_1 \text{ through } p_2 \text{ to a} \\ \text{boundary point of } p_3 \\ \text{false otherwise.} \end{cases}$$

Thus, $p(p_1, p_2, p_3)$ is true if both p_1 and p_3 touch or almost touch p_2 and a part of p_2 lies between p_1 and p_3 . A shape is characterized by a 5-tuple $M = (N, T, G, P, R)$ where N is the name of the shape, T is the type of the shape, $G = \{\text{(number-of-simple-parts, \#S), (number-of-intrusions, \#I)}\}$, $P = S \cup I$, and $R = \{R_p, R_i\}$. $R_p = \{(i_1, s, i_2) \in I \times S \times I \mid p(i_1, s, i_2) = \text{true}\}$ and $R_i = \{(s_1, i, s_2) \in S \times I \times S \mid p(s_1, i, s_2) = \text{true and } \delta(s_1, s_2) \leq \Delta\}$. Thus, R_p consists of triples of the form (intrusion 1, simple part, intrusion 2) where the simple part protrudes between the two intrusions, and R_i consists of triples of the form (simple part 1, intrusion, simple part 2) where the two simple parts touch or nearly touch and form part of the boundary of the intrusion. Note that although the relations R_i and R_p are currently defined using the predicate p which returns true or false, alternate definitions could be formulated where p would return a numeric value. So far no provision has been made for describing each simple part and each intrusion of a shape. The decomposition algorithm was designed to produce near-convex primitives, and it does not seem useful to decompose these primitives any further. However, a list of the properties of each primitive would add to the completeness of the description of the entire shape. Some of the properties of a primitive that might be useful are listed below:

- 1) circularity
- 2) elongation
- 3) convexity
- 4) crenation
- 5) relative length
- 6) relative width
- 7) relative area

Example

Fig. 1 illustrates the decomposition of a block letter E into four simple parts and two intrusions. We will name the simple parts 1, 2, 3, and 4 corresponding to clusters 1, 2, 3, and 4 in Fig. 1(a) and the intrusions 11 and 12 corresponding to clusters 1 and 2, respectively, in Fig. 1(b). Then the structural description of the letter E for any value of Δ is

$$(E, \text{letter}, G, P, \{R_p, R_i\})$$

$$G = \{\text{(number-of-simple-parts, 4), (number-of-intrusions, 2)}\}$$

$$P = \{1, 2, 3, 4, 11, 12\}$$

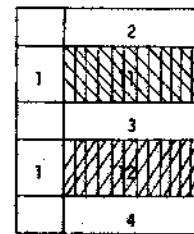
$$R_p = \{(11, 3, 12)\}$$

$$R_i = \{(1, 11, 2),$$

$$(1, 11, 3),$$

$$(1, 12, 3),$$

$$(1, 12, 4)\}$$



$$R_p = \{(11, 3, 12)\}$$

$$R_i = \{(1, 11, 2), \\ (1, 11, 3), \\ (1, 12, 3), \\ (1, 12, 4)\}$$

Fig. 5. Illustrates the protrusion relation R_p and the intrusion relation R_i of a block letter "E."

Fig. 5 illustrates this symbolically and Fig. 6 compares the decompositions of five block letters E, G, I, C, and B into simple parts and intrusions and the R_p and R_i relations for each letter. (Note that for brevity, we list only triples of the form (p_1, p_2, p_3) , $p_1 < p_3$ and omit their symmetric counterparts.)

IV. SHAPE MATCHING

For our purposes, two shapes *match* if there is a function that maps the primitives of the first shape to the primitives of the second in such a way that primitives map to like primitives and the interrelationships among the primitives are preserved. Such a structure preserving function is called a *relational homomorphism*. Haralick and Kartus [14] define a *relational homomorphism* as follows.

Let A be a finite set of objects, let L be a finite set of labels, let $R \subseteq A^N \times L$ be a labeled N -ary relation, and let $h: A \rightarrow B$ be a mapping from A to a second set B . The *composition* of the relation R with the function h is the labeled N -ary relation $R \circ h$ defined by

$$R \circ h = \{(b_1, \dots, b_N, \ell) \in B^N \times L \mid \text{there exists} \\ (a_1, \dots, a_N, \ell) \in R \text{ with } h(a_i) = b_i, i = 1, \dots, N\}.$$

Suppose $R \subseteq A^N \times L$ and $R' \subseteq B^N \times L$. A *relational homomorphism* from R to R' is a function $h: A \rightarrow B$ such that $R \circ h \subseteq R'$.

Thus, the composition of an N -ary relation with a function produces a second N -ary relation. The relational homomorphism is a structure preserving function that maps a labeled N -ary relation onto a subset of a second labeled N -ary relation. If shapes are represented by labeled relations, then two shapes match if there is a relational homomorphism from one relation to the other.

The structural description of a shape defined in Section III includes two ternary relations: R_p , the protrusion relation, and R_i , the intrusion relation. In order to simplify our description of the shape matching process, we will combine R_i and R_p to form a labeled ternary relation $R(R_i, R_p)$ defined by

$$R(R_i, R_p) = \{(p_1, p_2, p_3, \ell) \in P^3 \mid \text{either } (p_1, p_2, p_3) \in R_i \text{ and } \ell = i \text{ or } (p_1, p_2, p_3) \in R_p \text{ and } \ell = p\}.$$

Each element of $R(R_i, R_p)$ is a triple of either R_i or R_p with a label indicating whether it is from R_i or R_p .

The elements of $R(R_i, R_p)$ contain information about the

E	G	I	C	B
R_p	11,3,12	11,5,12	11,1,12	empty
R_i	1,11,2 1,11,3 1,12,3 1,12,4 2,12,3	1,12,2 2,11,3 3,11,4 4,11,5 2,12,3	1,11,2 1,12,2 1,11,3 1,12,3 1,12,3	1,11,2 2,11,3 3,11,4 4,11,5

Fig. 6. Illustrates decompositions of five different block letters and the R_p and R_i relations for each letter.

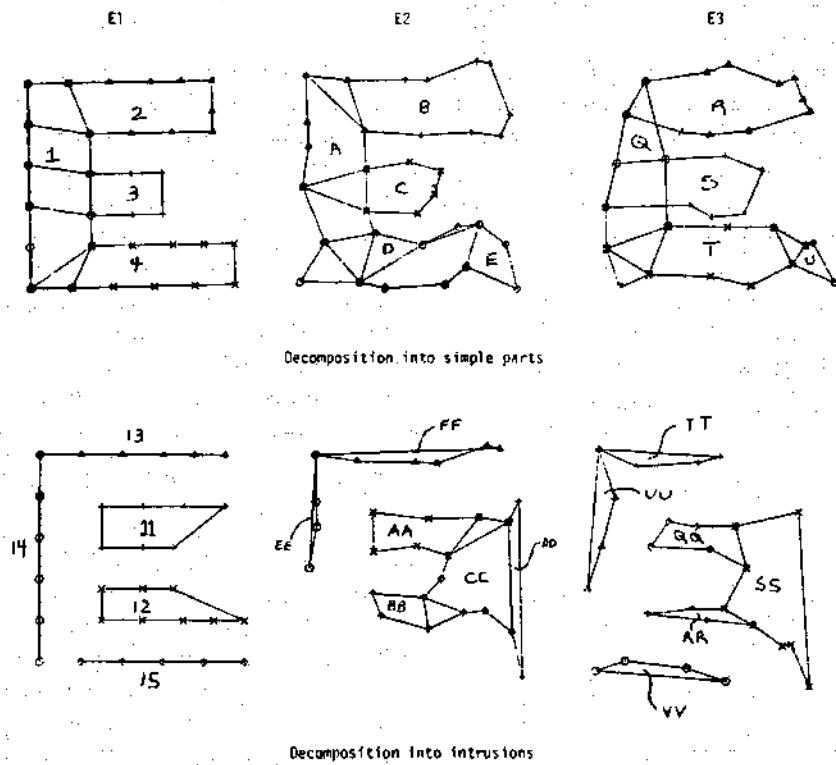


Fig. 7. Illustrates the decomposition of three shapes into simple parts and intrusions. Shape E1 was hand-printed and is considered a prototype letter "E." Shapes E2 and E3 were produced by adding noise to shape E1.

structure of the shape and indicate which primitives are simple parts and which are intrusions. One form of shape matching is to compare candidate shapes to a prototype shape and determine whether the candidate is similar enough to the prototype to be considered a match. Let $S_1 = (N_1, T_1, G_1, S_1 \cup I_1, \{R_{p1}, R_{i1}\})$ be a *prototype shape* and $S_2 = (N_2, T_2, G_2, S_2 \cup I_2, \{R_{p2}, R_{i2}\})$ be a *candidate shape*. Let $R_1 = R(R_{i1}, R_{p1})$ and $R_2 = R(R_{i2}, R_{p2})$. The candidate shape S_2 matches the prototype shape S_1 if there is a relational homomorphism h from R_1 to R_2 .

For example, consider the decomposition of shapes E_1, E_2 ,

and E_3 in Fig. 7. Suppose E_1 is a prototype shape for the letter 'E' and E_2 and E_3 are candidate shapes. The relations R_1, R_2 , and R_3 corresponding to E_1, E_2 , and E_3 are given below. Very thin exterior clusters (13, 14, 15, EE, DD, FF, TT, UU, and VV in Fig. 7) which correspond to almost straight parts of the boundary rather than true intrusions, have been eliminated from the descriptions. These "false intrusions" will be eliminated by the program that calculates R_i and R_p . They are currently produced by the algorithm used to find the LE relation and have not been eliminated earlier because we think they may prove useful in future shape analyses.

R_1	R_2	R_3
(1, 11, 2, i)	(A, AA, B, i)	(Q, QQ, R, i)
(1, 11, 3, i)	(A, AA, C, i)	(Q, QQ, S, i)
(1, 12, 3, i)	(A, BB, C, i)	(Q, RR, S, i)
(1, 12, 4, i)	(A, BB, D, i)	(Q, RR, T, i)
(11, 3, 12, p)	(A, BB, E, i)	(T, SS, U, i)
	(D, BB, E, i)	(QQ, S, RR, p)
	(D, CC, E, i)	(RR, U, SS, p)
	(AA, C, BB, p)	(RR, T, SS, p)
	(AA, C, CC, p)	

Now for E_2 to match E_1 there must be a homomorphism $h_{12}: (S_1 \cup I_1) \rightarrow (S_2 \cup I_2)$ from R_1 to R_2 . Four such homomorphisms are given below:

$p \in (S_1 \cup I_1)$	$h_{12}(p)$	$h_{12}(p)$	$h_{12}(p)$	$h_{12}(p)$
1	A	A	A	A
2	B	B	D	E
3	C	C	C	C
4	D	E	B	B
11	AA	AA	BB	BB
12	BB	BB	AA	AA

For each, i , $1 \leq i \leq 4$, it is easy to show that $R_1 \circ h_{12}^{(i)} \subseteq R_2$. For example, $R_1 \circ h_{12}^{(1)} = \{(A, AA, B, i), (A, AA, C, i), (A, BB, C, i), (A, BB, D, i), (AA, C, BB, p)\} \subseteq R_2$. Similarly, there are homomorphisms $h_{13}: (S_1 \cup I_1) \rightarrow (S_3 \cup I_3)$ from R_1 to R_3 . Thus, both candidate E_2 and candidate E_3 can be said to match prototype E_1 . Notice that some of the homomorphisms from a model to a candidate shape really represent the mappings from the model to the mirror image of the candidate. In the above example, $h_{12}^{(3)}$ and $h_{12}^{(4)}$ are the mirror image mappings.

A shape matching procedure must find a homomorphism from a prototype shape to a candidate shape. Several important considerations pertain to the design of such a procedure.

1) Some of the relationships in the prototype shape may be more important than others. Thus, it is desirable to assign weights to tuples in the description of the prototype.

2) It is possible for two similar shapes to decompose into different numbers of primitives. Thus, the mapping should be able to assign the same primitives in the candidate to several primitives in the prototype if they are physically close enough to each other. (If we choose to look for a binary relation instead of a mapping, then several primitives in the candidate could be associated with one primitive in the prototype.)

3) A homomorphism corresponds to a match, but does not necessarily indicate a good match. We need a measure of the goodness of a match. In the remainder of this section we will discuss each of these considerations in more detail.

Mappings on Weighted Relations

Suppose $R_1 \subseteq P_1^3 \times \{i, p\}$ is a prototype relation and $R_2 \subseteq P_2^3 \times \{i, p\}$ is a candidate relation. Let W be the set

$[0, 1]$ of real numbers between 0 and 1. A weighting function for R_1 is a mapping $w: R_1 \rightarrow W$ that assigns a weight to each triple of R_1 such that $\sum_{r \in R_1} w(r) = 1$. Let h be a mapping from P_1 to P_2 and w a weighting function for R_1 . A triple r of R_1 is satisfied by h with respect to R_2 if $h(r) \in R_2$. If h is a homomorphism from R_1 to R_2 , then

$$\sum_{r \in R_1} w(r) = 0 \text{ since } R_1 \circ h \subseteq R_2.$$

[r not satisfied by h]

In order to allow imperfect matches, we define an ϵ -homomorphism to be a mapping $h: P_1 \rightarrow P_2$ such that

$$\sum_{r \in R_1} w(r) \leq \epsilon.$$

[r not satisfied by h]

Thus, in an ϵ -homomorphism the sum of the weights of those triples that are not satisfied is not greater than ϵ . A homomorphism, as defined earlier, is a 0-homomorphism.

Using these ideas, a human or a computer can assign weights to the triples of the prototype relation. One method of obtaining the weights would be to let a program induce them from imperfect matches ranked as to goodness of match. Then the shape matching problem is to find ϵ -homomorphisms from the prototype to the candidate.

Mappings That Can Assign Several Candidate Primitives to One Prototype Primitive

We would like to allow mappings that are not one-one, but with the restriction that only primitives that are near enough to each other can map to a single primitive. This leads to the mergeability relation $M \subseteq (S_1 \times S_1) \cup (I_1 \times I_1)$. If $(p_1, p_2) \in M$ then p_1 and p_2 may map to the same candidate primitive. One possible definition for M is

$$M = \{(p_1, p_2) | \delta(p_1, p_2) \leq \Delta\},$$

A mapping $h: P_1 \rightarrow P_2$ is consistent with M if $h(p) = h(q)$ implies $(p, q) \in M$. Thus, we can expand the shape matching problem to find all ϵ -homomorphisms that are consistent with M .

Measuring the Goodness of a Match

The quantity

$$\sum_{r \in R_1} w(r)$$

[r satisfied by h]

measures the relational goodness of a mapping h . Two other measures are of interest:

1) a measure of the similarity between each primitive p of the prototype to the corresponding primitive $h(p)$ of the candidate;

2) a measure of the size of the matched subset of the candidate primitives.

The measure 1) would take into account properties of a primitive such as those mentioned in Section III. The strictness or leniency of the enforcement of this measure in the

matching criteria will determine how close to identical two shapes must be in order to match. The measure 2) might simply be the ratio of the area of the matched primitives (or just the simple parts) of the candidate to the total area of the primitives (or simple parts) of the candidate.

The Shape Matching Procedure

The problem of finding homomorphisms has been with us for some time. (See Barrow, Ambler, and Burstall [3], Ullman [33], and Corneil and Gotlieb [5].) Finding homomorphisms can be accomplished by means of a tree search incorporating

a "look-ahead" or "relaxation" operator (see Haralick and Shapiro [15]). Finding ϵ -homomorphisms can be accomplished by a similar procedure. We now define a simple procedure for finding ϵ -homomorphisms that are consistent with a mergeability relation. The following high-level algorithm describes the procedure for finding all ϵ -homomorphisms that are consistent with mergeability relation M from a weighted N-ary relation R_1 on a set U of primitives of a prototype to an N-ary relation R_2 on a set L of primitives of a candidate. The primitives of U are referred to as *units* and the primitives of L as *labels*.

```

procedure MAIN( );
comment LABELS is a table
      LABELS(u) contains a list of those elements
      of L that u may be mapped to by the homomorphism
      to be found;
read (e);
read (M);
read (U);
read (L);
read (R1);
read (R2);
R := R1 × R2;
for each u ∈ U do
    read (LABELS(u));
call TREESEARCH (R1, R, LABELS);
stop
end MAIN;

procedure TREESEARCH(R1, R, LABELS);
comment Recursively perform a tree search to find mappings from the set
      U of primitives of the prototype to the set L of primitives of
      the candidate. The mappings found will be  $\epsilon$ -homomorphisms from
      R1 to R2 where R = R1 × R2;
THISUNIT := an element of U that has the least number of labels;
NEXTLABEL: if (there are no more labels in LABELS(THISUNIT)
then return;
THISLABEL := next label in LABELS(THISUNIT);
R' := RESTRICTION(R, THISUNIT, THISLABEL);
LABELS' := EPSILON_PROJECTION(R1, R', LABELS);
if (LABELS' = ∅) then goto NEXTLABEL
comment the following 4 statements are included if the treesearch is to
use a relaxation procedure on R;
RELAXR: R' := EPSILON_RELAX(R1, R', LABELS');
LABELS' := EPSILON_PROJECTION(R1, R', LABELS');
if (LABELS' = ∅) then goto NEXTLABEL;
if (EPSILON_RELAX removed any N-tuples from R')
then goto RELAXR;
comment If relaxation is not being used, the partial homomorphism defined
by those elements of U that have only one label left in LABELS'
must be checked to see that it meets the requirements of an  $\epsilon$ -homomorphism;
if ( $\neg$  EPSILON_CONSISTENT(LABELS')) then goto NEXTLABEL;
comment Now if all the elements of U have only a single label left in LABELS',
we have a homomorphism, otherwise continue the tree search;
if (LABELS' is single-valued)
then print ('HOMOMORPHISM', LABELS')

```

```

else call TREESEARCH(R1,R',LABELS);
comment Continue looking for more mappings;
goto NEXTLABEL;
end TREESEARCH

procedure EPSILON_PROJECTION(R1,R,LABELS);
comment Produce a new table containing for each unit u those labels that an
epsilon-homomorphism may still map u to, given the current state of R;
for each unit UNIT that has only one label in LABELS do
begin
    EPSILON_PROJECTION(UNIT) := LABELS(UNIT);
    PARTIAL(UNIT) := LABELS(UNIT)
end;
for each unit UNIT that has more than one label in LABELS do
for each label LAB in LABELS(UNIT) do
begin
    PARTIAL(UNIT) := LAB;
    if EPSILON_CONSISTENT(PARTIAL,R1,R)
        then add LAB to EPSILON_PROJECTION(UNIT)
    end;
return;
end EPSILON_PROJECTION;

procedure EPSILON_CONSISTENT(LABELS,R1,R)
comment determine if the partial function defined by the single-valued
entries in LABELS is an epsilon-homomorphism;
ERROR_SUM := 0;
for each N-tuple  $(u_1, \dots, u_N) \in R1$  | such that each of  $u_1, \dots, u_N$  has just
one label left in LABELS do
begin
if  $((u_1, \text{LABELS}(u_1)), \dots, (u_N, \text{LABELS}(u_N)))$  is
not in R
then ERROR_SUM := ERRORSUM + weight  $(u_1, \dots, u_N)$ ;
if ERROR_SUM > epsilon then goto FAIL
end;
EPSILON_CONSISTENT = true;
return;
FAIL: EPSILON_CONSISTENT = false;
return;
end EPSILON_CONSISTENT;

```

The algorithm described by the above procedures was used for shape matching. The relations R_p and R_i were used in place of the general relation R . R_p and R_i were stored, restricted, projected, and relaxed on separately because the program ran faster using the two smaller separate relations and combining their results than by merging them into one larger relation. The program was tested with and without a relaxation operator. Because the shape relations are relatively small, the overhead involved in setting up and carrying out the relaxation was greater than the amount of time required to perform the tree-search with the restriction and projection, but without the relaxation. The weighted discrete relaxation procedure, which does improve time significantly on larger relations, will be described in a future paper.

V. EXPERIMENTAL RESULTS

The shape matching procedure has been tested on character data. The data consist of a set of hand-printed characters

which are considered to be prototype shapes and for each prototype shape, a set of imperfect versions of the shape. The imperfect versions were obtained by starting with a list of the boundary points of the prototype shape and calling on a uniform random number generator to produce horizontal and vertical displacements which were then added to the coordinates of the boundary points. By using different seeds for the random number generator, each prototype shape was used to generate several imperfect shapes with the same number of points, but with distorted boundaries. The amount of distortion was controlled by the range of displacements. Fig. 8 shows the prototype shapes and a set of corresponding imperfect versions.

The characters were first decomposed into simple parts and intrusions using the graph-theoretic clustering procedure described in [31]. Then the R_p and R_i relations were hand-coded for each character. In the hand-coding, we followed the definitions of R_p and R_i given in Section III as closely as

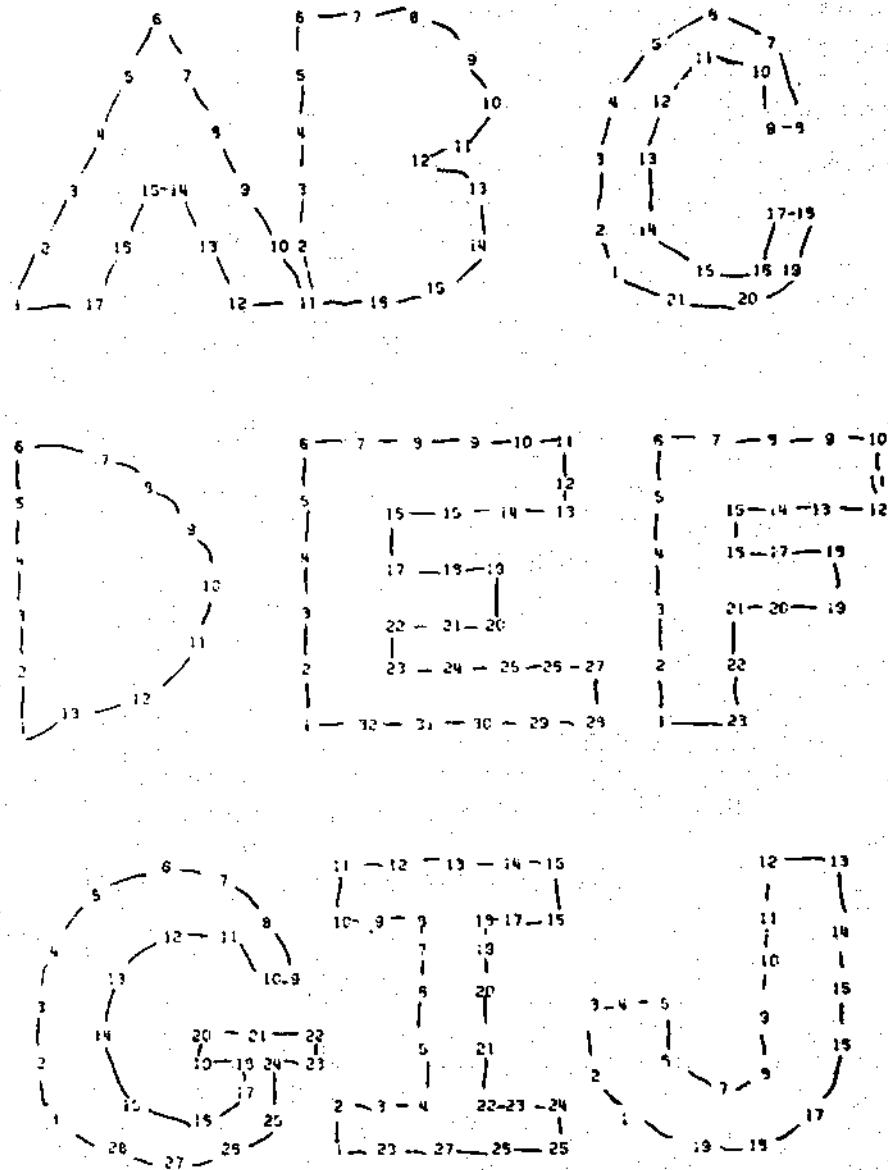


Fig. 8. Illustrates the prototype shapes and a set of corresponding imperfect versions.

possible using a distance function δ defined by $\delta(p_1, p_2) = 0$ if p_1 and p_2 have a point in common or if a point of p_1 is adjacent (along the boundary of the shape) to a point of p_2 and $\delta(p_1, p_2) = \infty$ otherwise. (The software to automatically produce the relations is under current development.) Several extra test shapes were constructed totally by hand in order to have some shapes which were not just imperfect versions of the prototype shapes, but were "near-misses" as used in Winston's concept-learning experiments [34]; that is, some important property of these shapes was missing. These shapes were used to further test the inexact matching.

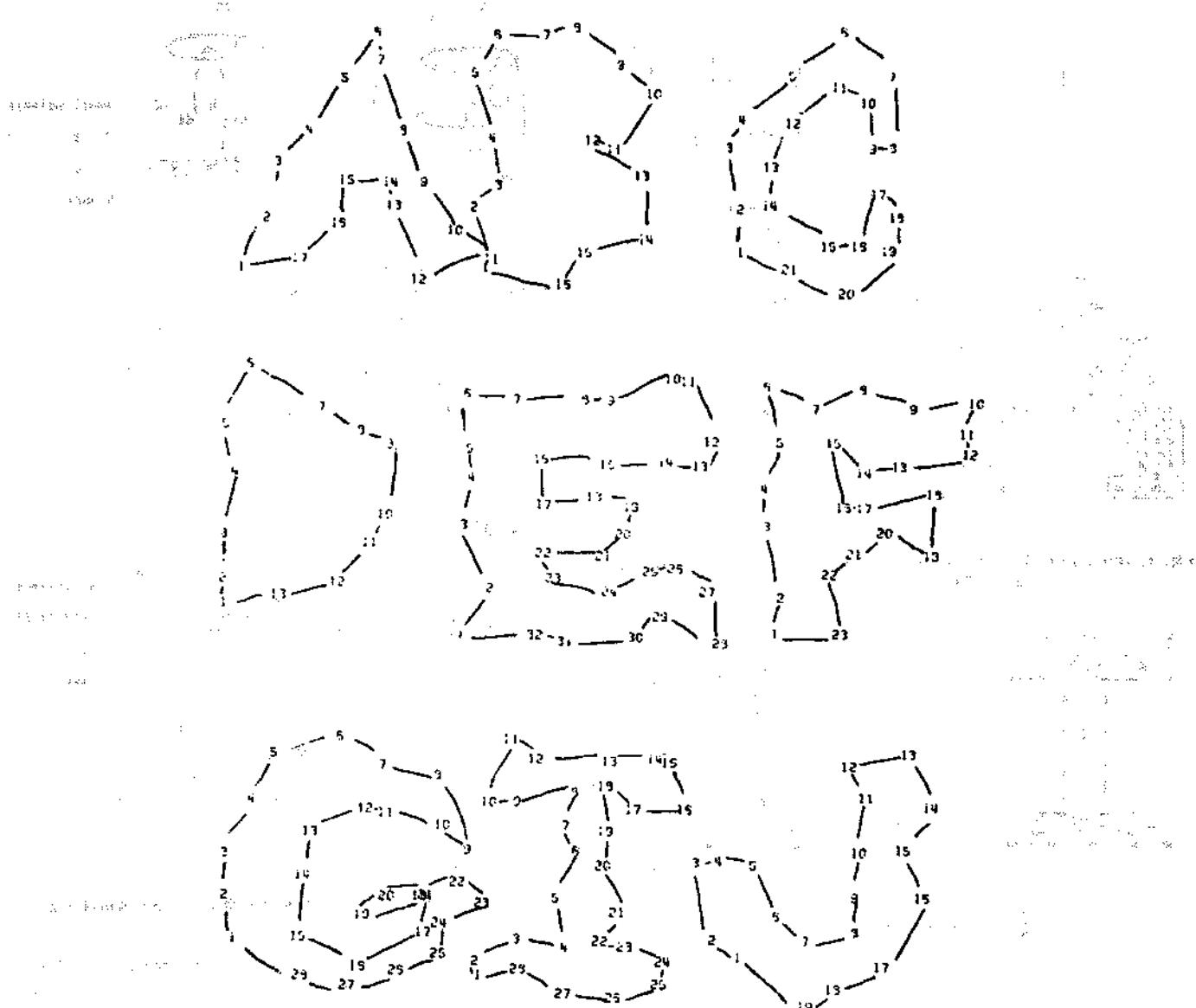
In each test, the matching program was given the R_p and R_i relations, a weighting function, and a mergeability relation for the prototype shape; the R_p and R_i relations for the candidate shape, and the inexact matching threshold ϵ . We will first describe some results where the weighting function assigned equal weights to each triple in the prototype relations, the mergeability relation M was empty or fixed, and ϵ was set to 0 for exact matching. We will then describe the effects of

varying ϵ , the mergeability relation, and the weights. We would like to emphasize at this point that our goals in running these experiments were not to test a new character recognition algorithm, but instead to learn about the concept of shape. Thus, for each test run, our interest was not in whether two shapes matched, but how much they matched and how much work the program had to do to decide how much they matched.

Example of Tests with Equal Weights,

Fixed Mergeability, and $\epsilon = 0$

We will illustrate this kind of test with a few characteristic examples. In one such test letter E1 of Fig. 7 was used as the prototype shape and letter E2 as the candidate shape. The program found the four legal 0-homomorphisms from E1 to E2 with a tree search in which 39 nodes were processed. Fig. 9 shows the tree search for this run. It is of interest to note that when the program used relaxation at each node of the tree search, only 16 nodes were processed, but the program



Imperfect Versions of Prototype Shapes

Fig. 8. (Continued.)

performed so many more operations due to the overhead of the relaxation that it executed four times as long.

As examples of nonmatches, letter E1 was matched against letter I3, and letter I1, the prototype I, was run against letter E2. The decompositions of letter I1 and letter I3 are shown in Fig. 10. In both cases the program reported no homomorphisms found while searching only six nodes of the tree. The tree searches are illustrated in Fig. 11.

Examples of Tests Where ϵ Was Varied

A number of tests were performed where ϵ was varied in order to watch the behavior of the program as it went from exact matching to inexact matching and to see what kind of inexact matches occurred at higher values of ϵ . The E1-E2 experiment which processed 39 nodes for $\epsilon = 0$, processed 31 nodes for $\epsilon = 0.25$, finding the four 0-homomorphisms plus four 0.25-homomorphisms. The additional 0.25 homomorphisms are given as follows:

1	2	3	4	11	12
A	D	C	E	AA	BB
A	E	C	D	AA	BB
A	D	C	E	BB	AA
A	E	C	D	BB	AA

The E1-I3 experiment which processed 6 nodes for $\epsilon = 0$, processed 45 nodes for $\epsilon = 0.5$, again reporting no homomorphisms. The I1-E2 experiment, which processed 6 nodes and found no matches for $\epsilon = 0$, processed 69 nodes in the time it was allowed to run and had found 13 0.5-homomorphisms before it reached its time limit. One such 0.5-homomorphism was $\{(1, C), (2, A), (3, A), (4, B), (5, B), (11, AA), (12, BB)\}$. In this mapping, the triples $(1, 11, 2)$ and $(1, 12, 3)$ of R_1 were satisfied and the triples $(1, 11, 4)$ and $(1, 12, 5)$ of R_1 were not satisfied. Since half the triples of R_1

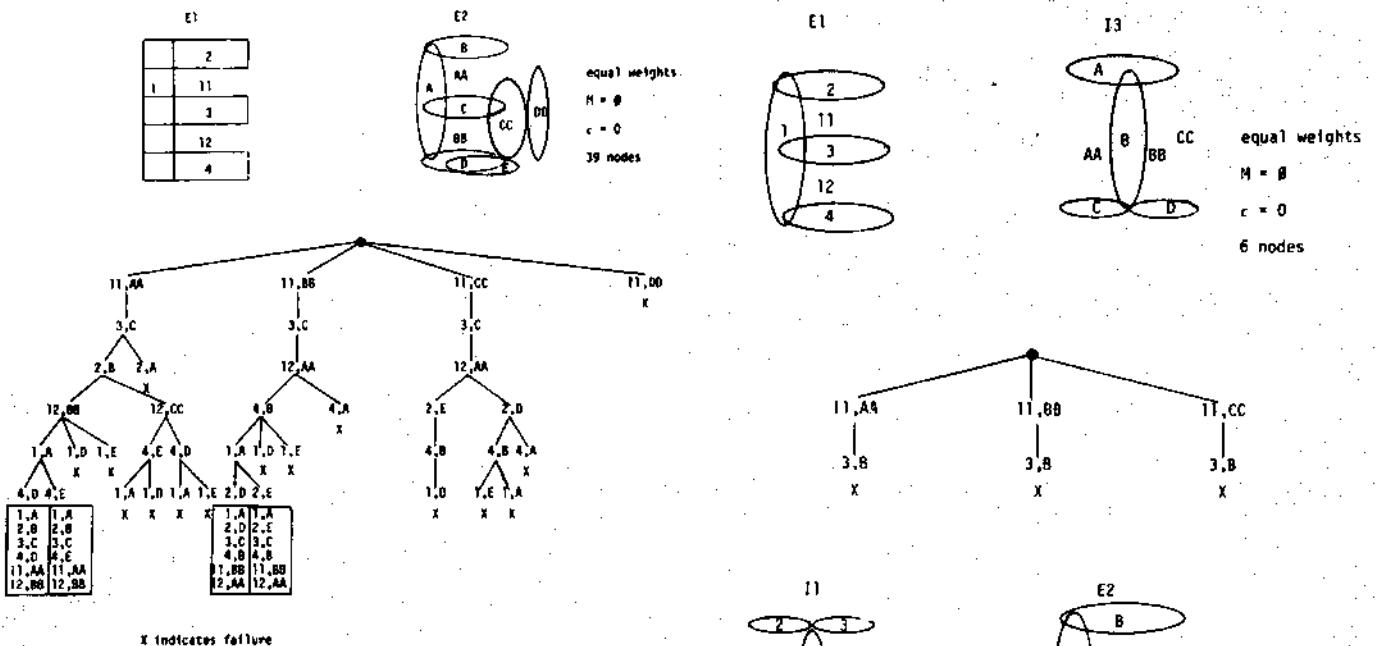
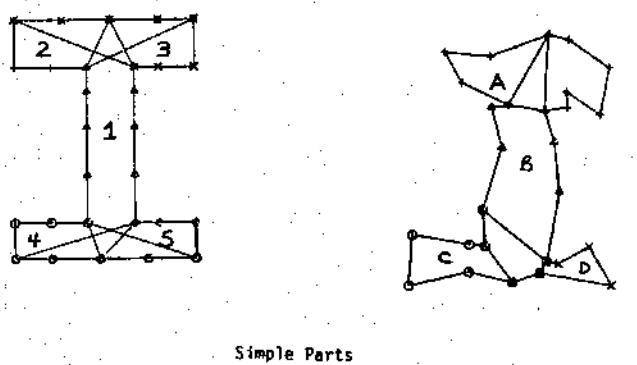


Fig. 9. Illustrates the tree search to find all homomorphisms from letter E1 of Fig. 7 to letter E2.



Simple Parts

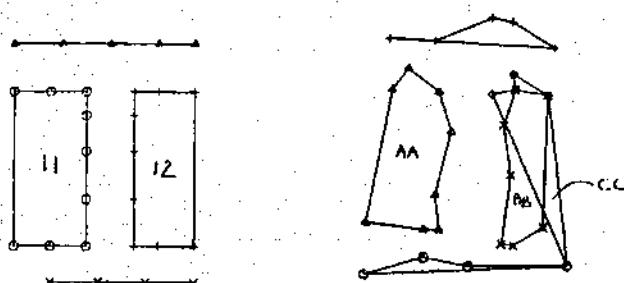


Fig. 10. Illustrates the simple parts and intrusions for the prototype shape I1 and the imperfect shape I3.

were satisfied, the mapping was a valid 0.5-homomorphism on R_i . R_p contained only the triple (11, 1, 12) which was satisfied. Fig. 12 illustrates this inexact match.

Fig. 13 shows the relations used in another inexact matching experiment. The prototype shape was letter E1 and the candidate a constructed relation NOT_E2. The program found no homomorphism for $\epsilon = 0$ processing 10 nodes, no homomorphisms for $\epsilon = 0.05$ processing 17 nodes, and no homomorphisms for $\epsilon = 0.25$ processing 31 nodes. The reason for no homomorphisms in this example is that no mapping could satisfy the R_p relation, which forces one of

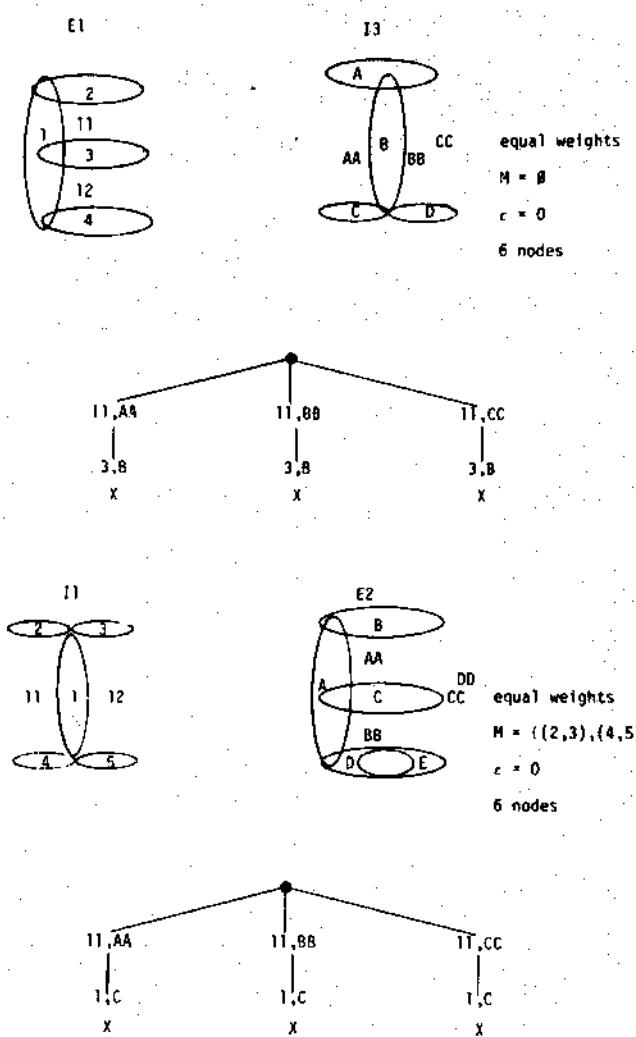
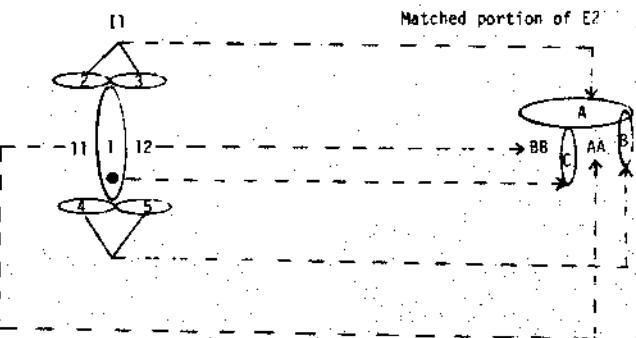


Fig. 11. Illustrates the tree search for two nonmatches.



R_i	
1,11,2	satisfied
1,12,3	
1,11,4	not satisfied
1,12,5	

R_p	
AA, BB	

Fig. 12. Illustrates an inexact match of I1 to a portion of E2 at $\epsilon = 0.5$.
the mappings $\{(11, AA), (3, C), (12, BB)\}$ or $\{(11, BB), (3, C), (12, AA)\}$, and simultaneously satisfy the R_i relation.
Fig. 14 shows the relations used and the tree searches in a

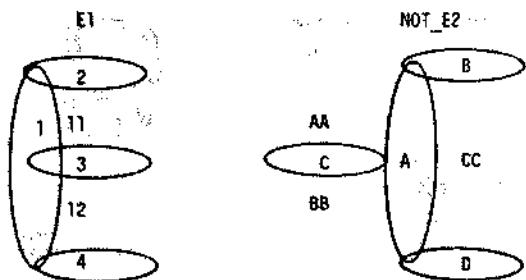


Fig. 13. Illustrates the relations used in an inexact matching experiment where no homomorphisms were found for $\epsilon = 0, 0.05$, or 0.25 .

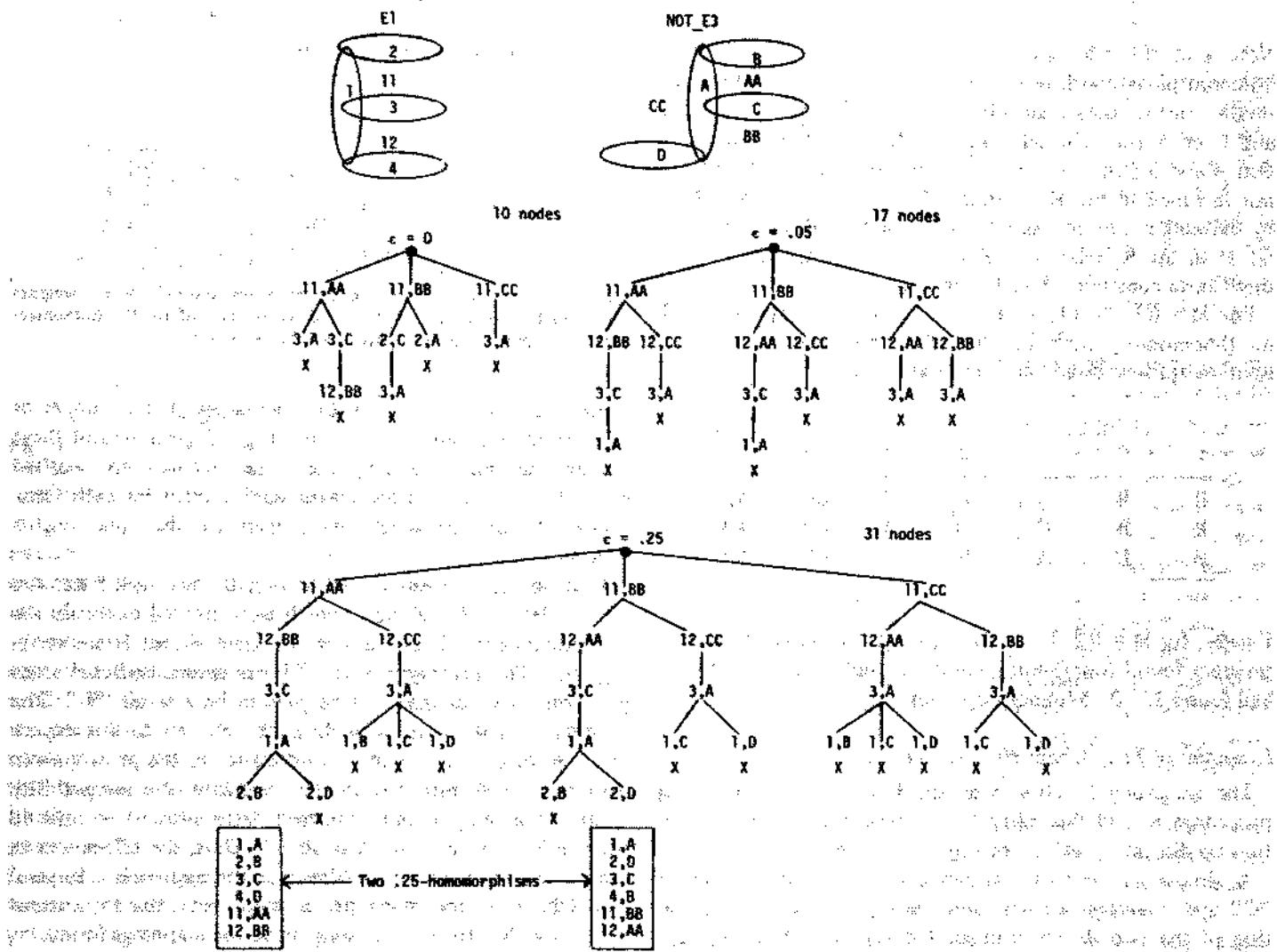


Fig. 14. Illustrates the relations and tree searches for an inexact matching experiment where two homomorphisms were found for $\epsilon = 0.25$.

similar experiment with prototypes E1 and candidate NOT_E3. This time two inexact matches were found when ϵ was set to 0.25. The program processed 10 nodes for $\epsilon = 0$, 17 nodes for $\epsilon = 0.05$, and 31 nodes for $\epsilon = 0.25$.

Examples of Tests Where M Was Varied

The mergeability relation M was the empty set for prototype E1 and was fixed at $\{(2, 3), (4, 5)\}$ for prototype II in the above examples. In this section we will describe some experiments using prototype II where M was varied.

Prototype II was matched against candidate I3 (both shown in Fig. 10) with $M = \emptyset, M = \{(2, 3)\}, M = \{(1, 2), (1, 3), (1, 4)\}$,

$(1, 5)\}$, and $M = \{(2, 3), (4, 5), (1, 2), (1, 3), (1, 4), (1, 5)\}$. Each value of M was tested with $\epsilon = 0$ and $\epsilon = 0.25$. For $M = \emptyset$, the program found no homomorphisms either with $\epsilon = 0$ or with $\epsilon = 0.25$. For $M = \{(2, 3)\}$, the program found two homomorphisms with $\epsilon = 0$ and eight homomorphisms with $\epsilon = 0.25$. The 0-homomorphisms are listed below:

	1	2	3	4	5	11	12
B	A	A	C	D	AA	BB	
B	A	A	D	C	BB	AA	

The 0.25-homomorphisms include the 0-homomorphisms and the six listed below:

1	2	3	4	5	11	12
* B	C	C	A	D	AA	BB
* B	D	D	C	A	AA	BB
B	A	A	C	D	AA	CC
* B	C	C	D	A	BB	AA
* B	D	D	A	C	BB	AA
B	A	A	D	C	CC	AA

Note that the mappings marked with "*" are valid 0.25-homomorphisms with respect to R_i and R_p , but are intuitively invalid because they map primitives 4 and 5 which touch to A and C or A and D which do not. This is caused by the fact that 4 and 5 join but border no intrusion and therefore were not included in the R_i relation. This problem can be solved by defining a null intrusion i_0 and including $(4, i_0, 5)$ and $(2, i_0, 3)$ in the R_i relation. Of course, for higher values of ϵ , these extra constraints could be ignored.

For $M = \{(1, 2), (1, 3), (1, 4), (1, 5)\}$, the program found no 0-homomorphisms and had discovered the three 0.25-homomorphisms listed below in its allotted time:

1	2	3	4	5	11	12
B	B	A	D	C	BB	AA
B	B	C	D	A	BB	AA
B	D	A	B	C	BB	AA

Finally, for $M = \{(2, 3), (4, 5), (1, 2), (1, 3), (1, 4), (1, 5)\}$ the program found four 0-homomorphisms and in its allotted time had found 222 0.25-homomorphisms.

Examples of Tests Where Weights Were Varied

The weighting function was added to the shape matching procedure to add flexibility to the experiments. In this section we discuss the effects of varying the weights.

A simple experiment was run using the prototype letter "C" and imperfect version shown in Fig. 8. The decomposition of the two shapes is illustrated symbolically in Fig. 15. The prototype C1 has five simple parts and one intrusion while the candidate C2 has only four simple parts and one intrusion. In the experiment the mergeability relation M was fixed at $\{(1, 2), (2, 3), (3, 5), (1, 4)\}$ so that any two simple parts that touched or nearly touched could map to a single simple part. The experimental weighting function assigned high weights (0.45) to the triples $(1, 11, 2)$ and $(2, 11, 3)$ which represent the joining of the three main primitives of the shape and low weights (0.05) to the triples $(1, 11, 4)$ and $(3, 11, 5)$ which represent the joining of the two optional end primitives to the remainder of the shape. The experiment was also run with equal weights as a comparison. The error threshold ϵ was fixed at 0.2.

As expected, the program reported no 0.2-homomorphisms when the weights were equal, since at least one of the four



R_{ij}	Unequal Weights	Equal Weights	R_{ij}
1,11,2	.45	.25	A,AA,B
2,11,3	.45	.25	B,AA,C
1,11,4	.05	.25	C,AA,D
3,11,5	.05	.25	

$$R_{p1} = \emptyset$$

$$R_{p2} = \emptyset$$

$$M = \{(1,2), (2,3), (3,5), (1,4)\}$$

	1	2	3	4	5	11	Not Satisfied	Error With Unequal Weights	Error With Equal Weights
h_1	A	B	C	D	A	AA	(1,11,4) (3,11,5)	.1	.5
h_2	B	C	D	A	D	AA	(3,11,5)	.05	.25

Fig. 15. Illustrates a matching experiment on letter C's where weights were varied. The functions h_1 and h_2 are two of the 0.2-homomorphisms found when the unequal weights were used.

triples was not satisfied. With unequal weights a number of 0.2-homomorphisms were found. Fig. 15 gives two of these homomorphisms, h_1 and h_2 , the triples that were not satisfied by each function, and the corresponding error for each function for the unequal weights experiment and the equal weights experiment.

A second experiment in which weights were varied was run on two letter "T" relations which were created expressly for the experiment. The relations for these shapes is shown in Fig. 16. The prototype shape T1 has several optional extra parts that may or may not be present on a letter "T." The candidate shape T2 is a standard "T" shape. In this experiment, instead of allowing two primitives in the prototype to map to a single primitive in the candidate, the mergeability relation was empty and primitives were allowed to map to the symbol NL meaning "no label." Thus, the effect was to look at partial mappings. Although the approach is impractical (there are too many partial mappings), the experiment does show the effect of the weights on the mappings found.

Fig. 16 shows the experimental weighting function and the equal weights function. The experimental weighting function assigns the highest weights to the relationships that make a shape "T"-shaped and lower weights to the optional relationships. The mapping h in Fig. 16 is one of the 0.2-homomorphisms that was found with the unequal weights, but did not exist when the equal weights were used. The two experiments show that varying the weights can be used to set up a model with many low priority optional parts and relationships.

VI. DISCUSSION AND CONCLUSIONS

We have discussed a general model for a structural description of a shape and defined a specific model based on decomposing the shape into simple parts and intrusions. In our model two relations, the intrusion relation and the protrusion

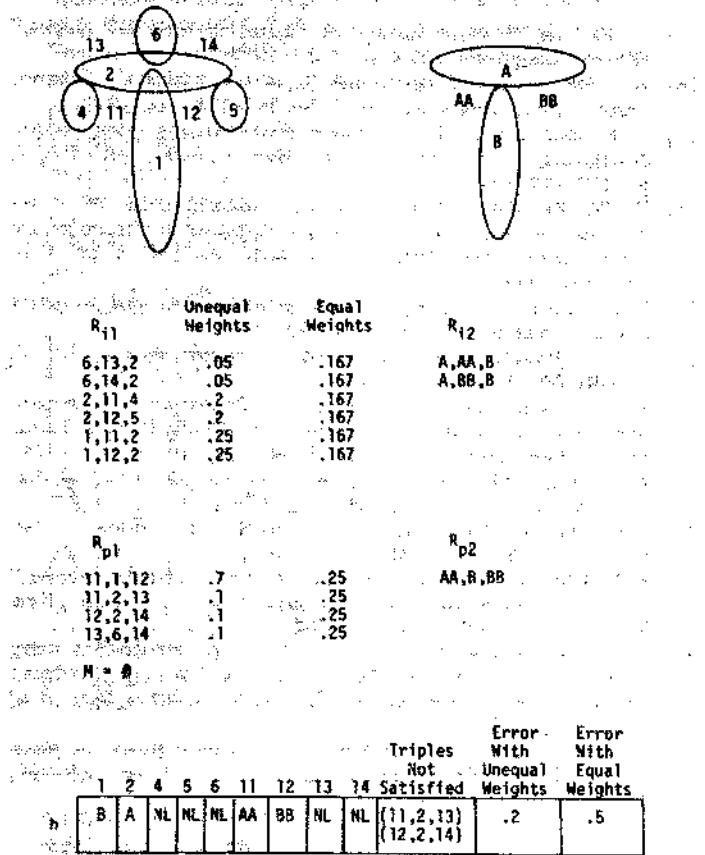


Fig. 16. Illustrates a matching experiment on two letter T's where weights were varied. The function h is a 0.2-homomorphism found when the unequal weights were used.

relation, characterize the shape. The descriptions can be enhanced by adding properties of the simple parts and intrusions.

We have defined shape matching as the problem of finding relational homomorphisms from a prototype shape to a candidate shape. We have discussed homomorphisms on weighted relations and homomorphisms that can assign several candidate primitives to one prototype primitive. We have also discussed some measures for the goodness of a match. Finally, we have described an experimental shape matching procedure that uses a tree search to find homomorphisms from prototype shapes to candidate shapes. At this point, we would like to comment on the results so far and the work yet to be done.

The main contribution of this work is the relational shape model and specialized shape matching procedure. The intrusion and protrusion relations appear to be sufficient to characterize a shape. The model is both simple and powerful. The shape matching procedure incorporating weights, thresholds, and a mergeability relation is a highly flexible matching procedure. Because not all of the parts have to be distinct and not all of the relationships have to be satisfied, inexact matches and partial matches are possible. Thus, shapes that are distorted or obstructed can still match their prototypes. This important characteristic is needed in scene analysis systems.

The entire shape recognition system at present consists of a set of Fortran programs that perform the decomposition of a shape and a Snobol program that performs the shape match-

ing procedure. The automation of extracting the R_i and R_p relations and other information from the results of the decomposition is currently being tackled. For a shape represented by N points, the Fortran programs perform $O(N^3)$ operations for computing relations and $O(N^2)$ operations for graph-theoretic clustering. We hope to reduce the complexity of the $O(N^3)$ program by improving the algorithm used. (The current one is just brute force.) The Snobol program, in the worst case, would search the entire tree since finding homomorphisms is an NP-complete problem. However, in our experiments, the program searched a relatively small portion of the tree. As mentioned previously, the program executed faster on these simple shapes without the relaxation procedure. However, when relaxation was used, the program rarely made a mistake requiring backup. Thus, for very complex shapes with many primitives to be matched, a relaxation procedure should be used. In practice, the number of nodes searched seems to be proportional to the number of primitives of the prototype shape times the number of homomorphisms found. As the error threshold ϵ increases, the program tends to search more nodes and to find more homomorphisms. Occasionally, the program will search fewer nodes with a higher value of ϵ because the order of the search has changed.

The current Snobol shape matching procedure is very slow; approximately $2\frac{1}{4}$ nodes/s are searched. This is mostly due to the fact that the program uses character strings to represent most data objects (triples, lists, units, and labels) and character string operators such as pattern matching and concatenation for manipulating the data objects. The program was initially written in Snobol4 because the powerful data structures and facilities of the language made for a short program and an equally short debugging time. More efficient matching programs are currently being developed in Fortran and Pascal.

The results reported here are only initial experimental results. We expect to perform many more experiments using more extensive data and studying further the effects of different weighting schemes and mergeability relations. We also wish to study the effects of adding properties of the primitives to the model (putting statistical and structural descriptions together), and we will be adding symmetry to the model. Another goal is to develop a shape matching procedure that constructs a third shape from which there are homomorphisms to each of two shapes being tested for similarity. This will take care of the problem of a primitive of the prototype shape mapping to two different primitives of the candidate shape. In summary, the shape recognition procedure as a whole seems to be a promising tool in scene analysis.

REFERENCES

- [1] F. L. Alt, "Digital pattern recognition by moments," *J. Ass. Comput. Mach.*, vol. 11, pp. 240-258, 1962.
- [2] A. K. Agrawala and A. V. Kulkarni, "A sequential approach to the extraction of shape features," *Comput. Graphics Image Processing*, vol. 6, pp. 538-557, Dec. 1977.
- [3] H. G. Barrow, A. P. Ambler, and R. M. Burstall, "Some techniques for recognizing structures in pictures," in *Frontiers of Pattern Recognition*, S. Watanabe, Ed. New York: Academic, 1972, pp. 1-29.
- [4] H. Blum, "A transformation for extracting new descriptions of shape," in *Symp. on Models for the Perception of Speech and Visual Form*. Cambridge, MA: M.I.T. Press, 1964.
- [5] D. G. Corneil and C. C. Gottlieb, "An efficient algorithm for

- graph isomorphism," *J. Ass. Comput. Mach.*, vol. 17, pp. 51-64, Jan. 1970.
- [6] L. S. Davis, "Shape matching using relaxation techniques," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, no. 1, pp. 60-72, 1979.
- [7] —, "Understanding shape: Angles and sides," *IEEE Trans. Comput.*, vol. C-26, pp. 236-242, Mar. 1977.
- [8] —, "Understanding shape II: Symmetry," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 204-212, Mar. 1977.
- [9] M. Eden, "Handwriting and pattern recognition," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 160-166, 1962.
- [10] H. F. Feng and T. Pavlidis, "Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition," *IEEE Trans. Comput.*, vol. C-24, pp. 636-650, June 1975.
- [11] H. Freeman, "Computer processing of line drawing images," *Computing Surveys*, vol. 6, pp. 57-97, Mar. 1974.
- [12] K. S. Fu and S. Y. Lu, "A clustering procedure for syntactic patterns," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 734-742, 1977.
- [13] G. H. Granlund, "Fourier preprocessing for hand print character recognition," *IEEE Trans. Comput.*, vol. C-21, pp. 195-201, Feb. 1972.
- [14] R. M. Haralick and J. Kartus, "Arrangements, homomorphisms, and discrete relaxation," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 600-612, Aug. 1978.
- [15] R. M. Haralick and L. G. Shapiro, "The consistent labeling problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, no. 2, 1979.
- [16] —, "Decomposition of polygonal shapes by clustering," in *Proc. First IEEE Conf. on Pattern Recognition and Image Processing*, June 1977, pp. 183-190.
- [17] R. M. Haralick, "Statistical shape descriptors for simple shapes," unpublished work.
- [18] S. L. Horowitz, "Peak recognition in waveforms," in *Syntactic Pattern Recognition Applications*, K. S. Fu, Ed. Berlin, Germany: Springer-Verlag, 1977.
- [19] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 179-187, Feb. 1962.
- [20] D. Langridge, "On the computation of shape," in *Frontiers of Pattern Recognition*, S. Watanabe, Ed. New York: Academic, 1962, pp. 347-365.
- [21] T. Lozano-Perez, "Parsing intensity profiles," *Comput. Graphics Image Processing*, vol. 6, pp. 43-60, 1977.
- [22] K. Maruyama, "A study of visual shape perception," Dep. Comput. Sci., Univ. of Illinois, Rep. VIVCDSC-R-72-533, Oct. 1972.
- [23] J. F. O'Callaghan, "Recovery of perceptual shape organization from simple closed boundaries," *Comput. Graphics Image Processing*, vol. 3, pp. 300-312, Dec. 1974.
- [24] T. Pavlidis and F. Ali, "A hierarchical syntactic shape analyzer," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 2-9, Jan. 1979.
- [25] T. Pavlidis, "Representation of figures by labelled graphs," *Pattern Recognition*, vol. 4, pp. 5-17, 1972.
- [26] —, "A review of algorithms for shape analysis," *Comput. Graphics Image Processing*, vol. 7, pp. 243-258, Apr. 1978.
- [27] E. Persoon and K. S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 170-179, 1977.
- [28] C. W. Richard and H. Hernami, "Identification of three-dimensional objects using Fourier descriptors of the boundary curve," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-4, pp. 371-377, July 1974.
- [29] B. Rosenberg, "The analysis of convex blobs," *Comput. Graphics Image Processing*, vol. 1, pp. 183-192, 1972.
- [30] —, "Computing dominant points on simple shapes," *Int. J. Man-Machine Studies*, vol. 6, pp. 1-12, 1975.
- [31] L. G. Shapiro and R. M. Haralick, "Decomposition of two-dimensional shapes by graph-theoretic clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 10-20, Jan. 1979.
- [32] A. C. Shaw, "Parsing of graph-representable pictures," *J. Ass. Comput. Mach.*, vol. 17, pp. 453-481, July 1970.
- [33] J. R. Ullman, "An algorithm for subgraph isomorphism," *J. Ass. Comput. Mach.*, vol. 23, pp. 31-42, Jan. 1976.
- [34] P. H. Winston, "Learning structural descriptions from examples," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York: McGraw-Hill, 1975, pp. 157-209.
- [35] K. C. You and K. S. Fu, "Syntactic shape recognition using attributed grammars," in *Proc. 1978 EIA Symp. on Emerging Patterns in Automatic Imagery Pattern Recognition*, Apr. 3-4, Gaithersburg, MD.
- [36] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. Comput.*, vol. C-21, pp. 269-281, 1972.



Linda G. Shapiro was born in Chicago, IL, in 1949. She received the B.S. degree in mathematics from the University of Illinois at Urbana-Champaign in 1970, and the M.S. and Ph.D. degrees in computer science from the University of Iowa, Iowa City, in 1972 and 1974, respectively.

She was an Assistant Professor of Computer Science at Kansas State University, Manhattan, from 1974 to 1978. She is currently an Assistant Professor at Virginia Polytechnic Institute and State University, Blacksburg. Her research interests include scene analysis, pattern recognition, spatial information systems, computer graphics, and data structures. She has completed an undergraduate textbook on data structures with R. Baron.

Dr. Shapiro is a member of the IEEE Computer Society, the Association for Computing Machinery, and the Pattern Recognition Society.

Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography

Martin A. Fischler and Robert C. Bolles
SRI International

A new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data is introduced. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and is thus ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors. A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): Given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. In response to a RANSAC requirement, new results are derived on the minimum number of landmarks needed to obtain a solution, and algorithms are presented for computing these minimum-markmark solutions in closed form. These results provide the basis for an automatic system that can solve the LDP under difficult viewing

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

The work reported herein was supported by the Defense Advanced Research Projects Agency under Contract Nos. DAAG29-76-C-0057 and MDA903-79-C-0588.

Authors' Present Address: Martin A. Fischler and Robert C. Bolles, Artificial Intelligence Center, SRI International, Menlo Park CA 94025.

© 1981 ACM 0001-0782/81/0600-0381\$00.75

and analysis conditions. Implementation details and computational examples are also presented.

Key Words and Phrases: model fitting, scene analysis, camera calibration, image matching, location determination, automated cartography.

CR Categories: 3.60, 3.61, 3.71, 5.0, 8.1, 8.2

I. Introduction

We introduce a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data; and illustrate its use in scene analysis and automated cartography. The application discussed, the location determination problem (LDP), is treated at a level beyond that of a mere example of the use of the RANSAC paradigm; new basic findings concerning the conditions under which the LDP can be solved are presented and a comprehensive approach to the solution of this problem that we anticipate will have near-term practical applications is described.

To a large extent, scene analysis (and, in fact, science in general) is concerned with the interpretation of sensed data in terms of a set of predefined models. Conceptually, interpretation involves two distinct activities: First, there is the problem of finding the best match between the data and one of the available models (the classification problem); Second, there is the problem of computing the best values for the free parameters of the selected model (the parameter estimation problem). In practice, these two problems are not independent—a solution to the parameter estimation problem is often required to solve the classification problem.

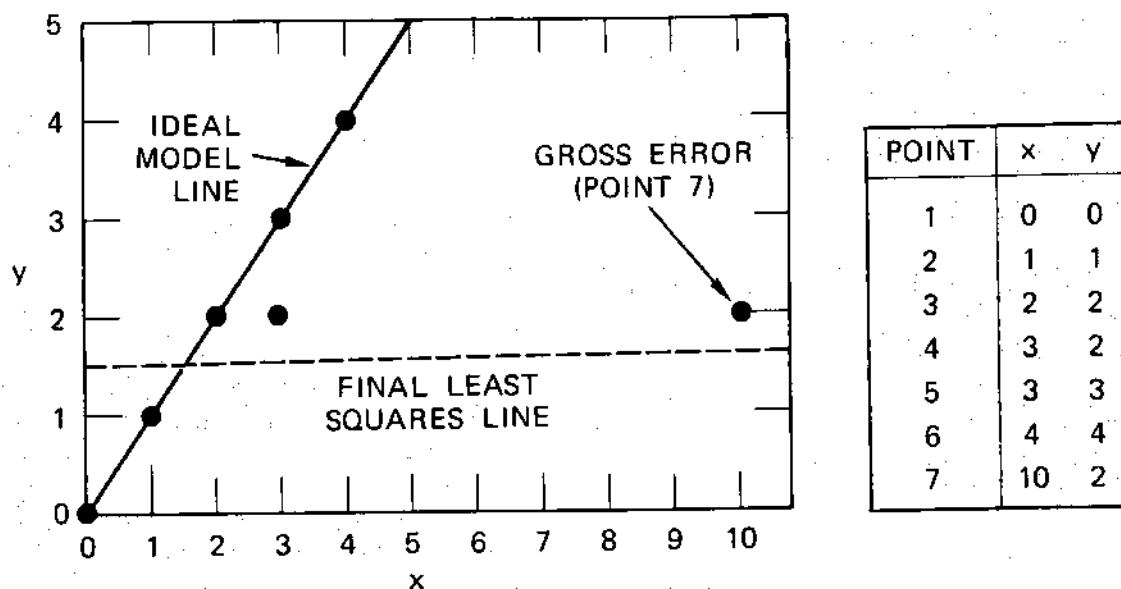
Classical techniques for parameter estimation, such as least squares, optimize (according to a specified objective function) the fit of a functional description (model) to *all* of the presented data. These techniques have no internal mechanisms for detecting and rejecting gross errors. They are averaging techniques that rely on the assumption (the smoothing assumption) that the maximum expected deviation of any datum from the assumed model is a direct function of the size of the data set, and thus regardless of the size of the data set, there will always be enough good values to smooth out any gross deviations.

In many practical parameter estimation problems the smoothing assumption does not hold; i.e., the data contain uncompensated gross errors. To deal with this situation, several heuristics have been proposed. The technique usually employed is some variation of first using all the data to derive the model parameters, then locating the datum that is farthest from agreement with the instantiated model, assuming that it is a gross error, deleting it, and iterating this process until either the maximum deviation is less than some preset threshold or until there is no longer sufficient data to proceed.

It can easily be shown that a single gross error ("poisoned point"), mixed in with a set of good data, can

Fig. 1. Failure of Least Squares (and the "Throwing Out The Worst Residual" Heuristic), to Deal with an Erroneous Data Point.

PROBLEM: Given the set of seven (x,y) pairs shown in the plot, find a best fit line; assuming that no valid datum deviates from this line by more than 0.8 units.



COMMENT: Six of the seven points are valid data and can be fit by the solid line. Using Least Squares (and the "throwing out the worst residual" heuristic), we terminate after four iterations with four remaining points, including the gross error at (10,2) fit by the dashed line.

SUCCESSIVE LEAST SQUARES APPROXIMATIONS		
ITERATION	DATA SET	FITTING LINE
1	1, 2, 3, 4, 5, 6, 7	$1.48 + .16x$
2	1, 2, 3, 4, 5, 7	$1.25 + .13x$
3	1, 2, 3, 4, 7	$0.96 + .14x$
4	2, 3, 4, 7	$1.51 + .06x$

COMPUTATION OF RESIDUALS				
POINT	ITERATION 1 RESIDUALS	ITERATION 2 RESIDUALS	ITERATION 3 RESIDUALS	ITERATION 4 RESIDUALS
1	-1.48	-1.25	-.96*	—
2	-0.64	-0.38	-.10	-.57
3	-0.20	0.49	.76	.37
4	0.05	0.36	.63	.31
5	1.05	1.36*	—	—
6	1.89*	—	—	—
7	-1.06	-0.57	-.33	-.11

cause the above heuristic to fail (for example, see Figure 1). It is our contention that averaging is not an appropriate technique to apply to an unverified data set.

In the following section we introduce the RANSAC paradigm, which is capable of smoothing data that contain a significant percentage of gross errors. This paradigm is particularly applicable to scene analysis because local feature detectors, which often make mistakes, are the source of the data provided to the interpretation algorithms. Local feature detectors make two types of errors—classification errors and measurement errors. Classification errors occur when a feature detector incorrectly identifies a portion of an image as an occurrence of a feature. Measurement errors occur when the feature detector correctly identifies the feature, but slightly miscalculates one of its parameters (e.g., its image location). Measurement errors generally follow a normal distribution, and therefore the smoothing assumption is applicable to them. Classification errors, however, are gross errors, having a significantly larger effect than measurement errors, and do not average out.

In the final sections of this paper the application of RANSAC to the location determination problem is discussed:

Given a set of "landmarks" ("control points"), whose locations are known in some coordinate frame, determine the location (relative to the coordinate frame of the landmarks) of that point in space from which an image of the landmarks was obtained.

In response to a RANSAC requirement, some new results are derived on the minimum number of landmarks needed to obtain a solution, and then algorithms are presented for computing these minimum-landmark solutions in closed form. (Conventional techniques are iterative and require a good initial guess to assure convergence.) These results form the basis for an automatic system that can solve the LDP under severe viewing and analysis conditions. In particular, the system performs properly even if a significant number of landmarks are incorrectly located due to low visibility, terrain changes, or image analysis errors. Implementation details and experimental results are presented to complete our description of the LDP application.

II. Random Sample Consensus

The RANSAC procedure is opposite to that of conventional smoothing techniques. Rather than using as much of the data as possible to obtain an initial solution and then attempting to eliminate the invalid data points, RANSAC uses as small an initial data set as feasible and enlarges this set with consistent data when possible. For example, given the task of fitting an arc of a circle to a set of two-dimensional points, the RANSAC approach would be to select a set of three points (since three points are required to determine a circle), compute the center and radius of the implied circle, and count the number of points that are close enough to that circle to suggest

their compatibility with it (i.e., their deviations are small enough to be measurement errors). If there are enough compatible points, RANSAC would employ a smoothing technique such as least squares, to compute an improved estimate for the parameters of the circle now that a set of mutually consistent points has been identified.

The RANSAC paradigm is more formally stated as follows:

Given a model that requires a minimum of n data points to instantiate its free parameters, and a set of data points P such that the number of points in P is greater than n ($\#(P) \geq n$), randomly select a subset S_1 of n data points from P and instantiate the model. Use the instantiated model M_1 to determine the subset S_1^* of points in P that are within some error tolerance of M_1 . The set S_1^* is called the consensus set of S_1 .

If $\#(S_1^*)$ is greater than some threshold t , which is a function of the estimate of the number of gross errors in P , use S_1^* to compute (possibly using least squares) a new model M_1^* .

If $\#(S_1^*)$ is less than t , randomly select a new subset S_2 and repeat the above process. If, after some predetermined number of trials, no consensus set with t or more members has been found, either solve the model with the largest consensus set found, or terminate in failure.

There are two obvious improvements to the above algorithm: First, if there is a problem related rationale for selecting points to form the S 's, use a deterministic selection process instead of a random one; second, once a suitable consensus set S^* has been found and a model M^* instantiated, add any new points from P that are consistent with M^* to S^* and compute a new model on the basis of this larger set.

The RANSAC paradigm contains three unspecified parameters: (1) the error tolerance used to determine whether or not a point is compatible with a model, (2) the number of subsets to try, and (3) the threshold t , which is the number of compatible points used to imply that the correct model has been found. Methods are discussed for computing reasonable values for these parameters in the following subsections.

A. Error Tolerance For Establishing Datum/Model Compatibility

The deviation of a datum from a model is a function of the error associated with the datum and the error associated with the model (which, in part, is a function of the errors associated with the data used to instantiate the model). If the model is a simple function of the data points, it may be practical to establish reasonable bounds on error tolerance analytically. However, this straightforward approach is often unworkable; for such cases it is generally possible to estimate bounds on error tolerance experimentally. Sample deviations can be produced by perturbing the data, computing the model, and measuring the implied errors. The error tolerance could then be set at one or two standard deviations beyond the measured average error.

The expected deviation of a datum from an assumed model is generally a function of the datum, and therefore the error tolerance should be different for each datum. However, the variation in error tolerances is usually

relatively small compared to the size of a gross error. Thus, a single error tolerance for all data is often sufficient.

B. The Maximum Number of Attempts to Find a Consensus Set

The decision to stop selecting new subsets of P can be based upon the expected number of trials k required to select a subset of n good data points. Let w be the probability that any selected data point is within the error tolerance of the model. Then we have:

$$E(k) = b + 2*(1-b)*b + 3*(1-b)^2*b \\ \dots + i*(1-b)^{i-1}*b + \dots,$$

$$E(k) = b*[1 + 2*a + 3*a^2 \dots + i*a^{i-1} + \dots],$$

where $E(k)$ is the expected value of k , $b = w^n$, and $a = (1 - b)$.

An identity for the sum of a geometric series is

$$a/(1-a) = a + a^2 + a^3 \dots + a^i + \dots.$$

Differentiating the above identity with respect to a , we have:

$$1/(1-a)^2 = 1 + 2*a + 3*a^2 \dots + i*a^{i-1} + \dots.$$

Thus,

$$E(k) = 1/b = w^{-n}$$

The following is a tabulation of some values of $E(k)$ for corresponding values of n and w :

w	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
0.9	1.1	1.2	1.4	1.5	1.7	1.9
0.8	1.3	1.6	2.0	2.4	3.0	3.8
0.7	1.4	2.0	2.9	4.2	5.9	8.5
0.6	1.7	2.8	4.6	7.7	13	21
0.5	2.0	4.0	8.0	16	32	64
0.4	2.5	6.3	16	39	98	244
0.3	3.3	11	37	123	412	—
0.2	5.0	25	125	625	—	—

In general, we would probably want to exceed $E(k)$ trials by one or two standard deviations before we give up. Note that the standard deviation of k , $SD(k)$, is given by:

$$SD(k) = \sqrt{[E(k^2) - E(k)^2]}.$$

Then

$$E(k^2) = \sum_{i=0}^{\infty} (b*i^2*a^{i-1}), \\ = \sum_{i=0}^{\infty} [b*i*(i-1)*a^{i-1}] + \sum_{i=0}^{\infty} (b*i*a^{i-1}),$$

but (using the geometric series identity and two differentiations):

$$2a/(1-a)^3 = \sum_{i=0}^{\infty} (i*(i-1)*a^{i-1}).$$

Thus,

$$E(k^2) = (2 - b)/(b^2),$$

and

$$SD(k) = [\sqrt{(1-w^n)}] * (1/w^n).$$

Note that generally $SD(k)$ will be approximately equal to $E(k)$; thus, for example, if ($w = 0.5$) and ($n = 4$), then $E(k) = 16$ and $SD(k) = 15.5$. This means that one might want to try two or three times the expected number of random selections implied by k (as tabulated above) to obtain a consensus set of more than t members.

From a slightly different point of view, if we want to ensure with probability z that at least one of our random selections is an error-free set of n data points, then we must expect to make at least k selections (n data points per selection), where

$$(1-b)^k = (1-z),$$

$$k = [\log(1-z)] / [\log(1-b)].$$

For example, if ($w = 0.5$) and ($n = 4$), then ($b = 1/16$). To obtain a 90 percent assurance of making at least one error-free selection,

$$k = \log(0.1) / \log(15/16) = 35.7.$$

Note that if $w^n \ll 1$, then $k \approx \log(1-z)E(k)$. Thus if $z = 0.90$ and $w^n \ll 1$, then $k \approx 2.3E(k)$; if $z = 0.95$ and $w^n \ll 1$, then $k \approx 3.0E(k)$.

C. A Lower Bound On the Size of an Acceptable Consensus Set

The threshold t , an unspecified parameter in the formal statement of the RANSAC paradigm, is used as the basis for determining that an n subset of P has been found that implies a sufficiently large consensus set to permit the algorithm to terminate. Thus, t must be chosen large enough to satisfy two purposes: that the correct model has been found for the data, and that a sufficient number of mutually consistent points have been found to satisfy the needs of the final smoothing procedure (which computes improved estimates for the model parameters).

To ensure against the possibility of the final consensus set being compatible with an incorrect model, and assuming that y is the probability that any given data point is within the error tolerance of an incorrect model, we would like y^{t-n} to be very small. While there is no general way of precisely determining y , it is certainly reasonable to assume that it is less than w (w is the *a priori* probability that a given data point is within the error tolerance of the correct model). Assuming $y < 0.5$, a value of $t - n$ equal to 5 will provide a better than 95 percent probability that compatibility with an incorrect model will not occur.

To satisfy the needs of the final smoothing procedure, the particular procedure to be employed must be specified. If least-squares smoothing is to be used, there are many situations where formal methods can be invoked

to determine the number of points required to produce a desired precision [10].

D. Example

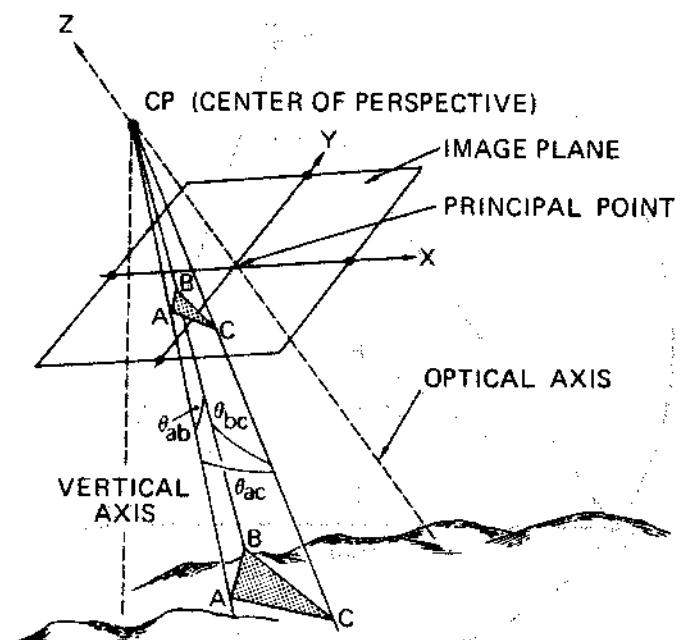
Let us apply RANSAC to the example described in Figure 1. A value of w (the probability that any selected data point is within the error tolerance of the model) equal to 0.85 is consistent with the data, and a tolerance (to establish datum/model compatibility) of 0.8 units was supplied as part of the problem statement. The RANSAC-supplied model will be accepted without external smoothing of the final consensus set; thus, we would like to obtain a consensus set that contains all seven data points. Since one of these points is a gross error, it is obvious that we will not find a consensus set of the desired size, and so we will terminate with the largest set we are able to find. The theory presented earlier indicates that if we take two data points at a time, compute the line through them and measure the deviations of the remaining points from this line, we should expect to find a suitable consensus set within two or three trials; however, because of the limited amount of data, we might be willing to try all 21 combinations to find the largest consensus set. In either case, we easily find the consensus set containing the six valid data points and the line that they imply.

III. The Location Determination Problem (LDP)

A basic problem in image analysis is establishing a correspondence between the elements of two representations of a given scene. One variation of this problem, especially important in cartography, is determining the location in space from which an image or photograph was obtained by recognizing a set of landmarks (control points) appearing in the image (this is variously called the problem of determining the elements of exterior camera orientation, or the camera calibration problem, or the image-to-database correspondence problem). It is routinely solved using a least-squares technique [11, 8] with a human operator interactively establishing the association between image points and the three-dimensional coordinates of the corresponding control points. However, in a fully automated system, where the correspondences must be based on the decisions of marginally competent feature detectors, least squares is often incapable of dealing with the gross errors that may result; this consideration, discussed at length in Sec. II, is illustrated for the LDP in an example presented in Sec. IV.

In this section a new solution to the LDP is presented based on the RANSAC paradigm, which is unique in its ability to tolerate gross errors in the input data. We will first examine the conditions under which a solution to the LDP is possible and describe new results concerning this question; we then present a complete description of the RANSAC-based algorithm, and finally, describe experimental results obtained through use of the algorithm.

Fig. 2. Geometry of the Location Determination Problem.



The LDP is formally defined as follows:

Given a set of m control points, whose 3-dimensional coordinates are known in some coordinate frame, and given an image in which some subset of the m control points is visible, determine the location (relative to the coordinate system of the control points) from which the image was obtained.

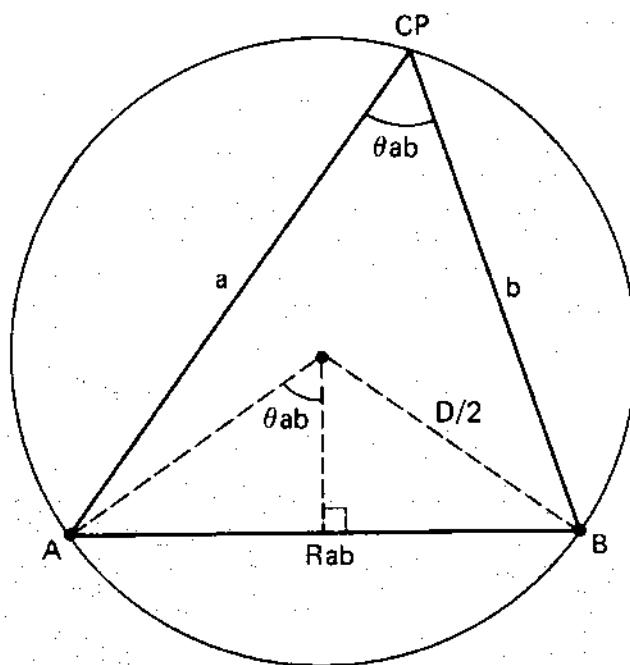
We will initially assume that we know the correspondences between n image points and control points; later we consider the situation in which some of these correspondences are invalid. We will also assume that both the principal point in the image plane (where the optical axis of the camera pierces the image plane) and the focal length (distance from the center of perspective to the principal point in the image plane) of the imaging system are known; thus (see Figure 2) we can easily compute the angle to any pair of control points from the center of perspective (CP). Finally, we assume that the camera resides outside and above a convex hull enclosing the control points.

We will later demonstrate (Appendix A) that if we can compute the lengths of the rays from the CP to three of the control points then we can directly solve for the location of the CP (and the orientation of the image plane if desired). Thus, an equivalent but mathematically more concise statement of the LDP is

Given the relative spatial locations of n control points, and given the angle to every pair of control points from an additional point called the Center of Perspective (CP), find the lengths of the line segments ("legs") joining the CP to each of the control points. We call this the "perspective- n -point" problem (PnP).

In order to apply the RANSAC paradigm, we wish to determine the smallest value of n for which it is possible to solve the PnP problem.

Fig. 3. Geometry of the P2P Problem.



$$\sin \theta_{ab} = \frac{R_{ab}/2}{D/2}$$

$$D = \frac{R_{ab}}{\sin \theta_{ab}}$$

A. Solution of the Perspective-n-Point Problem

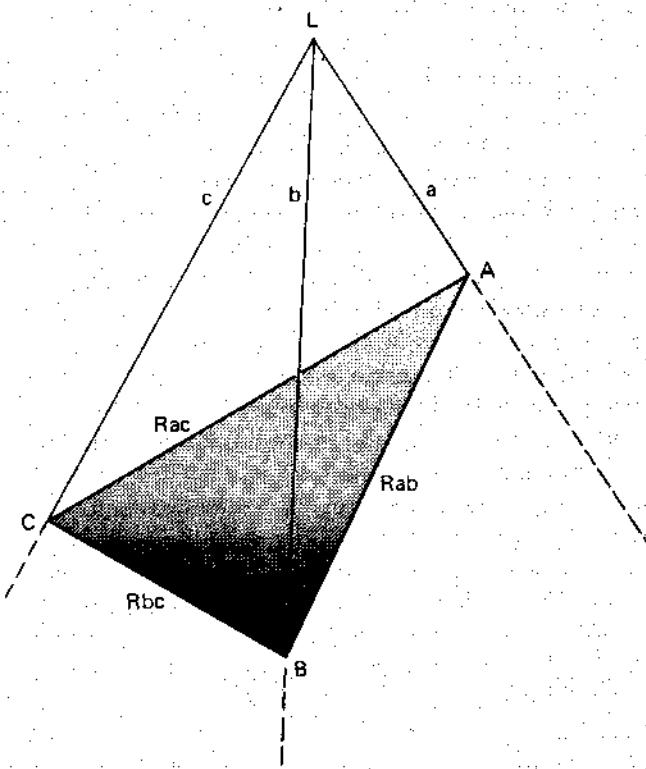
The P1P problem ($n = 1$) provides no constraining information, and thus an infinity of solutions is possible. The P2P problem ($n = 2$), illustrated in Figure 3, also admits an infinity of solutions; the CP can reside anywhere on a circle of diameter $R_{ab}/\sin(\theta_{ab})$, rotated in space about the chord (line) joining the two control points A and B.

The P3P problem ($n = 3$) requires that we determine the lengths of the three legs of a tetrahedron, given the base dimensions and the face angles of the opposing trihedral angle (see Figure 4). The solution to this problem is implied by the three equations [A^*]:

$$\begin{aligned} (R_{ab})^2 &= a^2 + b^2 - 2*a*b*[\cos(\theta_{ab})] \\ (R_{ac})^2 &= a^2 + c^2 - 2*a*c*[\cos(\theta_{ac})] \quad [A^*] \\ (R_{bc})^2 &= b^2 + c^2 - 2*b*c*[\cos(\theta_{bc})] \end{aligned}$$

It is known that n independent polynomial equations, in n unknowns, can have no more solutions than the product of their respective degrees [2]. Thus, the system A^* can have a maximum of eight solutions. However, because every term in the system A^* is either a constant or of second degree, for every real positive solution there is a geometrically isomorphic negative solution. Thus, there are at most four positive solutions to A^* , and in Figure 5 we show an example demonstrating that the upper bound of four solutions is attainable.

Fig. 4. Geometry of the P3P Problem (L is the Center of Perspective).



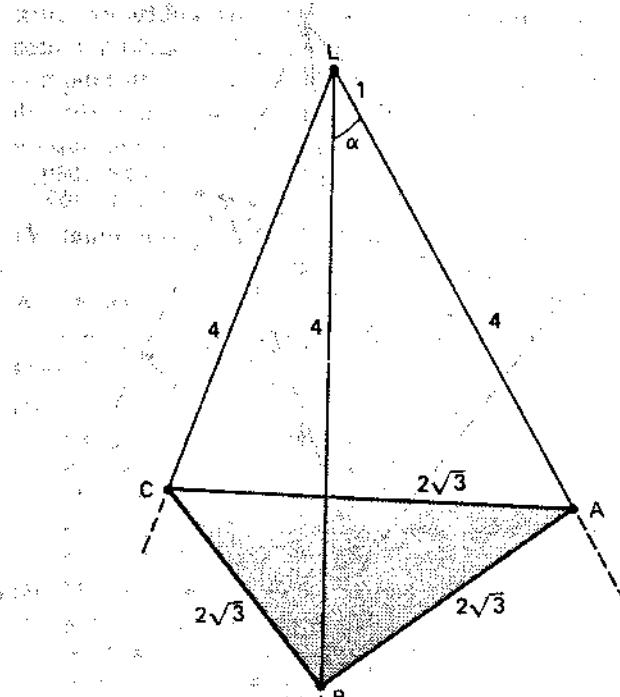
In Appendix A we derive an explicit algebraic solution for the system A^* . This is accomplished by reducing A^* to a biquadratic (quartic) polynomial in one unknown representing the ratio of two legs of the tetrahedron, and then directly solving this equation (we also present a very simple iterative method for obtaining the solutions from the given problem data).

For the case $n = 4$, when all four control points lie in a common plane (not containing the CP, and such that no more than two of the control points lie on any single line), we provide a technique in Appendix B that will always produce a unique solution. Surprisingly, when all four control points do not lie in the same plane, a unique solution cannot always be assured; for example, Figure 6 shows that at least two solutions are possible for the P4P problem with the control points in "general position."

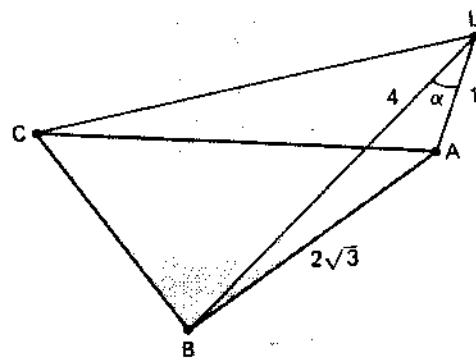
To solve for the location of the CP in the case of four nonplanar control points, we can use the algorithm presented in Appendix A on two distinct subsets of the control points taken three at a time; the solution(s) common to both subsets locate the CP to within the ambiguity inherent in the given information.

The approach used to construct the example shown in Figure 6 can be extended to any number of additional points. It is based on the principle depicted in Figure 3: If the CP and any number of control points lie on the same circle, then the angle between any pair of control points and the CP will be independent of the location on the circle of the CP (and hence the location of the CP cannot be determined). Thus, we are able to construct the example shown in Figure 7, in which five control points in general position imply two solutions to the P5P

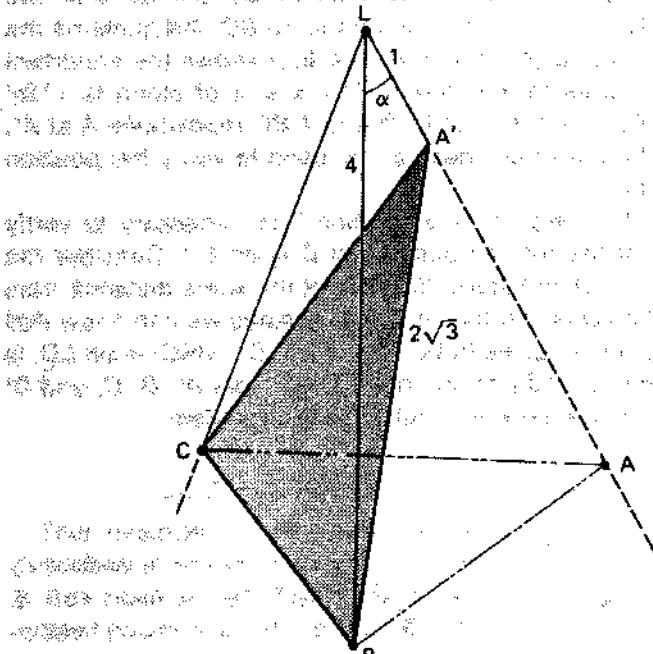
Fig. 5. An Example Showing Four Distinct Solutions to a P3P Problem.



(a)



(b)



(c)

Consider the tetrahedron in Figure 5(a). The base ABC is an equilateral triangle and the "legs" (i.e., LA , LB , and LC) are all equal. Therefore, the three face angles at L (i.e., $\angle ALB$, $\angle ALC$, and $\angle BLC$) are all equal. By the law of cosines we have:

$$\cos(\alpha) = 5/8.$$

This tetrahedron defines one solution to a P3P problem. A second solution is shown in Figure 5(b). It is obtained from the first by rotating L' about BC . It is necessary to verify that the length of $L'A$ can be 1, given the rigid triangle ABC and the angle α . From the law of cosines we have:

$$(2\sqrt{3})^2 = 4^2 + (L'A)^2 - 2 \cdot 4 \cdot (L'A) \cdot (5/8)$$

which reduces to:

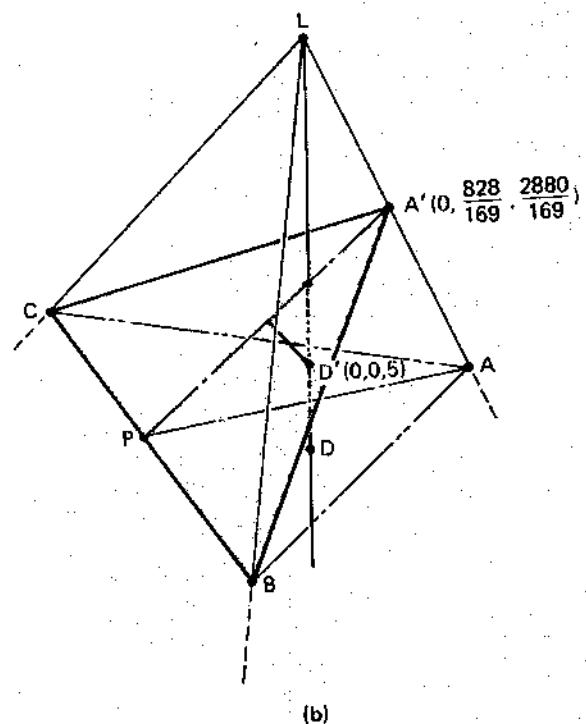
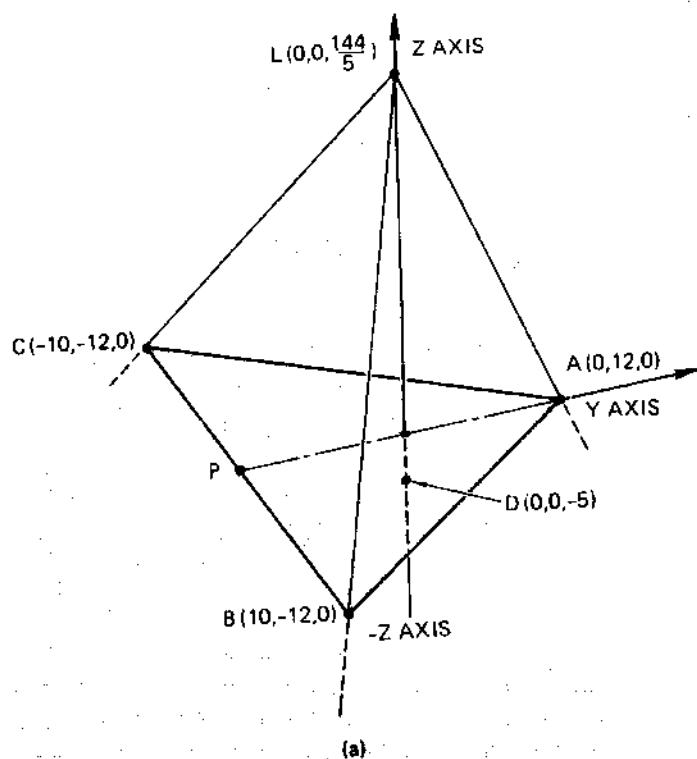
$$(L'A - 1) \cdot (L'A - 4) = 0.$$

Therefore, $L'A$ can be either 1 or 4. Figure 5(a) illustrates the $L'A = 4$ case and Figure 5(b) illustrates the $L'A = 1$ case.

Notice that repositioning the base triangle so that its vertices move to different locations on the legs is equivalent to repositioning L . Figure 5(c) shows the position of the base triangle that corresponds to the second solution.

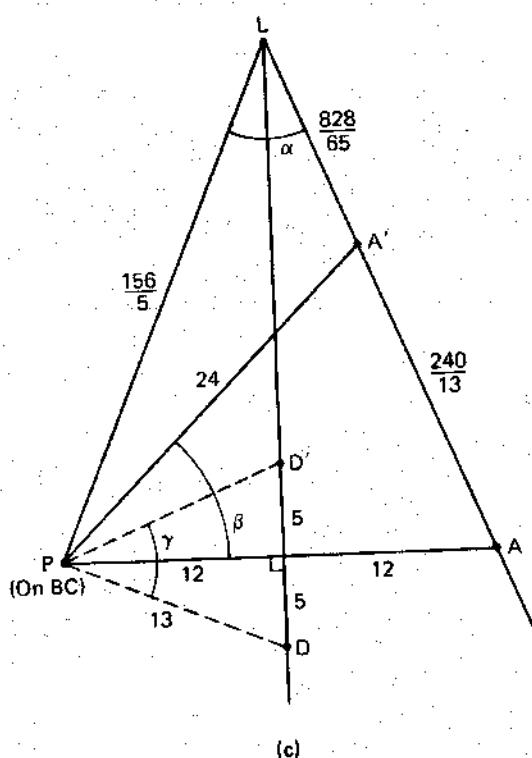
Since the tetrahedron in Figure 5(a) is threefold rotationally symmetric, two more solutions can be obtained by rotating the triangle about AB and AC .

Fig. 6. An Example of a P4P Problem with Two Solutions.



(a)

(b)



(c)

problem. While the same technique will work for six or more control points, four or more of these points must now lie in the same plane and are thus no longer in general position.

To prove that six (or more) control points in general position will always produce a unique solution to the P6P problem, we note that for this case we can always

Figure 6(a) specifies a P4P problem and demonstrates one solution. A second solution can be achieved by rotating the base about BC so that A is positioned at a different point on its leg (see Figure 6(b)). To verify that this is a valid solution consider the plane $X = 0$, which is normal to BC and contains the points, L , A , and D . Figure 6(c) shows the important features in this plane. The cosine of alpha is $119/169$. A rotation of beta about BC repositions A at A' . The law of cosines can be used to verify the position of A' .

To complete this solution it is necessary to verify that the rotated position of D is on LD . Consider the point D' in Figure 6(c). It is at the same distance from P as D is and by the law of cosines we can show that gamma equals beta. Therefore, D' , which is on LD , is the rotated position of D . The points A' , B , C , and D' form the second solution to the problem.

solve for the 12 coefficients of the 3×4 matrix T that specifies the mapping (in homogeneous coordinates) from 3-space to 2-space; each of the six correspondences provides three new equations and introduces one additional unknown (the homogeneous coordinate scale factor). Thus, for six control points, we have 18 linear equations to solve for the 18 unknowns (actually, it can

be shown that, at most, 17 of the unknowns are independent). Given the transformation matrix T , we can construct an additional (synthetic) control point lying in a common plane with three of the given control points and compute its location in the image plane; the technique described in Appendix B can now be used to find a unique solution.

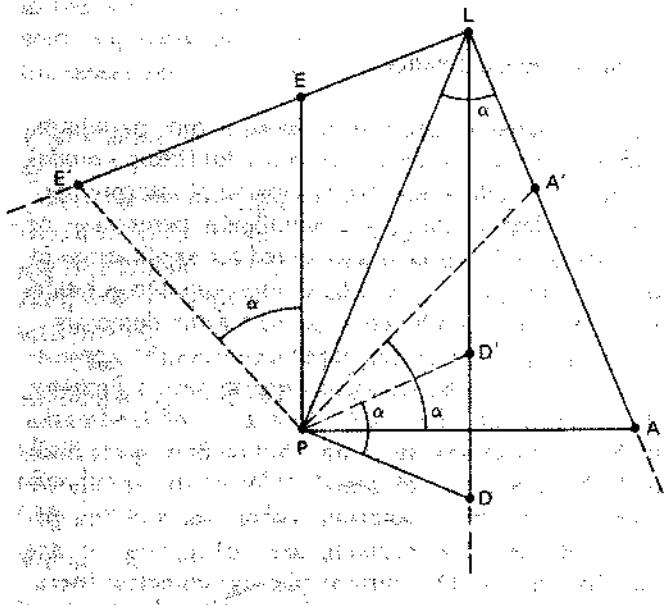
IV. Implementation Details and Experimental Results

A. The RANSAC/LD Algorithm

The RANSAC/LD algorithm accepts as input the following data:

- (1) A list L of m 6-tuples—each 6-tuple containing the 3-D spatial coordinates of a control point, its corresponding 2-D image plane coordinates, and an optional number giving the expected error (in pixels) of the given location in the image plane.
- (2) The focal length of the imaging system and the image plane coordinates of the principal point.
- (3) The probability $(1 - w)$ that a 6-tuple contains a gross mismatch.
- (4) A “confidence” number G which is used to set the internal thresholds for acceptance of intermediate results contributing to a solution. A confidence number of one forces very conservative behavior on the algorithm; a confidence number of zero will call almost anything a valid solution.

Fig. 7. An Example of a P5P Problem with Two Solutions.



This example is the same as the P4P example described in Figure 6 except that a fifth control point, E , has been added. The initial position for E and its rotated position, E' , are shown in Figure 7. The points E and E' were constructed to be the mirror images of A' and A about the line LP ; therefore, a rotation of α about P repositions E at E' . One solution of the P5P problem is formed by points A , B , C , D , and E (shown in Figure 6(a)) plus point E . The second solution is formed by points A' , B , C , D' , and E' . Consequently there are two different positions of L such that all five points lie on their appropriate legs.

The RANSAC/LD algorithm produces as output the following information:

- (1) The 3-D spatial coordinates of the lens center (i.e., the Center of Perspective), and an estimate of the corresponding error.
- (2) The spatial orientation of the image plane.

The RANSAC/LD algorithm operates as follows:

- (1) Three 6-tuples are selected from list L by a quasirandom method that ensures a reasonable spatial distribution for the corresponding control points. This initial selection is called S_1 .
- (2) The CP (called CPI) corresponding to selection S_1 is determined using the closed-form solution provided in Appendix A; multiple solutions are treated as if they were obtained from separate selections in the following steps.
- (3) The error in the derived location of CPI is estimated by perturbing the given image plane coordinates of the three selected control points (either by the amount specified in the 6-tuples or by a default value of one pixel), and recomputing the effect this would have on the location of the CPI.
- (4) Given the error estimate for the CPI, we use the technique described in [1] to determine error ellipses (dimensions based upon the supplied confidence number) in the image plane for each of the control points specified in list L ; if the associated image coordinates reside within the corresponding error ellipse, then the 6-tuple is appended to the consensus set S_1/CPI .
- (5) If the size of S_1/CPI equals or exceeds some threshold value (m , nominally equal to a value between 7 and mw), then the consensus set S_1/CPI is supplied to a least-squares routine (see [1] or [7]) for final determination of the CP location and image plane orientation.¹ Otherwise, the above steps are repeated with a new random selection S_2, S_3, \dots
- (6) If the number of iterations of the above steps exceeds $k = [\log(1 - G)]/[\log(1 - w^3)]$, then the largest consensus set found so far is used to compute the final solution (or we terminate in failure if this largest consensus set contains fewer than six members).

B. Experimental Results

To demonstrate the validity of our theoretical results, we performed three experiments. In the first experiment, we found a specific LDP in which the common least-squares pruning heuristic failed, and showed that RANSAC successfully solved this problem. In the second experiment, we applied RANSAC to 50 synthetic problems in order to check the reliability of the approach over a wide range of parameter values. In the third experiment we used standard feature detection techniques to locate landmarks in an aerial image and then used RANSAC to determine the position and orientation of the camera.

C. A Location Determination Problem Example of a Least Squares Pruning Error

The LDP in this experiment was based upon 20 landmarks and their locations in an image. Five of the 20 correspondences were gross errors; that is, their given locations in the image were further than 10 pixels from their actual locations. The image locations for the good

¹An alternative to least squares would be to average the parameters computed from random triples in the consensus set that fall within (say) the center 50 percent of the associated histogram.

correspondences were normally distributed about their actual locations with a standard deviation of one pixel.

The heuristic to prune gross errors was the following:

- * Use all of the correspondences to instantiate a model.
- * On the basis of that model, delete the correspondence that has the largest deviation from its predicted image location.
- * Instantiate a new model without that correspondence.
- * If the new model implies a normalized error for the deleted correspondence that is larger than three standard deviations, assume that it is a gross error, leave it out, and continue deleting correspondences. Otherwise, assume that it is a good correspondence and return the model that included it as the solution to the problem.

This heuristic successfully deleted two of the gross errors; but after deleting a third, it decided that the new model did not imply a significantly large error, so it returned a solution based upon 18 correspondences, three of which were gross errors. When RANSAC was applied to this problem, it located the correct solution on the second triple of selected points. The final consensus set contained all of the good correspondences and none of the gross errors.

D. 50 Synthetic Location Determination Problems

In this experiment RANSAC was applied to 50 synthetic LDPs. Each problem was based upon 30 landmark-to-image correspondences. A range of probabilities were used to determine the number of gross errors in the problems; the image location of a gross error was at least 10 pixels from its actual location. The location of a good correspondence was distributed about its actual location with a normal distribution having a standard deviation of one pixel. Two different camera positions were used—one looking straight down on the landmarks and one looking at them from an oblique angle. The RANSAC algorithm described earlier in this section was applied to these problems; however, the simple iterative technique described in Appendix A was used to locate solutions to the P3P problems in place of the closed form method also described in that appendix, and a second least-squares fit was used to extend the final consensus set (as suggested in Sec. II of this paper). Table I summarizes the results for ten typical problems (RANSAC successfully avoided including a gross error in its final consensus set in all of the problems); in five of these problems the probability of a good correspondence was 0.8, and in the other five problems, it was 0.6. The execution time for the current program is approximately 1 sec for each camera position considered.

E. A "Real" Location Determination Problem

Cross correlation was used to locate 25 landmarks in an aerial image taken from approximately 4,000 ft with a 6 in. lens. The image was digitized on a grid of 2,000 \times 2,000 pixels, which implies a ground resolution of approximately 2 ft per pixel. Three gross errors were

Table I. Typical Experimental Results Using RANSAC

No. of Good Correspondences	No. of Correspondences in Final Consensus Set	No. of Triples Considered	No. of Camera Positions Considered
<i>w = 0.8</i>			
22	19	6	10
23	23	1	3
19	19	2	3
25	25	1	2
24	23	3	8
<i>w = 0.6</i>			
21	20	11	21
17	17	1	1
17	16	6	8
18	16	9	21
21	18	9	15

made by the correlation feature detector. When RANSAC was applied to this problem, it located a consensus set of 17 on the first triple selected and then extended that set to include all 22 good correspondences after the initial least-squares fit. The final standard deviations about the camera parameters were as follows:

$$\begin{array}{ll} X: 0.1 \text{ ft} & \text{Heading: } 0.01^\circ \\ Y: 6.4 \text{ ft} & \text{Pitch: } 0.10^\circ \\ Z: 2.1 \text{ ft} & \text{Roll: } 0.12^\circ \end{array}$$

V. Concluding Comments

In this paper we have introduced a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and thus is ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors.

A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): Given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. Most of the results we presented concerning solution techniques and the geometry of the LDP problem are either new or not generally known. The current photogrammetric literature offers no analytic solution other than variants of least squares and the Church method for solving perspective-*n*-point problems. The Church method, which provides an iterative solution for the P3P problem [3, 11], is presented without any indication that more than one physically real solution is possible; there is certainly no indication that anyone realizes that physically real multiple solutions are possible for more than three control points in general position. (It should be noted that because the multiple solutions can be arbitrarily close together, even when an iterative technique is initialized to a value close to the correct solution there is no assurance that it will converge to the desired value.)

In the section on the LDP problem (and associated appendices) we have completely characterized the P3P problem and provided a closed-form solution. We have shown that multiple physically real solutions can exist for the P4P and P5P problems, but also demonstrated that a unique solution is assured when four of the control points reside on a common plane (solution techniques are provided for each of these cases). The issue of determining the maximum number of solutions possible for the P4P and P5P problems remains open, but we have shown that a unique solution exists for the P6P problem when the control points are in general position.

Appendix A. An Analytic Solution for the Perspective-3-Point Problem

The main body of this paper established that P3P problems can have as many as four solutions. In this appendix a closed form expression for obtaining these solutions is derived. Our approach involves three steps: (1) Find the lengths of the legs of the ("perspective") tetrahedron given the base (defined by the three control points) and the face angles of the opposing trihedral angle (the three angles to the three pairs of control points as viewed from the CP); (2) Locate the CP with respect to the 3-D reference frame in which the control points were originally specified; (3) Compute the orientation of the image plane with respect to the reference frame.

1. A Solution for the Perspective Tetrahedron (see Figure 4)

Given the lengths of the three sides of the base of a tetrahedron (Rab , Rac , Rbc), and given the corresponding face angles of the opposing trihedral angle (θ_{ab} , θ_{ac} , θ_{bc}), find the lengths of the three remaining sides of the tetrahedron (a , b , c).

A solution to the above problem can be obtained by simultaneously solving the system of equations:

$$(Rab)^2 = a^2 + b^2 - 2*a*b*\cos(\theta_{ab}); \quad (A1)$$

$$(Rac)^2 = a^2 + c^2 - 2*a*c*\cos(\theta_{ac}), \quad (A2)$$

$$(Rbc)^2 = b^2 + c^2 - 2*b*c*\cos(\theta_{bc}). \quad (A3)$$

We now proceed as follows:

$$\text{Let } b = x*a \text{ and } c = y*a, \quad (A4)$$

$$(Rac)^2 = a^2 + (y^2)*a^2 - 2*(a^2)*y*\cos(\theta_{ac}), \quad (A5)$$

$$(Rab)^2 = a^2 + (x^2)*a^2 - 2*(a^2)*x*\cos(\theta_{ab}), \quad (A6)$$

$$(Rbc)^2 = (x^2)*a^2 + (y^2)*a^2 - 2*(a^2)*x*y*\cos(\theta_{bc}). \quad (A7)$$

From Eqs. (A5) and (A7)

$$[(Rbc)^2]*[1 + (y^2) - 2*y*\cos(\theta_{ac})] = [(Rac)^2]*[(x^2) + (y^2) - 2*x*y*\cos(\theta_{bc})]. \quad (A8)$$

From Eqs. (A6) and (A7)

$$[(Rbc)^2]*[1 + (x^2) - 2*x*\cos(\theta_{ab})] = [(Rab)^2]*[(x^2) + (y^2) - 2*x*y*\cos(\theta_{bc})], \quad (A9)$$

$$\text{Let } \frac{(Rbc)^2}{(Rac)^2} = K1 \quad \text{and} \quad \frac{(Rbc)^2}{(Rab)^2} = K2. \quad (A10)$$

From Eqs. (A8) and (A9)

$$0 = (y^2)*[1 - K1] + 2*y*[K1*\cos(\theta_{ac}) - x*\cos(\theta_{bc})] + [(x^2) - K1]. \quad (A11)$$

From Eqs. (A9) and (A10)

$$0 = (y^2) + 2*y*[-x*\cos(\theta_{bc})] + [(x^2)*(1 - K2) + 2*x*K2*\cos(\theta_{ab}) - K2]. \quad (A12)$$

Equations (A11) and (A12) have the form:

$$0 = m^*(y^2) + p^*y + q, \quad (A13)$$

$$0 = m'^*(y^2) + p'^*y + q'. \quad (A14)$$

Multiplying Eqs. (A13) and (A14) by m' and m , respectively, and subtracting,

$$0 = [p^*m' - p'^*m]*y + [m'^*q - m^*q']. \quad (A15)$$

Multiplying Eqs. (A13) and (A14) by q' and q , respectively, subtracting, and dividing by y ,

$$0 = [m'^*q - m^*q']*(y^2) + [p'^*q - p^*q']*y, \quad (A16)$$

$$0 = [m'^*q - m^*q']*y + [p'^*q - p^*q']. \quad (A16)$$

Assuming $m'^*q \neq m^*q'$,

$$[(x^2) - K1] \neq [(x^2)*(1 - K1)*(1 - K2) + 2*x*K2*(1 - K1)*\cos(\theta_{ab}) - (1 - K1)*K2],$$

then Eqs. (A15) and (A16) are equivalent to Eqs. (A13) and (A14). We now multiply Eqs. (A15) by $(m'^*q - m^*q')$, and Eq. (A16) by $(p^*m' - p'^*m)$, and subtract to obtain

$$0 = (m'^*q - m^*q')^2 - [p^*m' - p'^*m]*[p'^*q - p^*q']. \quad (A17)$$

Expanding Eq. (A17) and grouping terms we obtain a biquadratic (quartic) polynomial in x :

$$0 = G4*(x^4) + G3*(x^3) + G2*(x^2) + G1*(x) + GO, \quad (A18)$$

where

$$G4 = (K1*K2 - K1 - K2)^2 - 4*K1*K2*\cos(\theta_{bc})^2, \quad (A19)$$

$$G3 = 4*[K1*K2 - K1 - K2]*K2*(1 - K1)*\cos(\theta_{ab}) + 4*K1*\cos(\theta_{bc})*[(K1*K2 + K2 - K1)*\cos(\theta_{ac}) + 2*K2*\cos(\theta_{ab})*\cos(\theta_{bc})], \quad (A20)$$

$$G2 = [2*K2*(1 - K1)*\cos(\theta_{ab})]^2 + 2*[K1*K2 + K1 - K2]*[K1*K2 - K1 - K2] + 4*K1*[(K1 - K2)*\cos(\theta_{bc})^2 + (1 - K2)*K1*(\cos(\theta_{ac})^2) - 2*K2*(1 + K1)*\cos(\theta_{ab})*\cos(\theta_{ac})*\cos(\theta_{bc})], \quad (A21)$$

$$G1 = 4*(K1*K2 + K1 - K2)*K2*(1 - K1)*\cos(\theta_{ab}) + 4*K1*[(K1*K2 - K1$$

$$+ K2)^* \cos(\theta ac)^* \cos(\theta bc) \\ + 2^* K1^* K2^* \cos(\theta ab)^* (\cos(\theta ac)^2)],$$

$$G0 = (K1^* K2 + K1 - K2)^2 - 4^* (K1^2)^* K2^* (\cos(\theta ac)^2), \quad (A23)$$

Roots of Eq. (A18) can be found in closed form [5], or by iterative techniques [4]. For each positive real root of Eq. (A18), we determine a single positive real value for each of the sides a and b . From Eq. (A6) we have

$$a = \frac{Rab}{\sqrt{(x^2) - 2^* x^* \cos(\theta ab) + 1}}, \quad (A24)$$

and from Eq. (A4) we obtain

$$b = a^* x. \quad (A25)$$

If $m'^* q \neq m^* q'$, then from Eq. (A16) we have

$$y = \frac{p'^* q - p^* q'}{m^* q' - m'^* q}. \quad (A26)$$

If $m'^* q = m^* q'$, then Eq. (A26) is undefined and we obtain two values of y from Eq. (A5):

$$y = \cos(\theta ac) \\ \pm \sqrt{(\cos(\theta ac))^2 + \frac{(Rac)^2 - (a^2)}{(a^2)}}. \quad (A27)$$

For each real positive value of y , we obtain a value of c from Eq. (A4):

$$c = y^* a \quad (A28)$$

When values of y are obtained from Eq. (A5) rather than Eq. (A26), the resulting solutions can be invalid; they must be shown to satisfy Eq. (A3) before they are accepted.

It should be noted that because each root of Eq. (A18) can conceivably lead to two distinct solutions, the existence of the biquadratic does not by itself imply a maximum of four solutions to the P3P problem; some additional argument, such as the one given in the main body of this paper, is necessary to establish the upper bound of four solutions.

2. Example

For the perspective tetrahedron shown in Figure 5, we have the following parameters:

$$Rab = Rac = Rbc = 2^* \sqrt{3}, \\ \cos(\theta ab) = \cos(\theta ac) = \cos(\theta bc) \\ = \frac{(a^2) + (b^2) - (Rab)^2}{2^* a^* b} = \frac{20}{32}.$$

Substituting these values into Eqs. (A19) through (A23), we obtain the coefficients of the biquadratic defined in Eq. (A18):

$$[-0.5625, 3.515625, -5.90625, 3.515625, -0.5625]$$

The roots of the above equation are

$$[1, 1, 4, 0.25]$$

For each root

Root	a	b	y	c
1	4	4	1	4
1	4	4	0.25	1
4	1	4	4	4
0.25	4	1	1	4

3. An Iterative Solution for the Perspective Tetrahedron (see Figure 8)

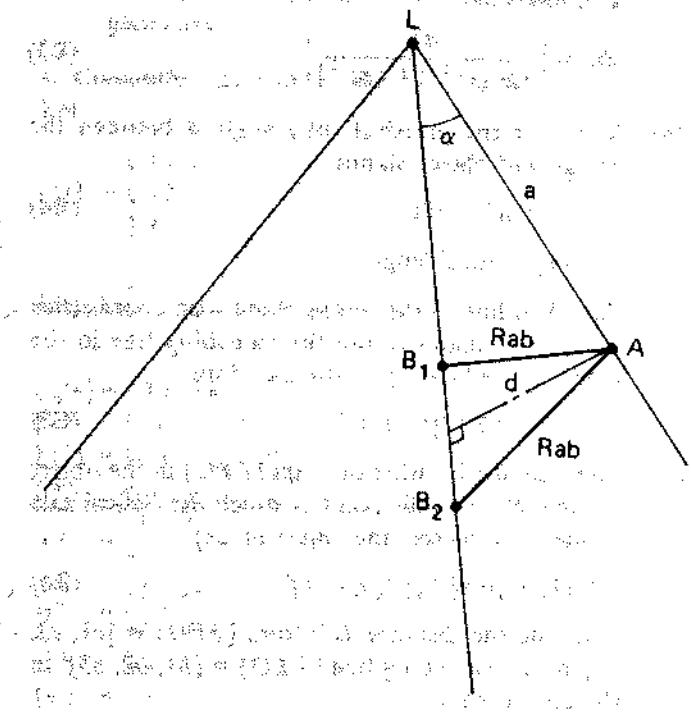
A simple way to locate solutions to P3P problems, which is sometimes an adequate substitute for the more involved procedure described in the preceding subsection, is to slide one vertex of the control-point triangle down its leg of the tetrahedron and look for positions of the triangle in which the other two vertices lie on their respective legs. If vertex A is at a distance a from L (L is the center of perspective), the lengths of the sides Rab and Rac restrict the triangle to four possible positions. Given the angle between legs LA and LB , compute the distance of point A from the line LB and then compute points $B1$ and $B2$ on LB that are the proper distance from A to insert a line segment of length Rab . Similarly, we compute at most two locations for C on its leg. Thus, given a position for A we have found at most four positions for a triangle that has one side of length Rab and one of length Rac . The lengths of the third sides (BC) of the four triangles vary nonlinearly as point A is moved down its leg. Solutions to the problem can be obtained by iteratively repositioning A to imply a third side of the required length.

4. Computing the 3-D Location of the Center of Perspective (see Figure 9)

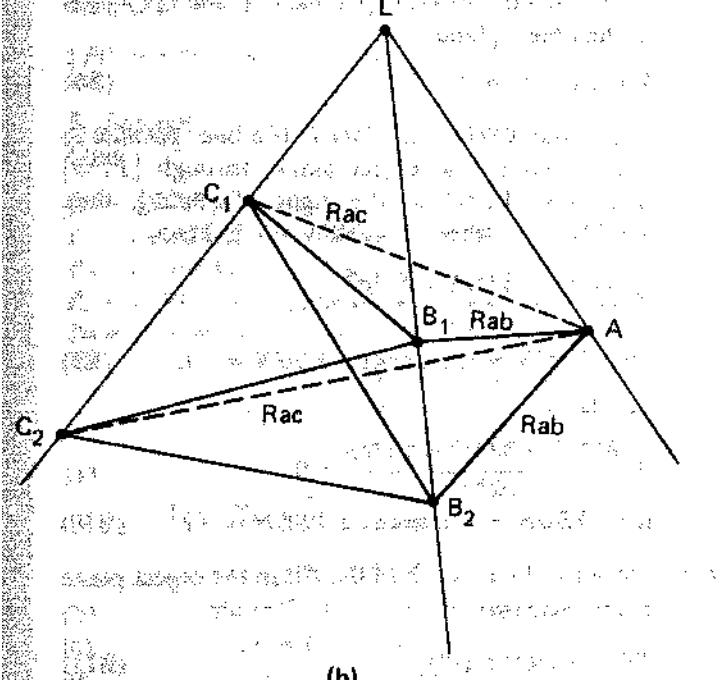
Given the three-dimensional locations of the three control points of a perspective tetrahedron, and the lengths of the three legs, the 3-D location of the center of perspective can be computed as follows:

- (1) Construct a plane $P1$ that is normal to AB and passes through the center of perspective, L . This plane can be constructed without knowing the position of L , which is what we are trying to compute. Consider the face of the tetrahedron that contains vertices A , B , and L . Knowing the lengths of sides LA , LB and AB , we can use the law of cosines to find the angle $\angle LAB$, and then the projection Q_A of LA on AB . (Note that angle $\angle LQA$ is a right angle, and the point Q is that point on line AB that is closest to L). Construct a plane normal to AB passing through Q ; this plane also passes through L .
- (2) Similarly construct a plane $P2$ that is normal to AC and passes through L .
- (3) Construct the plane $P3$ defined by the three points A , B , and C .
- (4) Intersect planes $P1$, $P2$, and $P3$. By construction, the point of intersection R is the point on $P3$ that is closest to L .
- (5) Compute the length of the line AR and use that in conjunction with the length of LA to compute the length of the line RL , which is the distance of L from the plane $P3$.
- (6) Compute the cross product of vectors AB and AC to form a vector perpendicular to $P3$. Then scale that vector by the length of RL and add it to R to get the 3-D location of the center of perspective L .

Fig. 8. Geometry for an Iterative Solution to the P3P Problem.



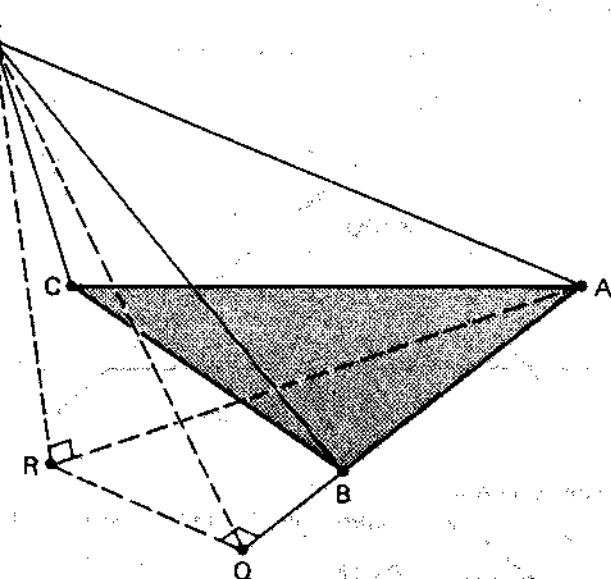
(a)



(b)

If the focal length of the camera and the principal point in the image plane are known, it is possible to compute the orientation of the image plane with respect to the world coordinate system; that is, the location of the origin and the orientation of the image plane coordinate system with respect to the 3-D reference frame. This can be done as follows:

Fig. 9 Computing the 3-D Location of the Center of Perspective (L).



- (1) Compute the 3-D reference frame coordinates of the center of perspective (as described above).
- (2) Compute the 3-D coordinates of the image locations of the three control points: since we know the 3-D coordinates of the CP and the control points, we can compute the 3-D coordinates of the three rays between the CP and the control points. Knowing the focal length of the imaging system, we can compute, and subtract from each ray, the distance from the CP to the image plane along the ray.
- (3) Compute the equation of the plane containing the image using the three points found in step (2). The normal to this plane, passing through the CP, gives us the origin of the image plane coordinate system (i.e., the 3-D location of the principal point), and the Z axis of this system.
- (4) The orientation of the image plane about the Z axis can be obtained by computing the 3-D coordinates of a vector from the principal point to any one of the points found in step (2).

Appendix B. An Analytic Solution for the Perspective-4-Point Problem (with all control points lying in a common plane)

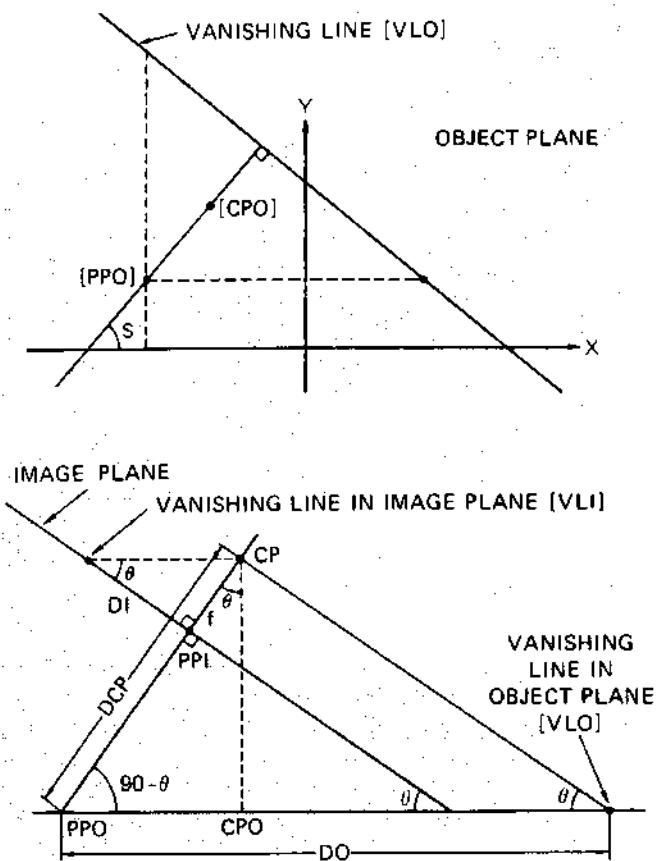
In this appendix an analytic technique is presented for obtaining a unique solution to the P4P problem when the four given control points all lie in a common plane.

1. Problem Statement (see Figure 10)

GIVEN: A correspondence between four points lying in a plane in 3-D space (called the object plane), and four points lying in a distinct plane (called the image plane); and given the distance between the center of perspective and the image plane (i.e., the focal length of the imaging system); and also given the principal point in the image plane (i.e., the location, in image plane coordinates, of the point at which the optical axis of the lens pierces the image plane).

FIND: the 3-D location of the center of perspective relative to the coordinate system of the object plane.

Fig. 10. Geometry of the P4P Problem (With all Control Points Lying in a Common Plane).



2. Notation

- * Let the four given image points be labeled $\{P_i\}$, and the four corresponding object points $\{Q_i\}$.
- * We will assume that the 2-D image plane coordinate system has its origin at the principal point (PPI).
- * We will assume that the object plane has the equation $Z = 0$ in the reference coordinate system. Standard techniques are available to transform from this coordinate system into a ground reference frame (e.g., see [6] or [9]).
- * Homogeneous coordinates will be assumed [12].
- * Primed symbols represent transposed structures.

3. Solution Procedure

- (a) Compute the 3×3 collineation matrix T which maps points from object plane to image plane (a procedure for computing T is given later):

$$[P_i] = [T]^*[Q_i],$$

where

$$(B1)$$

$$[P_i] = [k_i^* x_i, k_i^* y_i, k_i]^*,$$

$$[Q_i] = [X_i, Y_i, 1]^*.$$

- (b) The ideal line in the object plane, with coordinates $[0, 0, 1]'$ is mapped into the vanishing line in the image plane $[VLI]$ by the transformation:

$$[VLI] = [\text{inv}[T]]^*[0, 0, 1]'. \quad (B2)$$

- (c) Determine the distance DI from the origin of the image plane (PPI) to the vanishing line $[VLI]$ =

$$[a_1, a_2, a_3]':$$

$$DI = \left| \frac{a_3}{\sqrt{(a_1)^2 + (a_2)^2}} \right| \quad (B3)$$

- (d) Solve for the dihedral (tilt) angle θ between the image and object planes:

$$\theta = \arctan(f/DI), \quad (B4)$$

where f = focal length.

- (e) The ideal line in the image plane with coordinates $[0, 0, 1]'$ is mapped into the vanishing line in the object plane $[VLO]$ by the transform

$$[VLO] = [T]^*[0, 0, 1]'. \quad (B5)$$

- (f) Compute the location of point $[PPO]$ in the object plane ($[PPO]$ is the point at which the optical axis of the lens pierces the object plane):

$$[PPO] = [\text{inv}[T]]^*[0, 0, 1]'. \quad (B6)$$

- (g) Compute the distance DO from $[PPO] = [c_1, c_2, c_3]'$ to the vanishing line $[VLO] = [b_1, b_2, b_3]'$ in the object plane:

$$DO = \left| \frac{b_1*c_1 + b_2*c_2 + b_3*c_3}{\sqrt{b_1^2 + b_2^2}} \right|. \quad (B7)$$

- (h) Solve for the "pan" angle $\$$ as the angle between the normal to $[VLO] = [b_1, b_2, b_3]'$ and the X axis in the object plane:

$$\$ = \arctan(-b_2/b_1). \quad (B8)$$

- (i) Determine $XSGN$ and $YSGN$: If a line (parallel to the X axis in the object plane) through $[PPO]$ intersects $[VLO]$ to the right of $[PPO]$, then $XSGN = 1$. Otherwise $XSGN = -1$. Thus,

$$\text{if } \frac{b_1*c_1 + b_2*c_2 + b_3*c_3}{b_1*c_3} < 0,$$

then $XSGN = 1$, otherwise $XSGN = -1$. $(B9)$

Similarly,

$$\text{if } \frac{b_1*c_1 + b_2*c_2 + b_3*c_3}{b_2*c_3} < 0$$

then $YSGN = 1$, otherwise $YSGN = -1$. $(B10)$

- (j) Solve for the location of the CP in the object plane coordinate system:

$$DCP = DO * \sin(\theta) \quad (B11)$$

$$XCP = XSGN * \text{abs}[DCP * \sin(\theta) * \cos(\$)] + c_1/c_3 \quad (B12)$$

$$YCP = YSGN * \text{abs}[DCP * \sin(\theta) * \sin(\$)] + c_2/c_3 \quad (B13)$$

$$ZCP = DCP * \cos(\theta) \quad (B14)$$

Note: If $[VLI]$, as determined in (b), has the coordinates $[0, 0, k]$, then the image and object planes are parallel ($\theta = 0$). Rather than continuing with the above procedure, we now solve for the desired

information using similar triangles and Euclidean geometry.

4. Computing the Collineation Matrix T

Let

$$[Q] = \begin{vmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \\ X_3 & Y_3 & 1 \end{vmatrix} = [[Q_1]', [Q_2]', [Q_3]',]$$

$$[P] = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = [[P_1]', [P_2]', [P_3]',]$$

$$[Q_4] = [X_4, Y_4, 1]',$$

$$[P_4] = [x_4, y_4, 1]',$$

$$[V] = [\text{inv}[P]]^* [P_4] = [v_1, v_2, v_3]',$$

$$[R] = [\text{inv}[Q]]^* [Q_4] = [r_1, r_2, r_3]',$$

$$w_1 = \frac{v_1 * r_3}{r_1 * v_3},$$

$$w_2 = \frac{v_2 * r_3}{r_2 * v_3},$$

$$[w] = \begin{vmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & 1 \end{vmatrix}.$$

Then,

$$[T]' = [\text{inv}[Q]]^* [W]^* [P]$$

such that

$$[P_i] = k_i^* [x_i, y_i, 1] = [T]^* [Q_i].$$

5. Example

Given:

$$f = 0.3048 \text{ m (12 in.)}$$

$$\begin{aligned} P_1 &= (-0.071263, 0.029665) & Q_1 &= (-30, 80) \\ P_2 &= (-0.053033, -0.006379) & Q_2 &= (-100, -20) \\ P_3 &= (-0.014063, 0.061579) & Q_3 &= (140, 50) \\ P_4 &= (0.080120, -0.030305) & Q_4 &= (-40, -240) \end{aligned}$$

$$[T]' = \begin{vmatrix} 0.000212 & 0.000236 & 0.000925 \\ -0.000368 & 0.000137 & 0.000534 \\ -0.025404 & 0.021650 & 0.843879 \end{vmatrix}$$

(a)

$$[\text{inv}[T]]' = \begin{vmatrix} 1117.14 & -2038.86 & 0.0 \\ 3371.56 & 2302.22 & -5.14991 \\ -51.0636 & -120.442 & 1.31713 \end{vmatrix}$$

$$(b) [VLI] = [0, -5.14991, 1.31713]'$$

$$(c) DI = 0.255758$$

$$(d) \theta = 0.872665 \text{ rad (50°)}$$

$$(e) [VLO] = [0.000925, 0.000534, 0.843880]'$$

$$(f) [PPO] = [-51.0636, -120.442, 1.31713]'$$

$$(g) DO = 711.196$$

$$(h) \$ = -0.523599 \text{ rad (-30°)}$$

$$(i) XSGN = -1$$

$$(j) YSGN = -1$$

$$(k) DCP = 544.8081$$

$$XCP = -400.202$$

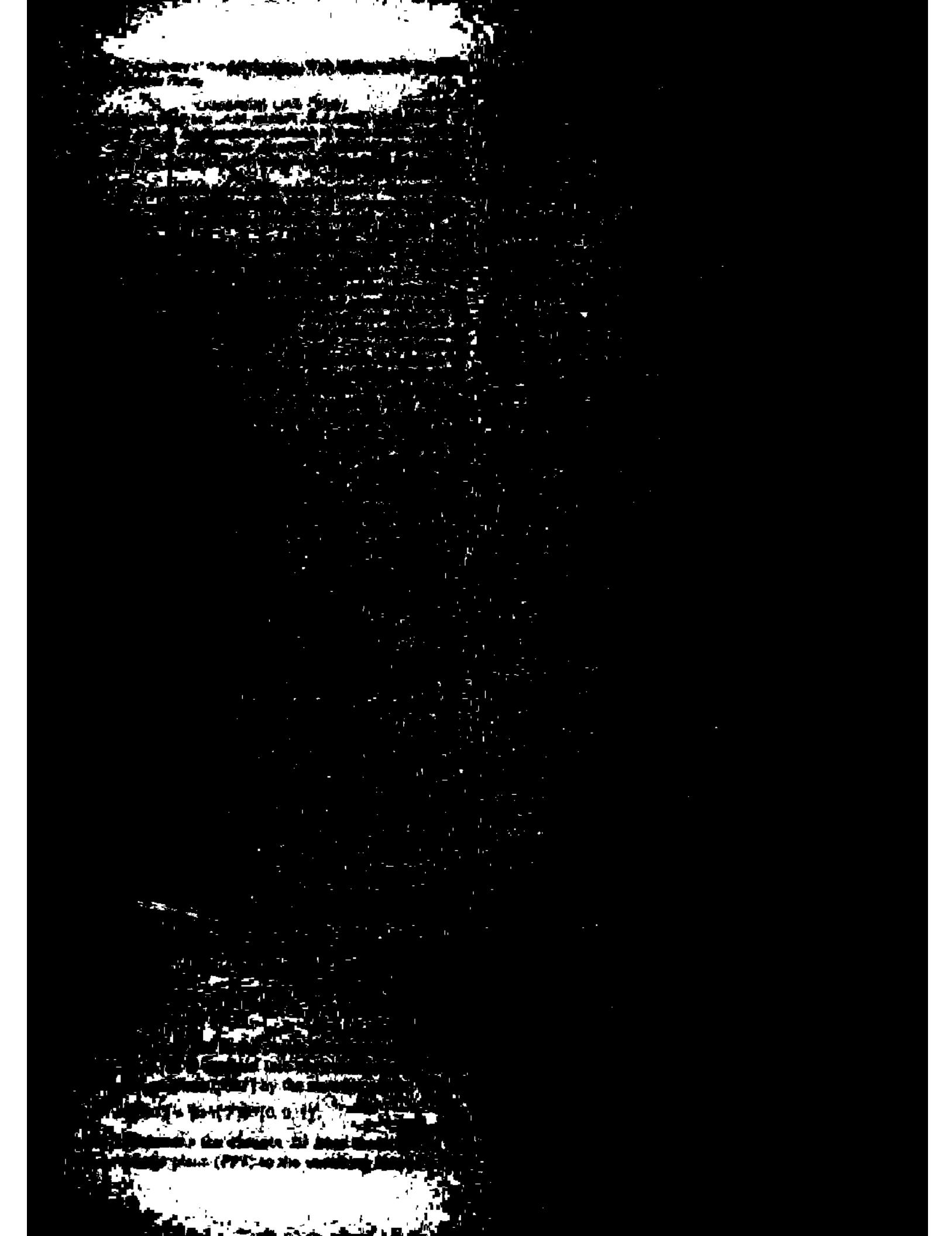
$$YCP = -300.117$$

$$ZCP = 350.196$$

Received 4/80; revised 1/81; accepted 1/81

References

1. Bolles, R.C., Quarn, L.H., Fischler, M.A., and Wolf, H.C. The SRI road expert: Image to database correspondence. In Proc. Image Understanding Workshop, Pittsburgh, Pennsylvania, Nov., 1978.
2. Chrystal, G. *Textbook of Algebra* (Vol 1). Chelsea, New York, New York 1964, p. 415.
3. Church, E. Revised geometry of the aerial photograph. *Bull. Aerial Photogrammetry*, 15, 1945, Syracuse University.
4. Conte, S.D. *Elementary Numerical Analysis*. McGraw Hill, New York, 1965.
5. Dehn, E. *Algebraic Equations*. Dover, New York, 1960.
6. Duda, R.O., and Hart, P.E. *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York, 1973.
7. Gennery, D.B. Least-squares stereo-camera calibration. Stanford Artificial Intelligence Project Internal Memo, Stanford, CA 1975.
8. Keller, M. and Tewinkel, G.C. Space resection in photogrammetry. ESSA Tech. Rept C&GS 32, 1966, U.S. Coast and Geodetic Survey.
9. Rogers, D.P. and Adams, J.A. *Mathematical Elements for Computer Graphics*. McGraw Hill, New York, 1976.
10. Sorensen, H.W. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum* (July 1970), 63-68.
11. Wolf, P.R. *Elements of Photogrammetry*. McGraw Hill, New York, 1974.
12. Wylie, C.R. Jr. *Introduction to Projective Geometry*. McGraw-Hill, New York, 1970.



Chapter 4: Constraint Exploitation and Shape Recovery

Mapping from a scene to an image is a many-to-one mapping; hence, in principle, information about a scene cannot be recovered unambiguously from its image. Information recovery is possible only if some assumptions are made about the objects in the scene and the relations among these objects. These assumptions are based on several kinds of knowledge. To recover information, vision systems exploit

- Knowledge about the image formation process;
- Domain-independent knowledge about the world; and
- Knowledge about the specific application domain.

Knowledge about the image formation process, such as the projection mechanism, helps in relating artifacts in the image to physical phenomena in the scene. Domain-independent knowledge helps in early vision because it involves general processes from which information recovery can begin. This type of knowledge comes in many different forms. One common example of domain-independent knowledge is surface coherence; many other forms of domain-independent knowledge are used in shape-from-shading, shape-from-stereo, and other similar intermediate-level operations. Knowledge about the specific application domain is usually the most explicit. It may be in the form of geometric models or feature models or in some other similar form.

Major issues in recovery of information in images include the representation and utilization of different types of knowledge in a vision system. The amount and quality of knowledge available, the knowledge representation methods, and the methodologies of applying knowledge in the system strongly influence the performance of a vision system. In fact, the extensibility, flexibility, and generality of a vision system are dependent on the blending of knowledge sources in the system. A system that makes explicit the representation and utilization of knowledge generally is easier to modify and maintain than a system that uses knowledge implicitly. Computer vision operations are usually categorized as low level, intermediate level, and high level. Low-level operations are mostly image related. Knowledge is used at this level implicitly and in minimal amounts. Good examples of low-level knowledge application are the use of similar intensity values for grouping points or the use of thresholds for obtaining binary images. Intermediate-level operations use image formation knowledge and knowledge about the environment, both usually in the form of constraints. In high-level operations, knowledge about the objects is represented explicitly.

A constraint represents a relationship between two or more entities in the "world" that is always satisfied. Constraints are the result of observations or assumptions. A major problem in computer vision is identifying suitable constraints that will help in recovering information without seriously constraining the applicability of the system. Many difficult problems become solvable once suitable constraints are discovered and techniques are developed to exploit them. On the other hand, the use of constraints that are valid only in a limited "world" will restrict the applicability of the system.

In this chapter, several variations of constraint-related problems will be discussed. The emphasis will be on applying constraints, not discovering them. The labeling problem will be examined first. Component labeling plays an important role in intermediate-level processes, after some initial segmentation. Methods for assigning labels to objects using directly available relational information and knowledge about valid relations expressed as constraints will be considered. Techniques for recovering three-dimensional shape from two-dimensional images — either by exploiting image cues such as shading, texture, and blur or by matching multiple images of the same scene — are also described in this chapter.

The labeling problem

The labeling problem involves a given set of objects and a set of labels that are to be assigned to the objects. The relationships between the objects in the image are known; based on knowledge about the domain, the conditions under which a certain set of labels may or may not be applied to a set of objects is also known. The problem, then, is to assign proper labels to the objects. In a general labeling problem, the following sets are given

- A set $O = O_1, O_2, \dots, O_n$ of objects;
- A set $L = L_1, L_2, \dots, L_m$ of labels;
- A set of relations R over the objects; and
- A set of constraints C specifying valid labels for related objects.

A solution to a labeling problem is a set (O_i, L_i) , where $O_i \in O$ and $L_i \in L$, which gives a label for each object. In most applications, an object should be assigned a unique label.

Although an exhaustive search for all possible object/label assignments satisfying all of the constraints would certainly result in a correct labeling of the objects, the combinatorial explosion in the search space makes this impractical. Several approaches that use computationally efficient heuristic techniques have been suggested for solving this problem.

A sequential approach. Initially, every label may be assigned to each object. The objective is to discard incorrect labels, using the relations and constraints given for the problem. Each object is considered, and any labels that are inconsistent with the labels of "related" objects are eliminated. It may be necessary to repeatedly examine the list of objects until either (1) each object has been assigned a unique label that is consistent with other objects or (2) no further elimination of labels is possible. The following example illustrates this approach.

An image has been segmented into several regions using a domain-independent method. (See Figure 4.1.¹¹⁰) The identification of these regions is now required. If it is given that the image represents an office scene, knowledge about the nature of objects in this domain may be used. To introduce this knowledge, a set of possible labels for the regions and a set of constraints on the regions will be required. The labels for objects in this domain are

$$L = \{\text{Door}, \text{Wall}, \text{Floor}, \text{Picture}, \text{Baseboard}, \text{Doorknob}\}$$

Based on observation, the following constraints should be satisfied by the objects in an image:

- Within: (Picture, Wall), (Doorknob, Door);
- Above: (Wall, Baseboard), (Baseboard, Floor), (Door, Floor);
- Beside: (Wall, Door), (Baseboard, Door); and
- Small: (Doorknob), (Picture).

The objects to be labeled are regions in the image, which have been numbered as

$$O = \{O_1, O_2, O_3, O_4, O_5, O_6\}$$

Spatial relationships between regions are determined from the segmented image. These relationships are as follows:

- Within: (O_2, O_1), (O_4, O_3);
- Above: (O_1, O_5), (O_5, O_6), (O_3, O_6);
- Beside: (O_1, O_3), (O_5, O_3); and
- Small: (O_4), (O_2).

Initially, each region is assigned all labels. The relations obtained from the image and the known constraints are then applied sequentially to eliminate incorrect labels. The process is iterated, if necessary. In this example, that each region is eventually assigned a unique label is easily verified.

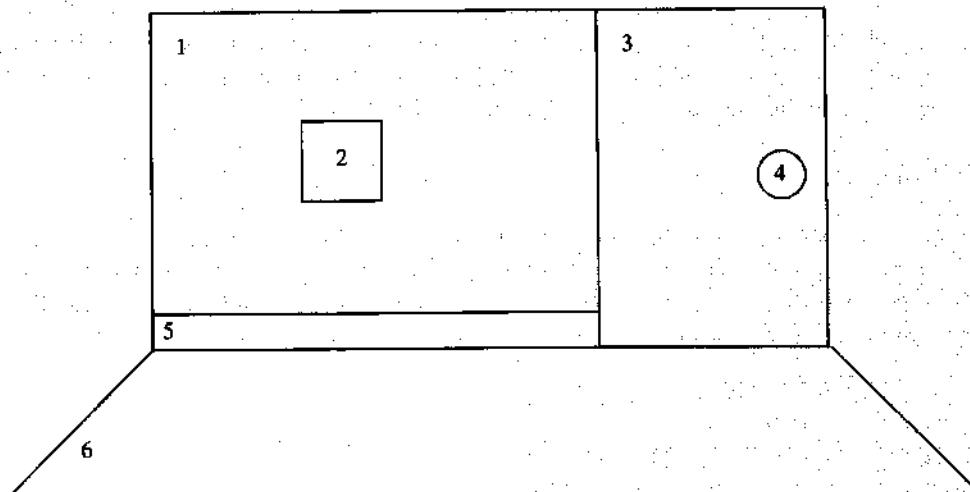


Figure 4.1: A scene with six regions (from Barrow and Tennenbaum).¹¹⁰

The preceding example illustrates the application of known properties and relationships in solving a labeling problem. Knowledge in the form of constraints can be used to resolve local ambiguities or uncertainties and can lead to a solution of the global problem. Although the filtering at each step uses local relationships between at most two objects, the effect is propagated to other objects.

Several problems exist with the approach just illustrated. The first is that the outcome of such a sequential process is dependent on the starting point. Second, the convergence of the approach is not well understood. A third problem is posed because determination of a unique labeling for a given set of constraints and relationships generally cannot be guaranteed. Since the process is iterative, a stopping point should be well defined. If a unique labeling can be achieved, then the stopping criterion is well defined. If not, a general rule may be to stop after going through a predetermined number of iterations over all objects; however, several objects may then not be assigned unique labels. An object may be assigned multiple labels if (1) not enough iterations were allowed, (2) the labeling problem is unsolvable based on the information, or (3) the given relations are "impossible" in the "world" that obeys the given constraints.

A parallel approach. In many cases, the labeling problem can be represented as a graph, with each node representing an object or entity that should be assigned a label. The arcs are labeled to represent different relationships between objects. Since the constraints specify relationships between only a few objects, the labeling process may be applied in parallel.

In the parallel approach, a processor is assumed at each node. Each processor "knows" the label of its node and of all nodes that are connected to it. It also knows the set of constraints C and all relationships involving its node. In the first iteration, the possible label set P_i of node i is L for all i . In other words, all nodes initially are assigned all possible labels. The labeling process will iteratively eliminate invalid labels from P_i . At any stage, labels are discarded considering only the node, its relationships, and the constraints; hence, each processor has sufficient information to refine its label set P_i . Thus, all processors can work synchronously. In each cycle, a processor uses the current information to refine its P_i by discarding invalid labels. However, each iteration propagates the effect through its neighbor or related nodes to other nodes that are not directly related; therefore, the circle of influence of a node increases with each iteration.

In the just-described formulation of the consistent-labeling problem, the assumption was that the process started with no knowledge about the objects. Under this assumption, every label in L was initially assigned to each object. In most applications, some knowledge about the objects is already available at the start of the labeling process. Segmentation, or some other process that takes place before labeling, often gives information that can be used to refine the initial set P_i for a node. A labeling process, known as "relaxation labeling," may be used to further refine these sets.

Relaxation processes

Although relaxation labeling¹¹¹ is another solution to the labeling problem, it differs from the preceding formulation in that it begins with an initial set of confidence values assigned to labels. Based on some unary relations, a label set can be assigned to an object. A confidence value is assigned to each possible label. As in subjective probability, this confidence value indicates a belief that the label is correct based on evidence in the image. Thus, for each element $L_k \in L$, a nonnegative probability p_k represents the confidence that the label L_k is the correct label for object O .

The task of the relaxation-labeling process¹¹¹ is now to use the constraints to refine the confidence value for each label. When the confidence in one label for a node is significantly stronger than the confidence in the other labels, that label can be assigned to the node. The confidence value will be influenced by the confidence values in the labels of the connected nodes. The process may terminate when either each node has been assigned a unique label or the confidence values achieve a steady state. Relaxation labeling can be considered to be a form of constraint propagation in which nodes are processed in parallel.

In relaxation labeling, the constraints are specified in terms of compatibility functions. Suppose that objects O_i and O_j are related by R_{ij} and that, under this relationship, labels L_i for object O_i and L_j for object O_j are highly likely to occur. The knowledge about the likelihood of these labels can be expressed in terms of a function that will increase the confidence in these labels for the objects under consideration. In such a situation, the presence of L_i at O_i encourages the assignment of L_j to O_j . In addition, the incompatibility of certain labels can be used to discourage labels by decreasing their confidence values.

Discrete relaxation labeling. In discrete relaxation labeling, label confidence values are allowed only two values: zero and one. The strategy used in constraint propagation is as follows:

- (1) Keep the label $\lambda = L_i$ at node O_i only if, for every neighbor O_j , a label $\lambda' \in L$ exists that is compatible with λ .
- (2) Repeat until no label is discarded.

Because of the parallel nature of this labeling, the number of iterations required for termination is determined by the maximum distance the information should travel to disambiguate the worst node. The local perspective of relaxation labeling may not be able to disambiguate some global inconsistencies. Thus, even after a large number of iterations, the process may not terminate. The convergence properties of a relaxation-labeling process generally cannot be predicted from the formulation of the problem.

Probabilistic relaxation labeling. As just discussed, confidence values can be assigned to the possible labels for an object. In probabilistic relaxation labeling, the probabilities of the labels are updated during each iteration instead of the labels being discarded. Compatibility functions replace constraint relations for updating the probabilities of labels. Both positive and negative influences can be propagated using compatibility functions. Determining the best label at a node involves considering both competing and cooperating forces. The confidence in a label is determined based on the sum of these competing and cooperating forces.

Like constraints, compatibility functions may be obtained by analyzing a problem. One way to formulate the relaxation-labeling problem is to use probability theory. The labels assigned to an object should satisfy probabilistic axioms. Thus, the probability of the label L_k for the object O_i must satisfy the following:

$$0 \leq p_i(L_k) \leq 1$$

and $\sum_{k=1}^m p_i(L_k) = 1$ (4.1)

To find suitable compatibility functions, all possible situations may be analyzed to find conditional probabilities $p_{ij}(L_k | L_j)$ such that object O_j has label L_j , given that its neighboring object O_i has label L_i . The probability of a label at a node can then be updated using

$$p_i(L_k) = \sum_j c_{ij} \sum_{l=1}^m p_{ij}(L_k | L_l) p_j(L_l)$$

where $\sum_j c_{ij} = 1$ (4.2)

and where c_{ij} represents the compatibility of node j with node i . Once appropriate values of c_{ij} have been selected, such as assigning larger values of C_{ij} to objects which are closer to O_i and smaller values to those which are farther, and the initial probabilities for these labels have been obtained, this probability equation can be used to update the probabilities of the labels. When the probabilities for labels converge, the process stops.

The two principal limitations to the method just discussed are now given. First, the probabilities converge to a solution that depends on the compatibility functions and the conditional probabilities obtained from an analysis of the application domain; initial labels assigned using image analysis are ignored. Second, incompatible assignment of labels is not properly propagated in updated probabilities.¹¹¹ An improved updating scheme uses compatibility functions that have a range of [-1,1]. The rule for updating the probability during the iteration step $t+1$ is then given by

$$P_i^{t+1}(L_k) = \frac{P_i^t(L_k) \cdot (1 + q_i^t(L_k))}{\sum_{l=1}^m P_i^t(L_l) \cdot (1 + q_i^t(L_l))}$$

where $q_i^t(L_k) = \sum_j c_{ij} \sum_{L_l \in L} r_{ij}(L_k, L_l) P_j^t(L_l)$ (4.3)

and where c_{ij} values are the coefficients analogous to those for the linear model and $r_{ij}(L_k, L_l)$ is the correlation between the following events: that O_i has label L_k and that O_j has label L_l . Thus, $P_i(t+1)$ is a function of both the previous estimate and the contributions from the probability distributions on the neighboring label sets. In many cases, the compatibility and the weights may be combined into one term. The updating equation just given combines the effects of all of the neighbors of an object. This relaxation method has been applied to many applications.

Although the process in the method just described is local, the sphere of influence for each node grows with the iteration number. Thus, this process allows even weak evidence to make an appropriate contribution to the final result by propagating throughout an image. The local nature is very attractive for real-time implementation using special architecture. A major problem in the implementation of relaxation labeling has been convergence characteristics. In most cases, no definite termination point exists. Heuristics stop either after there are no changes in labels or after a certain number of iterations are commonly used.

Understanding block-world scenes

A system that analyzes images of block-world scenes can demonstrate the efficacy of constraint propagation in achieving global consistency using only local information. If line segments that have been detected in a scene are given, constraint propagation can be used to determine the number and nature of the objects in the scene. By considering simple block-world scenes, the following discussion examines a few key ideas related to discovering and using constraints.

Suppose that a scene contains only polyhedral objects and, furthermore, that the scene contains only objects that have trihedral junctions. (A trihedral junction is a vertex of an object where a maximum of three planes meet.) To simplify even further, we will consider only scenes without shadows. The task of the system is to understand a block-world image; that is, to determine the number of three-dimensional objects (blocks) and the relationships between the objects in the scene. The scene-understanding problem can be posed as a labeling problem. By assigning proper labels to the lines in an image, objects can be separated from each other and relationships between objects can be determined.

For a known polyhedral object, the edges can be classified into a small number of classes. Labels can be assigned to edges depending on the nature of the edge. For trihedral objects, the labels will be one of the following:

- Boundary lines are assigned a directional label > or <. The direction depends on the direction in which the boundary should be traveled in order to keep the object always on the right.
- Convex edges are assigned a + label. These edges indicate where the surface normal is toward the viewer.
- Concave edges are assigned a - label. These edges indicate where the surface normal is away from the viewer.

Figure 4.2 shows an object to which labels have been correctly assigned. For a known object, labels can be easily assigned. For an unknown object, the nature of the object can be roughly determined by assigning labels, which may be helpful in analyzing the scene. The problem of assigning labels to lines should be approached by first analyzing the physical world to determine what types of junctions are formed by different types of lines. The first step, then, is to determine constraints of the physical world that will restrict the set of all possible junctions formed by different types of edges to only physically valid junctions. Analyzing the physical world to obtain constraints is a key idea.

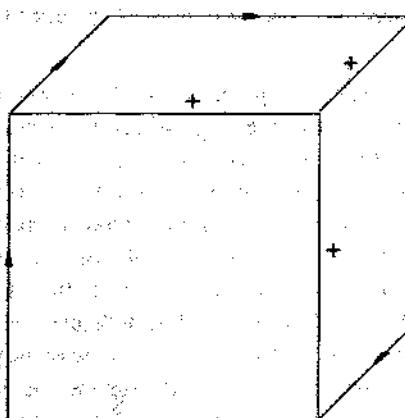


Figure 4.2: Assignment of labels to edges

For the simple case of trihedral objects, the junctions formed by the vertices must be one of the following types: ARROW, L, FORK, or T. In a more general situation, where the assumptions of trihedral objects and shadowless images are relaxed, a similar analysis will result in more applicable line labels and many more types of junctions.

Since only four labels ($>$, $<$, $+$, and $-$) can be considered for each line, only a finite number of possible ways exist to label a junction. A simple analysis shows that a FORK, an ARROW, and a T can each be labeled in 4^3 different ways and that an L can be labeled in 4^2 different ways. Thus, the total number of possible junction combinations is 208. An analysis of the physical world under the given restrictions shows that not all of these junctions actually occur. Considering all configurations of trihedral junctions shows that only 18 of the 208 possible junction combinations are really possible. Clearly, many labels for a junction can be eliminated immediately since they are impossible in the real world. The set of possible junction types in the shadowless trihedral-blocks world is given in Figure 4.3. This set represents the constraints imposed by the domain and can be used to assign proper labels to lines in a scene. An examination of the set of possible junctions provides more clues that may be helpful in scene analysis, as follows:

- The $>$ and $<$ labels appear only on borders.
- Only one ARROW junction exists that has $>$ labels on its barbs; the shaft of this arrow has a $+$ label.
- Only one FORK junction with any $+$ label exists; all of its lines have $+$ labels.

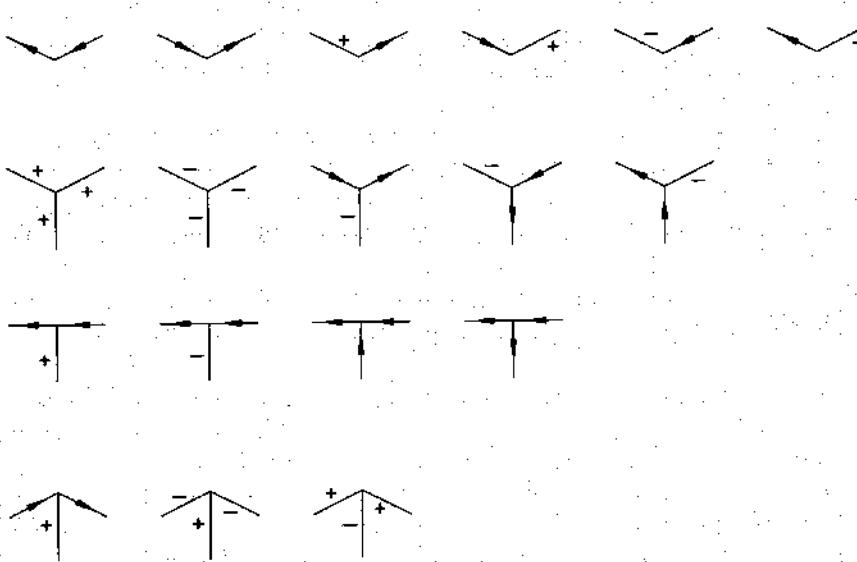


Figure 4.3: The set of possible junctions

In scene analysis, all lines must be found by using a segmentation scheme. Segmentation should give information about all of the lines in a scene and the junctions formed by these lines. The labeling process can then start by the assignment of $>$ and $<$ labels to all boundary lines. After the boundary lines have been labeled, arrows with $>$ barbs are found, and $+$ is assigned to their shafts. Then, any FORK with a $+$ label on any of its lines can have a $+$ assigned to all of its lines. If all of the lines in the scene are not labeled after the completion of these steps, then each junction that has at least one unlabeled line is examined. Considering the type of junction and the labels of the lines that have been assigned, all possible labels are assigned to the remaining unlabeled lines. Since a line is part of two junctions, its set of labels will be influenced from two sides. Since only one correct label exists for any line, the label eventually assigned is the one that is assigned by both junctions. This step can be iterated. In each iteration, some labels can be removed from the set of labels assigned to a line. In the case of the physically realizable trihedral-blocks world, this propagation process will eventually terminate, resulting in a labeling for a scene.

If, for some line, no unique label can be assigned, then the scene violates the assumptions about the world. Such a scene contains "impossible" objects. An impossible object, in this context, is one that cannot be realized using trihedral blocks. This labeling method demonstrates the following two important concepts:

- Local constraints can be used to derive a global interpretation of an image. The constraints are applied locally at each junction, but the effects of the constraints propagate through the image to yield a global interpretation of the scene.
- Constraints should be determined through an analysis of the physical world. Constraints may be available already in some cases; however, in many applications, suitable constraints must be discovered by an analysis of the problem domain.

Shape recovery

In Chapter 1, we discuss briefly the relationships among properties of objects (e.g., orientation and reflectance) in the three-dimensional scene and their two-dimensional image intensities. In this section, we discuss techniques for exploiting image cues — such as shading, texture, and blurring — to recover three-dimensional shape from two-dimensional images. We also describe briefly binocular- and photometric-stereo methods for recovering three-dimensional shape from multiple two-dimensional images. Shape recovery from motion is described in Chapter 6.

Shape from shading. Chapter 1 explains that the image intensity at a point is a function of the surface orientation. This dependence is captured in reflectance maps. For fixed illumination and imaging conditions and for a surface with known reflectance properties, changes in surface orientation translate into corresponding changes in image intensity. The inverse problem — recovering surface shape from changes in image intensity — is known as the “shape-from-shading problem.” The following discussion summarizes the procedure for solving this problem using the surface smoothness constraint. Details can be found in Horn³ and Ikeuchi and Horn.¹¹²

The relationship between image irradiance $E(x,y)$ and the orientation (p,q) of the corresponding point on the surface is given by

$$E(x,y) = R(p,q) \quad (4.4)$$

where $R(p,q)$ is the reflectance map of the surface. The goal is to recover surface shape by calculating the orientation (p,q) on the surface for each point (x,y) in the image. Note that although we have only one equation, two degrees of freedom for orientation exist. Thus, this problem cannot be solved unless additional constraints are imposed. A commonly imposed constraint is that of surface smoothness. We assume that objects are made up of piecewise smooth surfaces that depart from smoothness constraint only along edges.

Ikeuchi and Horn¹¹² described an iterative method to solve the shape-from-shading problem using occluding contours to supply boundary conditions. Since points along the occluding contours correspond to points at infinity in gradient space, they proposed the use of a stereographic projection coordinate system (f,g) instead of a gnomonic projection coordinate system (p,q) , as shown in Figure 4.4.³ In the stereographic projection system, points on the occluding contour correspond to points on a circle of radius 2 units in the (f,g) plane. Let $R(p,q)$, expressed in this new coordinate system, be denoted by $R_s(f,g)$. The smoothness constraint can be specified as minimizing the integral of the sum of the squares of the partial derivatives of surface orientation given by

$$e_s = \iint ((f_x^2 + f_y^2) + (g_x^2 + g_y^2)) dx dy \quad (4.5)$$

where the subscripts denote the variables of partial derivatives. Strictly speaking, we must minimize this integral subject to the constraint specified in the equation $E(x,y) = R(p,q)$; however, to account for noise that causes departure from this equation, the problem is posed as that of minimizing total error e , given by

$$e = e_s + \lambda e_i \quad (4.6)$$

where λ is a parameter that weighs the error in smoothness constraint relative to the error in image irradiance equation, which is given by

$$e_i = \iint (E(x,y) - R_s(f,g))^2 dx dy \quad (4.7)$$

Minimizing total error is a problem in the calculus of variations. An iterative solution for updating the value of (f,g) during the $n+1$ iteration is given by Ikeuchi and Horn, as follows:¹¹²

$$\begin{aligned} f_{ij}^{n+1} &= f_{ij}^{*n} + \lambda [E_{ij} - R_s(f_{ij}^{*n}, g_{ij}^{*n})] \frac{\partial R_s}{\partial f}, \\ g_{ij}^{n+1} &= g_{ij}^{*n} + \lambda [E_{ij} - R_s(f_{ij}^{*n}, g_{ij}^{*n})] \frac{\partial R_s}{\partial g}, \end{aligned} \quad (4.8)$$

where $*$ denotes the average values computed in a 2×2 neighborhood and the subscript ij denotes discrete coordinates in the image plane. Although the computations for a given iteration are local, global consistency is achieved by the propagation of constraints over many iterations.

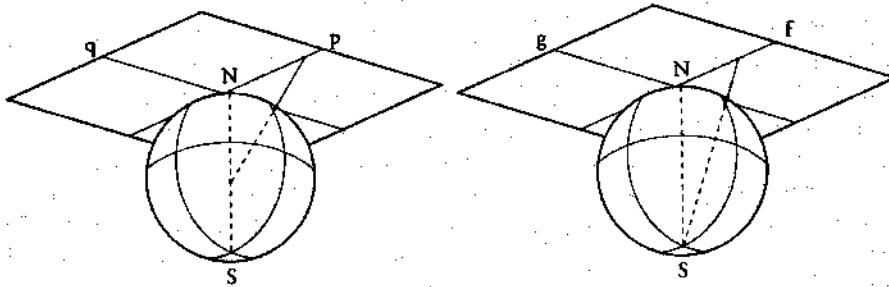


Figure 4.4: Illustration of gnomonic (left) and stereographic (right) projection (from B.K.P. Horn, *Robot Vision*, © 1986 McGraw-Hill, Inc., reprinted with permission.)³

In the shape-from-shading problem, we are attempting to recover the surface shape from a single image. When multiple images acquired from different illumination orientations are available, shape recovery is simplified. This technique, known as "photometric stereo," is described in the next section.

Shape from photometric stereo. In photometric stereo,¹¹³ two or more images of the scene are captured under different orientations of illuminating sources. Both the camera and the objects in the scene are assumed to remain in the same position, which implies that a point in the scene will have the same image coordinates in all the images. The correspondence problem of finding matching points in two images is thus avoided. Under these assumptions, we have as many equations of the form given by $E(x,y) = R(p,q)$ as there are images. If these equations are linear and independent, just two images are adequate to solve for the surface orientation parameters p and q . For other cases, we may need more than two images to obtain a unique solution. For example, in the case of a Lambertian surface illuminated by two different orientations of point sources, at most two surface orientations exist that produce a particular pair of image intensity values. These surface orientations are obtained by finding the intersections of corresponding contours in the gradient space, as shown in Figure 4.5.³ To find a unique solution, a third image is required. In principle, we are not required to impose additional constraints—such as surface smoothness—to recover shape from photometric stereo. However, applying such constraints helps in obtaining better results when the data are corrupted by noise. Having more than two images also helps to recover albedo, which characterizes the changes in reflectance properties of surfaces made up of the material. In other words, the actual reflectance map for a surface is the product of its albedo times the normalized reflectance map for the material.

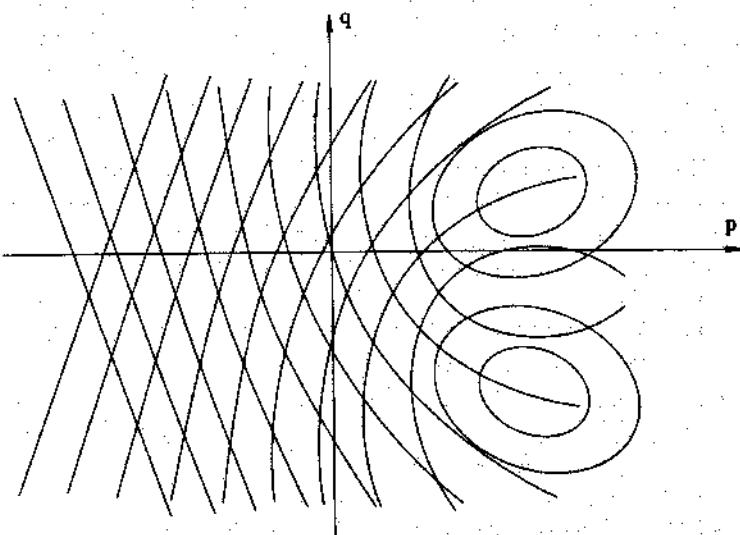


Figure 4.5: Illustration of shape recovery using photometric stereo method (from B.K.P. Horn, *Robot Vision*. Copyright © 1986 McGraw-Hill, Inc., reprinted with permission.)³

Shape from stereo. In the previous section, a method was described for recovering shape from multiple images acquired from different lighting conditions. Multiple images are much more easily captured by changing the camera position. In the literature, results of extensive research to recover three-dimensional shape from stereo pairs of images are reported. We do not even attempt to summarize these results here; we simply illustrate the principles and general methodologies involved.

A simple camera geometry for binocular-stereo imaging is shown in Figure 4.6.³ Let the baseline connecting the two lens centers be oriented parallel to the x -axis and perpendicular to the optical axes of the imaging system. The images of a point (x, y, z) are formed at (x_l, y_l) and (x_r, y_r) in the left and right camera planes, respectively. In the imaging system shown in Figure 4.6, with identical cameras and coplanar image planes that are perpendicular to the plane containing the two optical axes, the two y -coordinates, y_l and y_r , will be equal. The relationship between the disparity $(x_l - x_r)$ and the depth z , in terms of the imaging-system parameters, baseline length b , and focal length f , is given by

$$z = \frac{bf}{(x_l - x_r)} \quad (4.9)$$

Thus, the depth can be recovered if the disparities of corresponding image points are known.

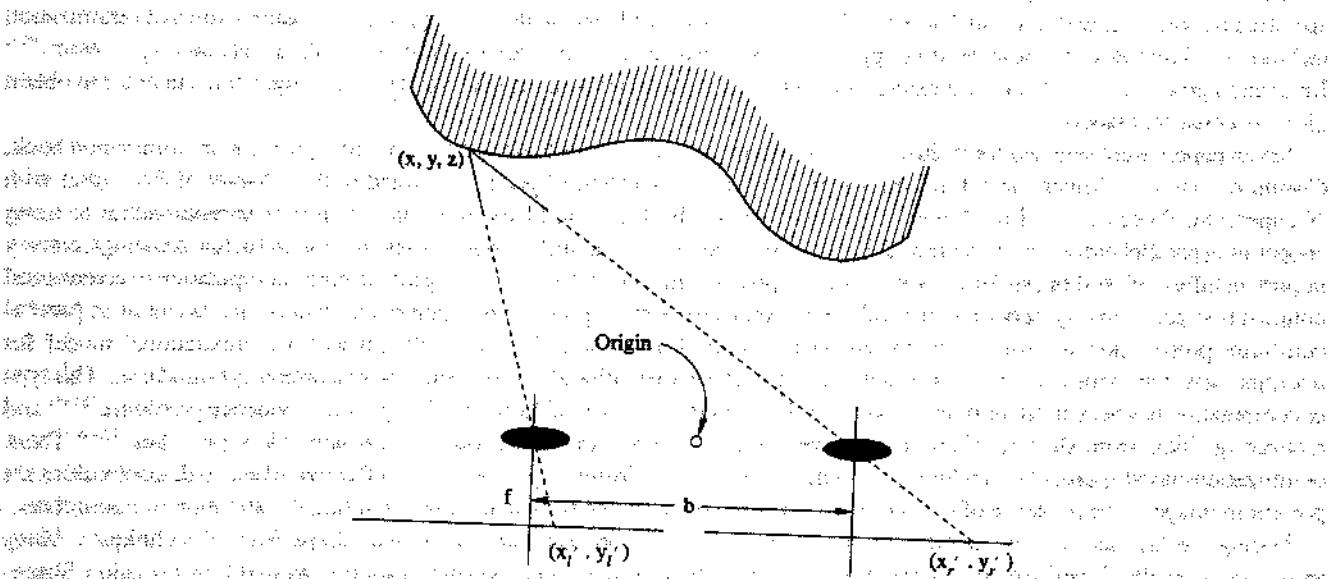


Figure 4.6: Simple camera geometry for stereo imaging (from B.K.P. Horn, *Robot Vision*. Copyright © 1986 McGraw-Hill, Inc., reprinted with permission.)³

One of the primary problems in recovering depth is finding corresponding points in the two images. One of the many methods developed for finding corresponding points is to identify "interesting points" in both images and to match these points using methods described in the previous chapter. Note that the image points corresponding to a scene point lie along the line of intersection between the image plane and the epipolar plane, which is defined as the plane containing the scene point and the two lens centers; thus, the search space is limited significantly. Such matching techniques yield sparse depth maps; additional constraints are applied to recover depth at all points. For example, iterative relaxation techniques that use the surface smoothness constraint are often employed to obtain consistent solutions.

A principal difficulty in stereo reconstruction is the selection of interesting points. This selection typically is based on high local variance in intensity, which unfortunately occurs more frequently at corners and at other surface discontinuities, where the smoothness constraint does not hold well. In some machine vision applications, the problem of selecting interesting points is solved by using structured light. Patterns of light are projected onto the surface, creating interesting points even in regions that would be otherwise smooth. Finding and matching such points are further simplified when the geometry of the projected patterns is known. Since these patterns create artificial texture on the surfaces, shape-from-texture techniques, described in the next section, can also be used to recover shape.

Shape from texture. Image plane variations in texture properties such as density, size, and orientation are the cues exploited by shape-from-texture algorithms.¹⁴ For example, the orientation of the surface can be determined using the texture gradient,

defined as the magnitude and direction of maximum change in the primitive size of texture elements. Also useful in determining orientation is quantifying the changes in the shape of the texture elements (e.g., circles appearing as ellipses). Orientation can be uniquely determined by finding vanishing points in images of surfaces with textures made up of regular grids of lines that are possibly attributable to structured lighting.¹¹⁵ Blostein and Ahuja¹¹⁶ describe an integrated algorithm for extracting texture elements and estimating surface orientation.

Shape from focus. That imaging systems have a finite depth of field is well known. Thus, only objects at a proper distance appear focused in the image, whereas those at other depths are blurred, in proportion to their distances. Algorithms to exploit this blurring effect have been proposed recently.¹¹⁷⁻¹¹⁹ The image is modeled as a convolution of focused image, with a point spread function determined by the camera parameters and the distance of the object from the camera. The depth is recovered by estimating the amount of blur in the image and using the known or estimated line spread function.

Research trends

Constraints play a very important role in all levels of vision. In the early processing of visual information, constraints are usually specified by some general laws, such as smoothness of surfaces, and then an iterative approach — using a form of optimization technique — is employed. These iterative approaches usually result in computations commonly called “relaxation processes.”¹¹¹ Relaxation processes have been used extensively to propagate local constraints in computer vision algorithms in order to obtain globally consistent labels.

Seven papers were selected for inclusion in this chapter: four are in this book and the remaining three in its companion book, *Computer Vision: Advances and Applications*. We begin the collection of papers reprinted in this chapter of *Principles* with “Cooperating Processes for Low-Level Vision: A Survey,” by Davis and Rosenfeld, in which various approaches to using cooperating parallel processes are reviewed. Brightness discontinuities at surface boundaries, as depicted in line drawings, convey important information about surface shape. Ullman¹²⁰ presents an approach for designing a distributed computation of constrained optimization problems by networks of locally interconnected simple processors. Such computations are essential in parallel constraint propagation for solving many vision problems. Barrow and Tenenbaum¹²¹ present a computational model for interpreting line drawings based on constraints on local surface orientation along extremal and discontinuity boundaries. This type of computation has been used for many tasks, including determining optical flow,¹²² solving correspondence problems,^{123,124} and recovering shape-from-shading,¹¹² and is very well suited to implementations using neural network approaches.^{125,126} These optimization-based approaches, including regularization and random-field approaches, are effective when no discontinuities are present in images. The presence of discontinuities poses serious problems that are now attracting the attention of researchers.

During the last decade, one of the most active research topics in computer vision was shape-from-X techniques. Many techniques were developed to recover three-dimensional information from images using a variety of available information. Shape-from-shading¹²⁷ and shape-from-texture¹²⁸ techniques have been investigated by several groups. Ikeuchi and Horn¹¹² describe an iterative shape-from-shading algorithm using occluding boundary information. The next set of papers describe various techniques for shape recovery from image cues. In *Principles*, “Photometric Method for Determining Surface Orientation From Multiple Images,” by Woodham describes a method for recovering shape information from multiple images acquired by varying the direction of illumination. Since the position of objects relative to the camera remains unchanged, this photometric-stereo technique does not suffer from the correspondence problem. In *Advances and Applications*, Nayer et al. in “Shape From Interreflections,” consider the problem of concave objects resulting in secondary illuminations.

Controlling the illumination in practical applications is often not possible, whereas obtaining stereo pairs of images of the same scene by using cameras that are spatially separated by a known distance is relatively simple. Analysis of such stereo pairs of images has received considerable attention in computer vision research. We have included “Region-Based Stereo Analysis for Robotic Applications,” by Marapane and Trivedi in *Principles* that covers this topic; Ohta and Kanade¹²⁹ also discuss stereo analysis. The most popular and well-investigated shape-from-X technique is certainly stereo. Many different forms of stereo have been investigated, including trinocular stereo^{130,131} and motion stereo.^{132,133} For solving difficulties in correspondence problems, different features were tried. Interesting points,⁹⁴ lines,¹³⁴⁻¹³⁶ and areas^{137,138} have all been used. The current trend is to combine stereo and motion^{139,140} and to use correspondenceless methods.^{141,142}

Analysis of texture plays a very important role in many applications of machine vision. Papers describing texture modeling, analysis, and segmentation are discussed in earlier chapters.^{39,42,59} In “Shape From Texture: Integrating Texture-Element Extraction and Surface Estimation,” Blostein and Ahuja in *Principles* describe a method for identifying texture elements while simultaneously recovering the orientation of the textured surfaces. In this method, true texture elements are selected from a set of candidates by finding the planar surface that best predicts the observed texture elements.

In some applications, especially those requiring qualitative depth information, shape from focus may prove very useful.^{117,119}

The finite depth-of-focus in an imaging system results in blurring of edges of out-of-focus objects. In *Advances and Applications*, "Depth Recovery From Blurred Edges," by Subbarao and Gurumoorthy describes a method that exploits this blurring effect to recover depth information. Many such inverse reconstruction problems are mathematically ill-posed. Reformulating these inverse reconstruction problems as well-posed variational principles using regularization theory has been studied. We conclude *Advances and Applications* with the paper entitled "Regularization of Inverse Visual Problems Involving Discontinuities," by Terzopoulos, which describes a method that reformulates such an inverse reconstruction problem using controlled-continuity constraints. Clearly, approaches in which information from various clues is combined or in which various sensors are used will be successful in recovering three-dimensional information. Such approaches are now receiving attention.^{143,144}

High-level constraints are domain dependent. Approaches that use high-level constraints are the main focus of knowledge-based vision systems^{145,146} and are also discussed in Chapter 7.

References Cited Chapter 4

110. Barrow and Tenenbaum, "A System for Reasoning About Scences," Technical Note #121, Technical Note #MSYS, Artificial Intelligence Center, Stanford Research Institute, March 1976.
111. A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 6, 1976, pp. 420-433.
112. K. Ikeuchi and B.K.P. Horn, "Numerical Shape from Shading and Occluding Boundaries," *Artificial Intelligence*, Vol. 17, 1981, pp. 141-184.
113. R.J. Woodham, "Photometric Method for Determining Surface Orientation from Multiple Images," *Optical Engineering*, Vol. 29, No. 1, 1980, pp. 139-144.
114. J.J. Gibson, *The Perception of the Visual World*, Riverside Press, Cambridge, Mass., 1950.
115. J.R. Kender, *Shape from Texture*, PhD thesis, Carnegie-Mellon University, 1980.
116. D. Blostein and N. Ahuja, "Shape from Texture: Integrating Texture-Element Extraction and Surface Estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 11, No. 12, 1989, pp. 1233-1251.
117. A.P. Pentland, "A New Sense for Depth of Field," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 9, No. 4, 1987, pp. 523-531.
118. P. Grossman, "Depth from Focus," *Pattern Recognition Letters*, Vol. 5, 1989, pp. 63-69.
119. M. Subbarao and N. Gurumoorthy, "Depth Recovery from Blurred Edges," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1988, pp. 498-503.
120. S. Ullman, "Relaxation and Constrained Optimization by Local Processes," *Computer Graphics and Image Processing*, Vol. 10, No. 2, 1979, pp. 115-125.
121. H.G. Barrow and J.M. Tenenbaum, "Interpreting Line Drawings as Three Dimensional Surfaces," *Artificial Intelligence*, Vol. 17, 1981, pp. 75-116.
122. B.K.P. Horn and B.G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, Vol. 17, 1981, pp. 185-203.
123. D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity," *Science*, Vol. 194 (4262), 1976, pp. 283-287.
124. D. Marr and T. Poggio, "A Computational Theory of Human Stereo Vision," *Proc. Royal Soc. Lond. B*, Vol. B204, 1979, pp. 301-328.
125. G.E. Hinton and T.J. Sejnowski, "Learning and Relearning in Boltzmann Machines," *Parallel Distributed Processing*, Vol. 1: Foundation, MIT Press, Cambridge, Mass., 1986.
126. Hutchinson, et al, "Computing Motion Using Analog and Binary Resistive Networks," *Computer*, Vol. 21, No. 3, 1988, pp. 52-63.
127. B.K.P. Horn and M.J. Brooks, eds., *Shape from Shading*, MIT Press, Cambridge, Mass., 1989.
128. K. Kanatani and T.C. Chou, "Shape from Texture: General Principle," *Artificial Intelligence*, Vol. 38, No. 1, 1989, pp. 1-48.
129. Y. Ohta and T. Kanade, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 7, No. 2, 1985, pp. 139-154.
130. N. Ayache and F. Lustman, "Fast and Reliable Trinocular Stereovision," *Proc. First Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1987, pp. 422-427.
131. C. Hansen, N. Ayache, and F. Lustman, "Towards Real-Time Trinocular Stereo," *Proc. Second Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1988, pp. 129-133.
132. R. Nevatia, "Depth Measurement by Motion Stereo," *Computer Graphics and Image Processing*, Vol. 5, 1976, pp. 203-214.
133. R.C. Jain, S. Bartlett, and N. O'Brien, "Motion Stereo Using Ego-Motion Complex Logarithmic Mapping," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 9, No. 3, 1987, pp. 356-369.
134. N. Ayache and B. Faverjon, "Efficient Registration of Stereo Images by Matching Graph Descriptions of Edge Segments," *Int'l J. Computer Vision*, 1987, pp. 107-131.
135. L.B. Wolff, "Accurate Measurement of Orientation from Stereo Using Line Correspondence," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 410-415.
136. G. Medioni and R. Nevatia, "Segment-Based Stereo Matching," *Computer Vision, Graphics, and Image Processing*, Vol. 31, 1985, pp. 2-18.
137. L. Cohen et al, "Hierarchical Region-Based Stereo Matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 416-421.
138. S.B. Marapane and M.M. Trivedi, "Region-Based Stereo Analysis for Robotic Applications," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 19, No. 6, 1989, pp. 1447-1464.
139. Z. Zhang, O.D. Faugeras, and N. Ayache, "Analysis of a Sequence of Stereo Scenes Containing Multiple Moving Objects Using Rigidity Constraints," *Proc. Second Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1988, pp. 177-186.
140. E. Grosso, G. Sandini, and M. Tistarelli, "3-D Object Reconstruction Using Stereo and Motion," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 19, No. 6, 1989, pp. 1465-1476.
141. B.D. Lucas, and T. Kanade, "Optical Navigation by the Method of Differences," *Proc. Int'l Joint Conf. Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., San Mateo, Calif., 1985, pp. 981-984.
142. K. Skifstad and R.C. Jain, "Range Estimation from Intensity Gradient Analysis," *Machine Vision and Applications*, Vol. 2, 1989, pp. 81-102.
143. E.P. Krotkov, *Active Computer Vision by Cooperative Focus and Stereo*, Springer-Verlag, New York, N.Y., 1989.
144. J.Y. Aloimonos and D. Shulman, *Integration of Visual Modules: An Extension of the Marr Paradigm*, Academic Press, Boston, Mass., 1989.

145. D.M. McKeown, Jr., "Building Knowledge-Based Systems for Detecting Man-Made Structures from Remotely-Sensed Imagery," *Proc. Royal Soc. Lond. B*, Vol. A324, 1988, pp. 423-435.
146. B.A. Draper et al., "The Schema System," *Int'l J. Computer Vision*, Vol. 2, No. 3, 1989, pp. 209-250.

Cooperating Processes for Low-level Vision: A Survey

Larry S. Davis

*Department of Computer Sciences, University of Texas,
Austin, TX 78712, U.S.A.*

Azriel Rosenfeld

*Computer Science Center, University of Maryland,
College Park, MD 20742, U.S.A.*

ABSTRACT

Cooperating local parallel processes can be used as aids in assigning numerical or symbolic labels to image or scene parts. Various approaches to using such processes in low-level vision are reviewed, and their advantages are discussed. Methods of designing and controlling such processes are also considered.

1. Introduction

The early stages of computer vision involve assigning symbolic and numerical labels to image parts. For example, pixels can be assigned symbolic land-use category labels based on their spectral signatures, or numerical stereo or motion disparity labels based on local comparisons between pairs of pictures.

The enormous amount of data comprising an image demands that such labelling processes be very fast. Sequential labelling processes, while they can make full use of context, cannot be speeded up in general. Moreover, the labellings which they compute are often sensitive to the order in which the parts are considered. A more promising approach—one that is also motivated by studies of biological visual systems—is to make the processes highly *parallel*. This requires that each picture part be analyzed and labelled independently of the others. When we do this, however, many errors are made, because contextual information is not adequately used.

A solution to this problem is to assess the labelling possibilities for every part independently and then compare each part's assessments to those of other,

Artificial Intelligence 17 (1981) 245-263

0004-3702/81/0000-0000/\$02.50 © North-Holland

related parts, in order to detect and correct potential inconsistencies. Since both the assessment and the comparison can be done independently for every part, each stage of the process is parallel. On the other hand, context is now being used at the comparison stage, when related parts are able to communicate and 'cooperate'. To keep the computational cost low, the comparisons should be *local*; they should involve only parts that are directly related (e.g., neighboring pixels). This localness can be compensated for by *iterating* the comparison process, in order to allow information to propagate.

These considerations lead naturally to the design of a 'cooperative' approach to labelling picture parts which allows context to be used in the labelling process while still permitting fast parallel implementation and low computational cost. Such processes are called 'relaxation' processes, because of their resemblance to certain iterative processes used in numerical analysis. Very generally, a relaxation process is organized as follows:

- (a) A list of possible labels is independently selected for each part, based on its intrinsic characteristics. A measure of confidence can also be associated with each possible label.¹
- (b) The possibilities (and confidences) for each part are compared with those for related parts, based on a model for the relationships between the possible labels of picture parts. Labels are deleted or confidences are adjusted to reduce inconsistencies.

(c) Step (b) can be iterated as many times as required.

This approach is very general: We have not specified how to formulate label relationship models, choose possibilities, estimate confidences, or adjust them; nor have we discussed when the process should be iterated, and if so, how many times. The next three sections of this paper discuss these issues, and survey applications of such processes to problems in low-level computer vision.

2. Cooperation/Competition

A relaxation process is a computational mechanism which allows a set of 'myopic' local processes associated with picture parts to interact with one another in order to achieve a globally consistent interpretation of a picture. This interaction involves the updating of each picture part's self-assessment which is represented as a discrete or fuzzy *labelling*. A discrete labelling simply associates a set of possible labels, or names, with each picture part, while a fuzzy labelling additionally associates a likelihood with each label.

Labels are usually specified extensionally by actually listing the appropriate labels for each picture part. The list is a subset of some given, finite universe of labels. For some applications the natural label set is infinite. For example, the

¹It is tacitly assumed that the correct label of each part is on the initial list of labels for that part.

label for a picture part might represent the range, or distance, from the sensor to some specific point in the picture part such as its centroid. In such cases, a labelling may need to be specified intensionally; for example, an interval of numbers may be used to specify the range—i.e., we assume that the true range is between a nearest distance r_1 and a farthest distance r_2 . All the applications we will consider in this paper use only finite universes of labels.

2.1. Neighborhood models

A relaxation process is determined by specifying a model for the neighborhood of a picture part and a model for the interaction between labellings of neighboring picture parts.

The neighborhood model for a relaxation process specifies which pairs of picture parts directly communicate with one another in the relaxation process, and determines the topology of the graph on which the relaxation process operates. This graph has individual picture parts as nodes. Its arcs connect those pairs of parts that communicate with one another. The neighborhood model is usually designed to establish connections only between 'nearby' parts to satisfy the locality constraint.

A neighborhood model is specified by a set of neighbor relations $r = \{r_1, r_2, \dots, r_n\}$. Each r_i is a binary relation defined over the appropriate set of picture parts. For example, if the picture parts are pixels, then the neighborhood model might specify that a pixel is connected to every pixel in its 3×3 neighborhood. In this case, there are still several possibilities for the relations contained in the set r . For example, r might be the set {directly above, directly below, etc.} which would distinguish between pairs of points that are horizontally adjacent, vertically adjacent, etc., or it could be the singleton relation 'in the 3×3 neighborhood'. In the latter case, the connections between pairs of pixels would not be recoverable from the graph on which the relaxation process will operate. The choice of r will, in general, be determined by the isotropy of the universe of labels. For example, if we are designing a relaxation process for edge reinforcement, then the relative positions of pixels are crucial since edges generally 'line up', while if we are designing a relaxation process to enhance an image's grey levels, then the positional information may not be required.

When the picture parts are regions rather than pixels, then connections might be formed between adjacent regions only. In some situations, it might be necessary to distinguish between regions that are above, below, inside, surrounding, etc.

The neighborhood model determines which pairs of picture parts directly communicate through the relaxation process. The next section discusses the various ways in which they may communicate.

2.2. Interaction models

The interaction model defines how a picture part changes its labelling based on the labellings of its neighbors. An interaction model is composed of two parts:

- (1) a knowledge representation for the relationships between labels, and
- (2) a mechanism, or procedure, for applying the knowledge in (1) to change, or update, labellings.

For discrete labellings the simplest knowledge representation is a set of the pairs of labels that can simultaneously be associated with pairs of neighboring picture parts. It can be represented by a binary relation R defined over the universe of labels D . Intuitively, $(d, d') \in R$ if a pair of neighbors can simultaneously be labelled with d and d' . In general, there is a binary relation associated with each neighbor relation.

The most obvious updating mechanism is a label discarding process, which looks at pairs of picture parts at a time. A label, d , can be deleted from the labelling of a picture part if, for some neighboring picture part, that neighbor does not contain a label, d' , in its labelling with $(d, d') \in R$. This is, essentially, Waltz's filtering algorithm [1]. Rosenfeld et al. [2] show that label discarding can, in principle, be applied in parallel at every picture part and that by iterating the process of discarding labels a unique, maximally consistent labelling is computed. The process can be generalized in a variety of ways—e.g., the knowledge representation might be in terms of n -ary relations (for example, a 3-ary relation is required to specify that a picture part is between two others). The label discarding process now considers a picture part and $n-1$ of its neighbors at a time, rather than one neighbor at a time. There are many other possibilities based on computing lower-order projections of n -ary relations. See Haralick et al. [3] and Haralick and Shapiro [4] for a detailed discussion.

The binary relation knowledge representation can be generalized to fuzzy labellings by specifying a real-valued compatibility function, C , whose domain is $D \times D$. As before, in general, a compatibility function is defined for each picture relation in the set r . A variety of applications have used compatibility functions whose range is $(-1, 1)$. Intuitively, if $C(d, d') = -1$, then d and d' are maximally incompatible, and the strong presence of d' at one picture part (i.e., d' has a high likelihood at that part) should depress the likelihood of d at a neighboring picture part. If $C(d, d') = 1$, then d and d' are maximally compatible, and the strong presence of d' at a picture part should increase the likelihood of d at a neighboring picture part. Finally, if $C(d, d') = 0$, then the presence of d' at a picture part should have no effect on the likelihood of d at a neighboring part. Intermediate values of C should have intermediate effects.

As an example, suppose we are designing a relaxation process to enhance the results of a local line detection algorithm. Then the set of labels may be horizontal (h), vertical (v), left-diagonal (dl), and right-diagonal (dr), and the set r might contain the relations vertically-adjacent (V), horizontally-adjacent (H), left-diagonally-adjacent (L) and right-diagonally-adjacent (R). If, in the

COOPERATING PROCESSES FOR LOW-LEVEL VISION

class of images being considered, linear features are thin and have few corners (i.e., the curvature is ordinarily low), then we would expect, e.g., that $C_V(v,v)$ would be high, while $C_V(v,h)$ would be low, since an h vertically adjacent to a v would form a right angle. $C_H(v,v)$, on the other hand, would be low, since the linear features are thin.

Several mechanisms have been suggested for applying this knowledge representation to updating labellings. For example, Rosenfeld et al. [2] suggested the formula:

$$p'_i(d) = p_i(d)(1 + Q_i(d))/N$$

where

$$Q_i(d) = \sum m_i \sum_{d'} C(d,d') p_i(d')$$

and N is a normalizing factor which guarantees that $\sum p'_i(d) = 1$. The m_i values can be used to give higher weight to some neighbors at part i than others. $Q_i(d)$ measures the overall support of the neighborhood of part i for label d ; it takes on values in the range $[-1, 1]$ and can be interpreted similarly to C . The above operation is applied in parallel at every part and for every label. The p' values then replace the p values, and the operation can be iterated.

Variations on the above theme are possible and lead to better results in some applications. For example, one can apply a 'max-min' rule where

$$Q_i(d) = \min \{ \max_{d'} \{ C(d,d') p_i(d') \} \}$$

which reduces to the discrete algorithm described above when C and p are constrained to take on only the values 0 or 1.

There are several disadvantages to the relational knowledge representation for the interactions between labels. First, it is a single-level representation scheme. The solution to many image understanding problems requires that images be described at several levels of abstraction. Attempting to compile all interactions between conceptually higher-level pictorial entities down to interactions between only the lowest level pictorial features is almost always cumbersome and inefficient, and is sometimes impossible. Section 2.3 discusses hierarchical relaxation systems.

A second important shortcoming of the relational framework is that the algebraic combination of evidence treats all of the interactions between labels uniformly, which is often not desirable. Furthermore, there are classes of intuitively plausible constraints that can only be represented very inefficiently in a relational framework. For example, the very simple constraint

A picture part can be a d_1 only if all adjacent picture parts
can be d_2 's

requires an n -ary relation to represent it, where n is the maximum degree of

any node in the graph on which the relaxation procedures operates.

Such problems can be overcome by adopting a more powerful representation for label interactions than relations. For example, constraints between labels can be represented using logic statements [5]. This allows a much wider class of constraints to be efficiently represented and applied to the analysis of a picture. The natural mechanism for applying the constraints is, then, a general inference procedure. Such a scheme has not yet been applied to any image understanding problem; its application to linear feature detection is currently under investigation.

2.3. Hierarchy

Very often, a natural and economical solution to an image analysis problem requires that pictorial entities be described at several levels of detail. For example, to recognize an image segment as the top view of an airplane based on the shape of its boundary might require recognizing airplane pieces as engines, wings, tail sections, etc., and then grouping them into larger pieces of airplanes, and finally into a complete airplane shape. Or, as a second, more complex, example, reading a word in cursive script involves segmenting the word into primitive parts such as strokes, grouping the strokes into large letter pieces, those pieces into letters and finally the letters into a word.

As discussed above a single level relaxation system is specified by a neighborhood model and a label interaction model. To design a hierarchical relaxation system having k levels, one needs to not only define a neighborhood model and an interaction model at each level, but also a construction model (which, given the labelling of pieces at level m , can construct the pieces and their labellings at level $m+1$), and an across-level neighborhood model. This last model is ordinarily based simply on constituency—i.e., a level $m+1$ piece is linked to each of the level m pieces from which it was formed [6].

An important design criterion for such processes is that the construction models and the interaction models be *consistent*. Intuitively, the consistency constraint means that the relations between level m labels implied by the construction models for all higher levels do not contradict the explicit relations between level m labels mentioned in the level m interaction model. A simple example should help clarify this point.

Suppose that we are constructing a hierarchical relaxation system for reading cursive script, and that for the particular corpus of words that we wish to read, there are no words in which the letter 'h' precedes the letter 'u'. Now, suppose that at the large letter piece level, an 'h' ends with a piece p_1 and a 'u' begins with a piece p_2 , and that no other letter contains either a p_1 or a p_2 . Then clearly, the compatibility of p_1 and p_2 at the large letter piece level should be as low as possible. If it were not, then the construction model taking letters into

COOPERATING PROCESSES FOR LOW-LEVEL VISION

words would be inconsistent with the interaction model for large letter pieces.

One possible approach to guaranteeing such consistency is to specify only the construction models, and then *compile* the interaction models from the construction models. This not only guarantees that the interaction models are consistent with the construction models, but also avoids the tedious task of specifying all the constraints contained in the interaction models.

For example, in reading cursive script from a known corpus, the letter cooccurrence probabilities can be compiled directly from the corpus, and these can be used as an interaction model at the letter level. Then, given a decomposition of letters into large letter pieces a similar process can produce cooccurrence probabilities for large letter pieces, etc. This was done by Hayes [7] in his handwriting analysis system.

As a second example, in [8] Davis and Henderson describe a hierarchical shape analysis system. Shapes are modeled by hierarchical relational networks which describe the arrangement and geometrical properties of shape pieces at several levels of detail. The representation is designed in such a way that local constraints about the appearance of the shape can be automatically compiled from the representation. Thus the representation serves as a set of construction models, and the compiled constraints are used as interaction models.

Although the compilation of interaction models from construction models is a powerful idea in the design of hierarchical relaxation systems, it does not address the issue of how one determines whether or not additional constraints, not derivable from the construction model, can be consistently added to the interaction models. This situation might arise when analysis of one part of an image yields information that can be used to guide the analysis of other parts, or if prior knowledge is available that is not ordinarily available. The following simple example illustrates the problem. Suppose that we are attempting to recognize airplanes, and that prior information is available about the angle that the wings of planes will make with the fuselage. How can we determine that this extra information is consistent with the existing airplane model? (If it is not, no shapes will be recognized as airplanes!) How does the relaxation process even make use of this information, assuming that it is consistent with its interaction models? For currently existing systems, there is no effective means for checking the consistency of externally specified information with current knowledge; or for uniformly applying such information to enhance the relaxation process. This points out another advantage of the logic representation mentioned in Section 2.2. If construction models are specified as statements in logic, then interaction models can still be compiled from the construction models. Furthermore, the consistency of added information with current knowledge can be determined, and the general inferencing capabilities associated with logic would enable such a system to make use of any additional information as well.

3. Applications

A wide variety of labelling processes can be used at various stages of computer vision. Many of these processes operate at the pixel level—i.e., the parts to be labelled are individual pixels, and the interaction is between neighboring pixels. In general, image segmentation can be regarded as pixel labelling, with the labels defining a partition of the image into subsets. Thus relaxation methods are applicable to most of the standard image segmentation techniques, including pixel classification based on gray level or color (thresholding, multispectral classification), as well as detection of local features (peaks or spots, ridges or curves, edges, corners, or matches to arbitrary templates). Many examples of such methods are given in Sections 3.1 and 3.2.

Relaxation methods can also be applied to situations involving several images (e.g., disparity measurement in motion or stereo), or several sets of labels simultaneously applied to a single image (e.g., ‘intrinsic image’ labels); see Section 3.3. They can also be used to label picture parts that are larger than single pixels, i.e., windows or regions, and to detect specified local configurations of such parts, as briefly discussed in Section 3.4.

3.1. Pixel classification based on gray level or color

Suppose that a scene is composed of a few objects or regions each of which is homogeneous in color. The colors of the pixels in an image of that scene should then display clustering behavior: There should be clusters in the scatter plot of color values, corresponding to the color characteristics of the regions. Under these circumstances, a natural way to segment the image is to classify the pixels as belonging to these clusters. In fact, this is the standard method of segmenting multispectral terrain images into terrain types or land use classes, based on clustering the spectral signatures of the pixels. It is also widely used to segment black-and-white images into light and dark regions by thresholding the gray levels so as to separate peaks on the histogram. Analogous methods can be used for arrays of other types of values, e.g., range data.

Conventionally, the pixels are classified independently. In order to use a cooperative approach, possible class memberships must be determined, or class membership confidences estimated, for each pixel, and the results compared with those for neighboring pixels. To define an appropriate interaction model, some assumptions must be made about the kinds of neighbors that we expect a pixel to have. Since the scene consists of a few homogeneous regions, most pixels will be in the interior of a region. The neighbors of those pixels should all be alike. Some pixels, of course, will be on interregion boundaries; in this case some of the neighbors of the pixel will belong to the same region as the pixel itself, but others will belong to a different region. Neighborhoods at which three or more regions meet will be rare, and will be ignored here.

The situation just described is characteristic of a large class of cooperative

pixel labelling problems: the neighbors of a pixel belong, with rare exceptions, to at most two classes, one of which is the class containing the pixel itself. Several types of interaction models can be used in such situations:

(1) The neighborhood used can consist, for each pixel, of those neighbors that most resemble the pixel. If the neighbors cluster into two classes, this is straightforward; if not, one can use a fixed number of 'best' neighbors, on the assumption that these neighbors are the ones most likely to belong to the same region as the pixel. For the chosen neighbors, the interaction model is then quite simple: like reinforces like [9].

Alternatively, we can examine a set of one-sided neighborhoods of the pixel, and choose the one in which the gray level or color is most homogeneous. This one is presumably contained within a single region, so that we can safely use a like-reinforces-like scheme on it [10, 11].

(2) If we do not want to commit ourselves to choosing a fixed set of neighbors, we can assign weights to the neighbors, or define link strengths between the pixels and the neighbors, such that neighbors similar to the pixel get high weights (or link strengths). In the interaction model, the reinforcement contributions are then proportional to the weights. For example, the m_i factors in eq. (2) can be chosen so as to give some neighbors more weight than others. If the reinforcement process is iterated, the weights can themselves be adjusted at each iteration, based on revised estimates of neighbor similarity—e.g., the m_i might change from one iteration to the next [12–14].

(3) Finally, we can simply use the same neighborhood for every pixel, and let like reinforce like. For pixels in the interior of a region, this behaves as desired; but on the border of a region, the pixel labels are likely to remain ambiguous, since they are being influenced by neighbors that belong to two regions. In fact, sharp corners on the borders will be smoothed out, since a pixel at such a corner will have most of its neighbors in another region, so that the reinforcement process will make it confident that it too belongs to that region [15, 16].

It should be pointed out that, rather than reinforce label confidences, we can adjust the pixel gray level or color values themselves; in other words, we can use the methods just described to smooth an image without blurring the edges between regions. Fig. 1 illustrates one of the methods.

The preceding discussion assumed that each region is 'flat', i.e., has relatively constant gray level or color. This is a reasonable assumption about some classes of scenes (e.g., characters on a printed page, chromosomes against a uniform background), but is not correct for others. More generally, the image can be modeled as piecewise linear and the relaxation process can examine a set of one-sided neighborhoods of the pixel, and choose the one that best fits a plane. Within the chosen neighborhood, the reinforcements should depend on closeness of fit to the hypothesized plane, rather than on similarity [17].

A similar approach applies, in principle, if we want to classify pixels based on the values of local properties measured over their neighborhoods, assuming

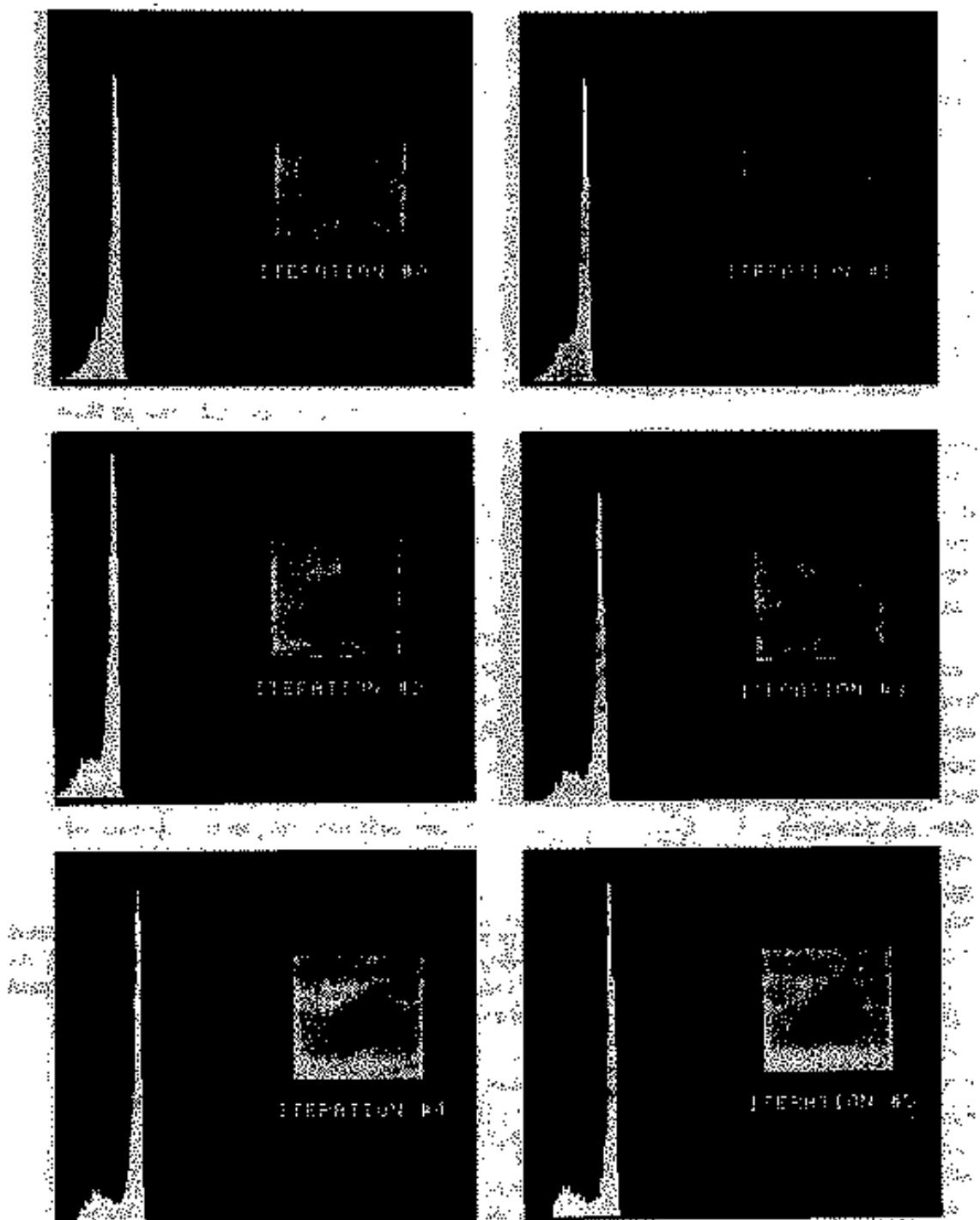


FIG. 1. Cooperative pixel classification based on gray level. (Lowest gray level: 1; highest gray level: 49.)

#0: Original (infrared image of a tank) and its histogram. The gray levels in this image were mapped into 'light' and 'dark' probabilities proportional to their distances from the ends of the grayscale.

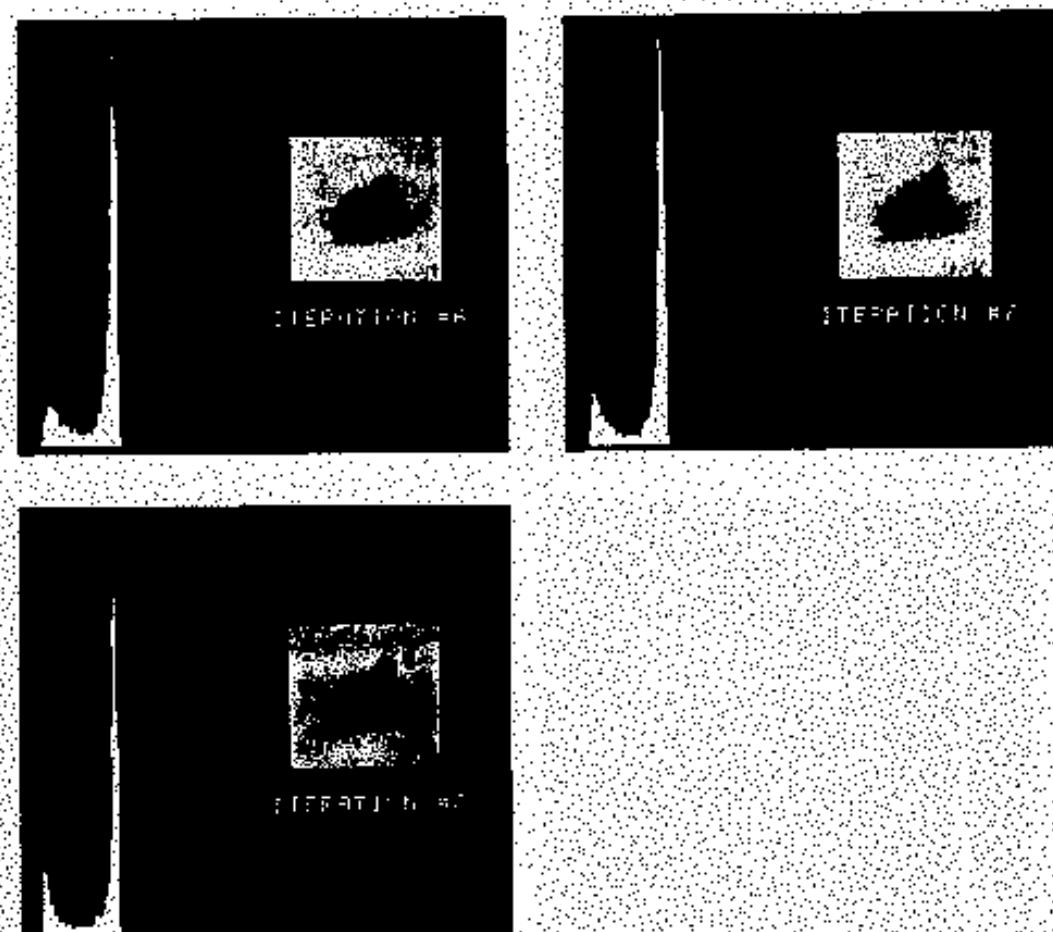


FIG. 1. Continued.

41-#8: Eight iterations of a relaxation process (like reinforcing like) applied to the initial probabilities, with the resulting probabilities displayed as gray levels using the inverse mapping. As the histograms show, the probabilities are tending toward (0, 1) and (1, 0), resulting in good discrimination between the tank and the background.

Iteration #	Mean pixel value	Scale (pixels/dot)
0	40.20	7
1	40	7
2	40.19	6
3	40.01	7
4	39.92	7
5	39.80	7
6	39.36	8
7	39.58	8
8	39.52	10

that the image consists of regions that are homogeneously textured. Here, however, larger neighborhoods should be used, since local properties tend to be more variable than single-pixel properties. Of course, when we use large neighborhoods, the problems encountered at region borders become more severe.

3.2. Local feature detection

If the labelling task involves local feature detection or template matching, we must use neighborhood and interaction models appropriate to the type of feature or pattern being detected. In the following paragraphs we discuss the detection of spots (i.e., peaks), streaks (ridges, curves), edges, and corners, as well as matches to an arbitrary template.

To detect peaks, i.e., local maxima, the neighborhood must be large enough to contain a peak. The peak label at a pixel is then positively reinforced by the presence of lower values at its neighbors, and negatively reinforced by higher values, where the amounts of reinforcement depend on the differences in value. In other words, small reinforces large, while large competes with large. Detection of pits (local minima) is exactly analogous. The same approach can be used to detect peaks on waveforms, histograms, or scatter plots [18, 19].

To detect ridges or ravines, i.e., high-valued lines or streaks on a low-valued background or vice versa, as in linear feature detection, we use a neighborhood somewhat larger than the streak width. The reinforcement model should now depend on the orientation of the streak: high values should reinforce one another along the streak, while low values should reinforce high values across the streak. To implement this, we initially estimate a streak confidence for each orientation, or more simply, estimate a single streak confidence and an associated streak direction. For neighbors in the direction along the streak, high values reinforce one another, provided the orientations are consistent; for neighbors in the direction across the streak, low values reinforce high ones, as in the case of peak detection [20, 21]. This approach is illustrated in Fig. 2.

Detecting edges is similar to detecting streaks, since an edge is a streak-like locus of high rates of change. Note that in this case directions must be measured modulo 360° rather than modulo 180°; in other words, for a given direction, we must take into account the sign of the rate of change, so that high edge values reinforce one another only if their dark sides and light sides match, and they compete otherwise. If desired, we can associate edge values with the 'cracks' between adjacent pairs of pixels, rather than with the pixels themselves; this is more appropriate if the edges are sharp [22, 23].

For both edges and streaks, our model implicitly assumes that they are straight or smoothly curved; if they have sharp corners or angles, dissimilar directions will be present in a single neighborhood, and these will compete with one another. To detect corners, we must allow a pixel to interact with pairs of

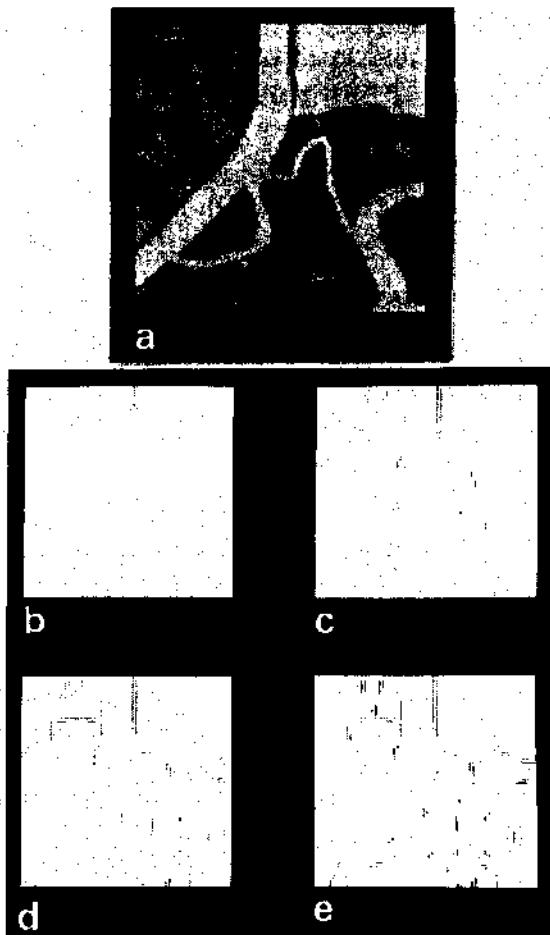


FIG. 2. Cooperative detection of smooth edges. (a): Portion of a LANDSAT image. (b): Result of applying an edge detector to (a), and scaling the results to yield a 'no edge' probability, and edge probabilities in each of 8 directions, at each point. For display purposes, if the highest probability is 'no edge', the point is displayed as white; otherwise, the highest edge probability is displayed as a gray level (1 = black). (c)-(e): Results of three iterations of a relaxation process (see text) applied to the initial probabilities displayed in the same way.

its neighbors, rather than with each neighbor separately; we can then reinforce the 'cornerity' value of the pixel if there exist pairs of neighboring edge or streak values that have sharply different directions. [Alternatively, we can detect corners in the gray level (or color) domain based on the presence of suitable combinations of high and low levels at neighbors, e.g., a high value on one side and low values on several other sides; but this requires us to work with k -tuples of neighbors for $k > 2$.] At the same time, low cornerity values at the neighbors of a pixel should reinforce a high value at the pixel, just as in the case of peak detection. Analogous methods can be used to detect corners on ideal borders or curves represented by chain codes; here again, cornerity is

reinforced by the presence of neighboring slopes that differ sharply, and by low neighbor cornerity [24].

In general, we can employ a cooperative approach to detect matches of a given template with the image by considering the template as composed of pieces; detecting matches with the pieces by some conventional method; and reinforcing a match to a given piece based on the occurrence of matches to the other pieces in approximately the correct relative positions. This approach is preferable to straightforward matching of the entire template for two reasons: it has lower computational cost, and it is less sensitive to geometrical distortion. Note that in this reinforcement process, if there are matches to a given piece in several neighboring positions, we can use the best one, but we should not sum their influences, since only one of them can be correct; thus a reinforcement rule using the max, rather than the sum, is more appropriate here [25, 26].

3.3. Processes involving multiple properties or multiple images

A more complex class of cooperating processes can be used to assign two or more interrelated sets of labels to the pixels in an image. As an example, consider the gray level and edge labelling processes discussed in Sections 3.1 and 3.2. These two sets of labels are not independent; for example, the gray levels on the light side of an edge are more likely to belong to a light than to a dark class, and vice versa. Thus we can design a compound cooperating process in which both sets of labels interact; such processes should yield better results than if we use either of the individual processes alone [27, 28].

The gray level at a given pixel of an image is the resultant of several 'intrinsic' properties at the corresponding point of the scene, including illumination, reflectivity, and surface slope. It is impossible to separate the effects of these factors by examining the pixels individually; but one can attempt to separate them using a cooperative process, based on assumptions about how the factors vary from point to point. For example, let us assume that the scene is composed of regions over which the intrinsic properties are constant. It may then be possible to determine which property is changing at a given edge, by analyzing the gray level variations at the edge. If this can be done, we can try to estimate the property values cooperatively, by hypothesizing initial values and letting like reinforce like except across edges [29].

Finally, we consider cooperating processes that involve more than one image. Given two images of the same scene, taken at different times or from different positions, it will not be possible in general to register the images globally, since parts of the scene may have moved, or their projections on the image may have shifted by different amounts because they are at different distances from the sensor. However, we can try to match pieces of one image with pieces of the other to determine a piecewise correspondence; from the variations in this correspondence we can then estimate the motion or distance information. The accuracy of these estimates can be enhanced using a

cooperative approach, if we assume that the scene is made up of parts each having a uniform motion or distance; the approach is analogous to that used in Section 3.1 (see [30, 31]).

3.4. Region-level processes

Cooperating processes can also be used to assign labels to windows or regions of an image, or to detect configurations of regions that match given models. Such processes are briefly discussed in the following paragraphs.

As a simple example, suppose that we have broken up an image into windows, and want to classify the textures in the windows. If we use small windows, the classifications become unreliable; but if we use large ones, border effects become a major factor, since it is hard to classify windows that overlap two or more differently textured windows. One solution is to use small windows, and adjust the classifications (or the feature values) cooperatively, based on those of neighboring windows, in such a way that windows belonging to different regions (most likely) do not influence one another; this is analogous to the cooperative approach to pixel classification described in Section 3.1 (see [32]).

More generally, suppose that we have segmented an image into regions, and want to classify the regions, based on their geometrical or textural properties. If we know what pairs of classifications are possible for neighboring pairs of regions in given relative positions, we can use this knowledge to cooperatively adjust the label possibilities or confidences. For example: (1) In labelling the edges in a blocks-world scene as convex, concave, or occluding, we can use the constraints imposed when the edges meet at junction [1]. (2) In assigning regions in an indoor scene to classes such as 'door', 'doorknob', 'wall', and 'light switch', the light switch label is reinforced, and the doorknob label weakened, by the wall label on a surrounding region, and vice versa for the door label on a surrounding region [33].

Matching a configuration of regions to a given model is analogous to matching a piece of an image to a template. We can represent the regions, their properties, and their relationships by a 'scene graph' in which the nodes and arcs are labelled by property or relation names (and values). The model can be similarly represented by a labelled graph, and we can then attempt to find occurrences of the model graph as a subgraph of the scene graph. Just as in the template case, this can be done cooperatively by finding scene graph nodes that match model graph nodes, and reinforcing matches for which the proper neighboring nodes are present [34, 35].

4. Issues

Relaxation processes have proved very useful for deriving relatively unambiguous labellings of image or scene parts at a variety of levels. The design and control of such processes, however, are not as yet well understood. Given a

labelling task, how do we choose appropriate neighborhood and interaction models? (In other words, how do we represent our knowledge about the given problem domain in the form of an iterative local process?) Given such a process, how many times should it be iterated, and how should its performance be evaluated? In this section we briefly review some of the approaches that have been proposed to these problems of knowledge representation and control in relaxation processes.

4.1. Knowledge representation

As mentioned in Section 2.2, for discrete relaxation processes the interaction model is defined by a set of compatible label pairs; but for fuzzy labellings, the compatibility relation must be quantitative. It can be defined, for example, by specifying a 'compatibility coefficient' for each pair of labels on each pair of neighboring parts. These coefficients can be defined in a problem-specific manner; for example, the compatibility between two given edge or line directions at a pair of neighboring pixels could be taken as inversely proportional to the bending energy required to bend a spline so that it changes direction in the given way.

Another possibility is to define compatibilities on probabilistic grounds. Consider the probability ratio $r(d,d') = p(d,d')/p(d)p(d')$, where the numerator is the joint probability of the pair of labels (d,d') on the given pair of neighboring objects, and the terms in the denominator are the prior probabilities of the two labels. Intuitively, if d and d' are compatible, $p(d,d')$ should be greater than $p(d)p(d')$; if d and d' are independent, they should be equal; and if d and d' are incompatible, $p(d,d')$ should be less than $p(d)p(d')$. Thus we have $r(d,d') > 1$, $= 1$, and < 1 iff d and d' are compatible, independent, or incompatible, respectively. If we want compatibilities that lie in the range $[-1, 1]$, we can use $\log r$ rather than r ; this is positive, zero, or negative according to whether d and d' are compatible, independent, or incompatible. (The log does not automatically lie in the range $[-1, 1]$; if we want it to, it must be truncated and rescaled.) Note that $\log r$ is the *mutual information* of the pair of labels d, d' . The probabilities can be estimated by counting occurrences of d and d' , and joint occurrences of both. The use of mutual information to define compatibilities is suggested in [36]. If we drop the restriction that the compatibilities lie in the range $[-1, 1]$, we can use r itself, rather than $\log r$, as a compatibility function. In fact, in a Bayesian approach to relaxation developed by Peleg, the compatibility coefficients turn out to be the r 's (see [37]).

4.2. Control

A second critical question concerns the control of relaxation processes: when should the iteration be stopped? How can its progress be evaluated?

For a discrete relaxation process, termination criteria are straightforward to formulate and justify. For example, when binary (or higher-order) relations are used as a knowledge representation, then the process terminates when no further labels can be discarded from any picture part. At this point, each label at each picture part (if any remain) has a consistent label at every neighboring picture part. Or, if logic statements are used as a knowledge representation, then the process terminates when no new inferences can be formed. In both cases, the destination of the process is a consistent labelling, the notion of consistency is well-defined and it is straightforward to prove that the relaxation process has as its 'fixed point' a consistent labelling.

For a probabilistic relaxation process, the situation is more complicated. One possible approach is that the relaxation process should be iterated until the probability densities for each picture part converge. There are, however, both practical and theoretical disadvantages to this approach:

- (a) In practice, relaxation processes often converge to results which are quite poor, even though the first several iterations lead to significant improvements.
- (b) There are very few theoretical results concerning convergence, and these simply characterize sufficient conditions for convergence, rather than necessary conditions [38]. Moreover, the limit points have not been characterized as solutions to a well-defined problem, except in some specific cases [39].

Various criteria have been proposed for evaluating the performance of relaxation processes [40], but none of them seem to be satisfactory. Convergence (i.e., decrease in rate of change) is not an acceptable criterion, since the limit point may not be a desirable labelling. Unambiguity (e.g., low entropy) is also not acceptable, since there are many unambiguous labelings, most of which are highly inconsistent with the given initial labelling. Combinations of these criteria might be used [41], but these are also subject to similar objections [42]. A more promising approach uses a composite criterion for evaluating a labelling based on its consistency with both the initial labelling and the model (i.e., the compatibilities) [43]. This area is still the subject of active research.

5. Concluding Remarks

Relaxation processes have potential speed advantages because they can be implemented in parallel (hardware permitting). They have been successfully applied to a wide variety of labelling problems by a growing number of investigators; our survey makes no claim to completeness. In spite of these successes, little is as yet known about the design and control of these processes. However, a number of promising approaches to their theoretical formulation are being pursued, and it is hoped that a deeper understanding of their nature will soon be achieved.

ACKNOWLEDGMENT

The support of the National Science Foundation under Grants MCS-76-23763 to the University of Maryland, and ENG-79-04037 to the University of Texas, is gratefully acknowledged, as is the help of Eleanor Waters in preparing this paper. The authors also wish to thank Shmuel Peleg and Michael O. Shneier for helpful discussions and comments.

REFERENCES

1. Waltz, D., Understanding line drawings of scenes with shadows, in: Winston, P.H. (Ed.), *The Psychology of Computer Vision* (McGraw-Hill, New York, 1975) 19-91.
2. Rosenfeld, A., Hummel, R. and Zucker, S.W., Scene labelling by relaxation operations, *IEEE Trans. Systems, Man, Cybernetics* 6 (1976) 420-433.
3. Haralick, R.M., Davis, L.S., Milgram, D.L. and Rosenfeld, A., Reduction operators for constraint satisfaction, *Information Sci.* 14 (1978) 199-219.
4. Haralick, R.M. and Shapiro, L.G., The consistent labelling problem: Part I, *IEEE Trans. Pattern Analysis Machine Intelligence* 1 (1979) 173-183; Part II, ibid. 2 (1980) 193-203.
5. Davis, L.S., A logic model for constraint propagation, Tech. Rept. TR-137, Computer Sciences Dept., Univ. of Texas (1980).
6. Davis, L.S. and Rosenfeld, A., Hierarchical relaxation for waveform parsing, in: Hanson, A. and Riseman, E. (Eds.), *Computer Vision Systems* (Academic Press, New York, 1978) 101-109.
7. Hayes, K.C. Jr., Reading handwritten words using hierarchical relaxation, TR-783, Computer Science Center, University of Maryland, College Park, MD (July 1979) Abridged version to appear in *Computer Graphics Image Processing*.
8. Davis, L.S. and Henderson, T.C., Hierarchical constraint processes for shape analysis, TR-115, Computer Sciences Dept., University of Texas, Austin, TX (November 1979).
9. Davis, L.S. and Rosenfeld, A., Noise cleaning by iterated local averaging, *IEEE Trans. Systems, Man, Cybernetics* 8 (1978) 705-710.
10. Tomita, F. and Tsuji, S., Extraction of multiple regions by smoothing in selected neighborhoods, *IEEE Trans. Systems, Man, Cybernetics* 7 (1977) 107-109.
11. Nagao, M. and Matsuyama, T., Edge preserving smoothing, *Computer Graphics Image Processing* 9 (1979) 394-407.
12. Lev, A., Zucker, S.W. and Rosenfeld, A., Iterative enhancement of noisy images, *IEEE Trans. Systems, Man, Cybernetics* 7 (1977) 435-442.
13. Scher, A., Velasco, F.R.D. and Rosenfeld, A., Some new image smoothing techniques, *IEEE Trans. Systems, Man, Cybernetics* 10 (1980) 153-158.
14. Eklundh, J.O. and Rosenfeld, A., Image smoothing based on neighbor linking, *IEEE Trans. Pattern Analysis Machine Intelligence* 3 (1981) in press.
15. Eklundh, J.O., Yamamoto, H. and Rosenfeld, A., A relaxation method in multispectral pixel classification, *IEEE Trans. Pattern Analysis Machine Intelligence* 2 (1980) 72-75.
16. Rosenfeld, A. and Smith, R.C., Thresholding using relaxation, *IEEE Trans. Pattern Analysis Machine Intelligence* 3 (1981) in press.
17. Haralick, R.M. and Watson, L., A facet model for image data, *Proc. IEEE Conf. Pattern Recognition Image Processing* (August 1979) 489-497.
18. Davis, L.S. and Rosenfeld, A., Iterative histogram modification, *IEEE Trans. Systems, Man, Cybernetics* 8 (1978) 300-302.
19. Peleg, S., Iterative histogram modification, 2, *IEEE Trans. Systems, Man, Cybernetics* 8 (1978) 555-556.
20. Eberlein, R., An iterative gradient edge detection algorithm, *Computer Graphics Image Processing* 5 (1976) 245-253.

COOPERATING PROCESSES FOR LOW-LEVEL VISION

21. Zucker, S.W., Hummel, R.A. and Rosenfeld, A., An application of relaxation labeling to line and curve enhancement, *IEEE Trans. Computers* 26 (1977) 394-403, 922-929.
22. Schachter, B.J., Lev, A., Zucker, S.W. and Rosenfeld, A., An application of relaxation methods to edge reinforcement, *IEEE Trans. Systems, Man, Cybernetics* 7 (1977) 813-816.
23. Hanson, A.R. and Riseman, E.M., Segmentation of natural scenes, in: Hanson, A. and Riseman, E. (Eds.), *Computer Vision Systems* (Academic Press, New York, 1978) 129-163.
24. Davis, L.S. and Rosenfeld, A., Curve segmentation by relaxation labelling, *IEEE Trans. Computers* 26 (1977) 1053-1057.
25. Davis, L.S. and Rosenfeld, A., An application of relaxation labelling to spring-loaded template matching, *Proc. 3rd Int. Joint Conf. on Pattern Recognition* (November 1976) 591-597.
26. Ranade, S. and Rosenfeld, A., Point pattern matching by relaxation, *Pattern Recognition* 12 (1980) 269-275.
27. Zucker, S.W. and Hummel, R.A., Toward a low-level description of dot clusters: Labelling edge, interior, and noise points, *Computer Graphics Image Processing* 9 (1979) 213-233.
28. Danker, A. and Rosenfeld, A., Blob extraction by relaxation, *IEEE Trans. Pattern Analysis Machine Intelligence* 3 (1981) in press.
29. Barrow, H.G. and Tenenbaum, J.M., Recovering intrinsic scene characteristics from images, in: Hanson, A. and Riseman, E. (Eds.), *Computer Vision Systems* (Academic Press, New York, 1978) 3-26.
30. Marr, D. and Poggio, T., Cooperative computation of stereo disparity, *Science* 194 (1976) 283-287.
31. Barnard, S.T. and Thompson, W.B., Disparity analysis of images, *IEEE Trans. Pattern Analysis Machine Intelligence* 2 (1980) 333-346.
32. Hong, T.H., Wu, A.Y. and Rosenfeld, A., Feature value smoothing as an aid in texture analysis, *IEEE Trans. Systems, Man, Cybernetics* 10 (1980).
33. Barrow, H.G. and Tenenbaum, J.M., MSYS: A system for reasoning about scenes, TN-121, Artificial Intelligence Center, SRI, Inc., Menlo Park, CA (April 1976).
34. Kitchen, L. and Rosenfeld, A., Discrete relaxation for matching relational structures, *IEEE Trans. Systems, Man, Cybernetics* 9 (1979) 869-874.
35. Kitchen, L., Relaxation applied to matching quantitative relational structures, *IEEE Trans. Systems, Man, Cybernetics* 10 (1980) 96-101.
36. Peleg, S. and Rosenfeld, A., Determining compatibility coefficients for curve enhancement relaxation processes, *IEEE Trans. Systems, Man, Cybernetics* 8 (1978) 548-555.
37. Peleg, S., A new probabilistic relaxation scheme, *IEEE Trans. Pattern Analysis Machine Intelligence* 2 (1980) 362-369.
38. Zucker, S.W., Leclerc, Y.G. and Mohammed, J.L., Continuous relaxation and local maxima section—conditions for equivalence, *Proc. 6th Int. Joint Conf. on Artificial Intelligence* (August 1979) 1014-1016.
39. Ullman, S., Relaxation and constrained optimization by local processes, *Computer Graphics Image Processing* 10 (1979) 115-125.
40. Fekete, G., Eklundh, J.O. and Rosenfeld, A., Relaxation: evaluation and applications, *IEEE Trans. Pattern Analysis Machine Intelligence* 3 (1981) in press.
41. Faugeras, O. and Berthod, M., Scene labelling: an optimization approach, *Proc. IEEE Conf. Pattern Recognition Image Processing* (August 1979) 318-326.
42. Peleg, S. and Rosenfeld, A., A note on the evaluation of probabilistic labellings, TR-805, Computer Science Center, University of Maryland, College Park, MD (August 1979).
43. Peleg, S., Monitoring relaxation algorithms using labelling evaluation, TR-842, Computer Science Center, University of Maryland, College Park, MD (December 1979).

Received October 1980

Photometric method for determining surface orientation from multiple images

Robert J. Woodham

Department of Computer Science

University of British Columbia

2075 Wesbrook Mall

Vancouver, B.C., Canada

V6T 1W5

Abstract. A novel technique called photometric stereo is introduced. The idea of photometric stereo is to vary the direction of incident illumination between successive images, while holding the viewing direction constant. It is shown that this provides sufficient information to determine surface orientation at each image point. Since the imaging geometry is not changed, the correspondence between image points is known *a priori*. The technique is photometric because it uses the radiance values recorded at a single image location, in successive views, rather than the relative positions of displaced features.

Photometric stereo is used in computer-based image understanding. It can be applied in two ways. First, it is a general technique for determining surface orientation at each image point. Second, it is a technique for determining object points that have a particular surface orientation. These applications are illustrated using synthesized examples.

Key Words: bidirectional reflectance distribution function (BRDF), image processing, imaging geometry, incident illumination, photometric stereo, reflectance map, surface orientation.

Optical Engineering 19:1:139-144 (January/February 1980).

"Photometric Method for Determining Surface Orientation from Multiple Images" by R.J. Woodham from *Optical Engineering*, Volume 19, Number 1, January/February 1980, pages 139-144. Copyright © 1980 Society of Photo-Optical Instrumentation Engineers, reprinted with permission.

I. INTRODUCTION

Work on computer-based image understanding has led to a need to model the imaging process. One aspect of this concerns the geometry of image projection. Less well understood is the radiometry of image formation. Relating the radiance values recorded in an image to object shape requires a model of the way surfaces reflect light.

A reflectance map is a convenient way to incorporate a fixed scene illumination, surface reflectance and imaging geometry into a single model that allows image intensity to be written as a function of surface orientation. This function is not invertible since surface orientation has two degrees of freedom and image intensity provides only one measurement. Local surface shape cannot, in general, be determined from the intensity value recorded at a single image point. In order to determine object shape, additional information must be provided.

This observation has led to a novel technique called photometric stereo in which surface orientation is determined from two or more images. Traditional stereo techniques determine range by relating two images of an object viewed from different directions. If the correspondence between picture elements is known, then distance to the object can be calculated by triangulation. Unfortunately, it is difficult to determine this correspondence. The idea of photometric stereo is to vary the direction of the incident illumination between successive images, while holding the viewing direction constant. It is shown that this provides sufficient information to determine surface orientation at each image point. Since the imaging geometry is not changed, the correspondence between image points is known *a priori*. The technique is photometric because it uses the radiance values recorded at a single image location, in successive views, rather than the relative positions of displaced features.

Original manuscript 5015 received Feb. 20, 1979.

Revised manuscript received March 12, 1979.

Accepted for publication July 18, 1979.

This paper is a revision of a paper presented at the SPIE seminar on Image Understanding Systems & Industrial Applications, Aug. 30-31, 1978, San Diego, which appears in SPIE Proceedings Vol. 155.

II. THE REFLECTANCE MAP

The fraction of light reflected by an object surface in a given direction depends upon the optical properties of the surface material, the surface microstructure and the spatial and spectral distribution and state of polarization of the incident illumination. For many surfaces, the fraction of the incident illumination reflected in a particular direction depends only on the surface orientation. The reflectance characteristics of such a surface can be represented as a function $\phi(i, e, g)$ of the three angles i , e and g defined in Figure 1. These are called, respectively, the *incident*, *emergent* and *phase* angles. The angles i and e are defined relative to a local surface normal. $\phi(i, e, g)$ determines the ratio of surface radiance to irradiance measured per unit surface area, per unit solid angle, in the direction of the viewer. The reflectance function $\phi(i, e, g)$ defined here is related to the bidirectional reflectance distribution function (BRDF) defined by the National Bureau of Standards.¹

Image forming systems perform a perspective transformation, as illustrated in Figure 2(a). If the size of the objects in view is small compared to the viewing distance, then the perspective projection can be approximated as an orthographic projection, as illustrated in Figure 2(b). Consider an image forming system that performs an orthographic projection. To standardize the imaging geometry, it is convenient to choose a coordinate system such that the viewing direction is aligned with the negative z -axis. Also, assume appropriate scaling of the image plane such that object point (x, y, z) maps onto image point (u, v) where $u = x$ and $v = y$. With these assumptions, image coordinates (x, y) and object coordinates (x, y) can be referred to interchangeably.

If the equation of an object surface is given explicitly as

$$z = f(x, y)$$

then a surface normal is given by the vector:

$$\left[\frac{\partial f(x, y)}{\partial x}, \frac{\partial f(x, y)}{\partial y}, -1 \right]$$

If parameters p and q are defined by:

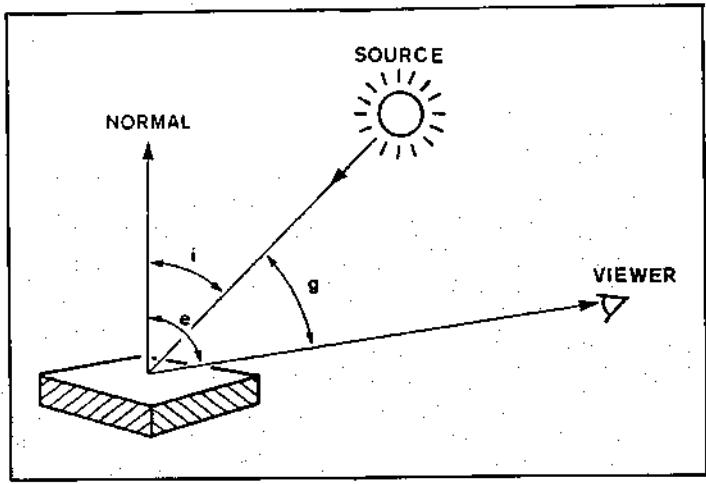


Figure 1. Defining the three angles i , e and g . The incident angle i is the angle between the incident ray and the surface normal. The emergent angle e is the angle between the emergent ray and the surface normal. The phase angle g is the angle between the incident and emergent rays.

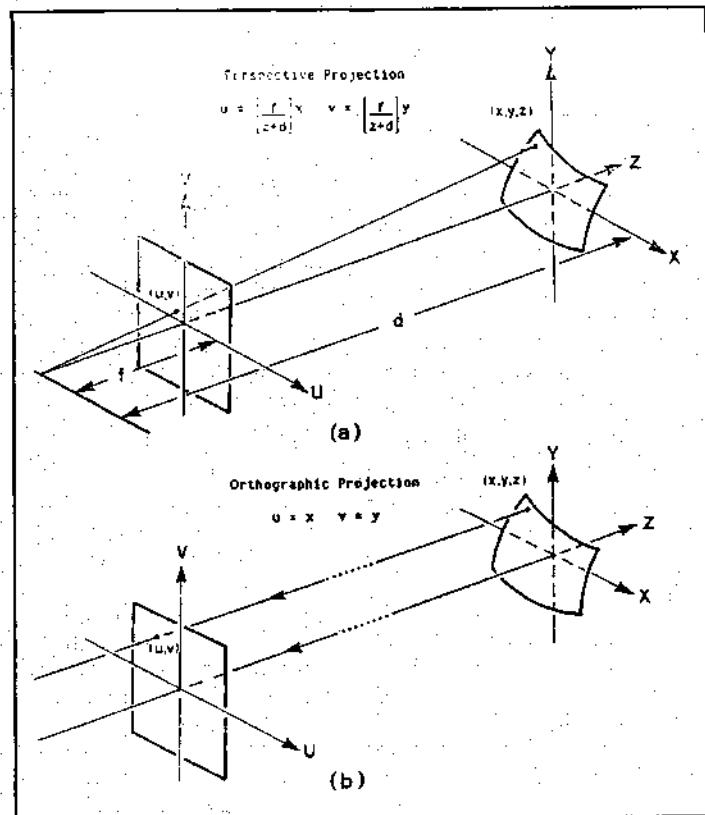


Figure 2. Characterizing image projections. (a) illustrates the well-known perspective projection. [Note: to avoid image inversion, it is convenient to assume that the image plane lies in front of the lens rather than behind it.] For objects that are small relative to the viewing distance, the image projection can be modeled as the orthographic projection illustrated in (b). In an orthographic projection, the focal length f is infinite so that all rays from object to image are parallel.

$$p = \frac{\partial f(x,y)}{\partial x} \text{ and } q = \frac{\partial f(x,y)}{\partial y}$$

then the surface normal can be written as $[p,q,-1]$. The quantity (p,q) is called the *gradient* of $f(x,y)$ and *gradient space* is the two-dimensional space of all such points (p,q) . Gradient space is a con-

venient way to represent surface orientation. It has been used in scene analysis.² In image analysis, it is used to relate the geometry of image projection to the radiometry of image formation.³ This relation is established by showing that image intensity can be written explicitly as a function of gradient coordinates p and q .

An ideal imaging device produces image irradiances proportional to scene irradiances. In an orthographic projection, the viewing direction, and hence the phase angle g , is constant for all object points. Thus, for a fixed light source and viewer geometry, the ratio of scene radiance to irradiance depends only on gradient coordinates p and q . Further, suppose each object surface element receives the same incident radiance. Then, the scene radiance, and hence image intensity, depends only on gradient coordinates p and q .

The *reflectance map* $R(p,q)$ determines image intensity as a function of p and q . A reflectance map captures the surface reflectance of an object material for a particular light source, object surface and viewer geometry. Reflectance maps can be determined empirically, derived from phenomenological models of surface reflectivity or derived from analytic models of surface microstructure.

In this paper, it will be assumed that image projection is orthographic and that incident illumination is given by a single distant point source. Extended sources can be modeled as the superposition of single sources. The reflectance map can be extended to incorporate spatially varying irradiance and perspective. A formal analysis of the relation between the reflectance map and the bidirectional reflectance distribution function (BRDF) has been given.⁴

Expressions for $\cos(i)$, $\cos(e)$ and $\cos(g)$ can be derived using normalized dot products of the surface normal vector $[p,q,-1]$, the vector $[p_s, q_s, -1]$ which points in the direction of the light source and the vector $[0,0,-1]$ which points in the direction of the viewer. One obtains:

$$\cos(i) = \frac{1 + pp_s + qq_s}{\sqrt{1 + p^2 + q^2} \sqrt{1 + p_s^2 + q_s^2}}$$

$$\cos(e) = \frac{1}{\sqrt{1 + p^2 + q^2}}$$

$$\cos(g) = \frac{1}{\sqrt{1 + p_s^2 + q_s^2}}$$

These expressions can be used to transform an arbitrary surface reflectance function $\phi(i,e,g)$ into a reflectance map $R(p,q)$.

One simple idealized model of surface reflectance is given by:

$$\phi_a(i,e,g) = \rho \cos(i)$$

This reflectance function corresponds to the phenomenological model of a perfectly diffuse (lambertian) surface which appears equally bright from all viewing directions. Here, ρ is a reflectance factor and the cosine of the incident angle accounts for the foreshortening of the surface as seen from the source. The corresponding reflectance map is given by:

$$R_a(p,q) = \frac{\rho (1 + pp_s + qq_s)}{\sqrt{1 + p^2 + q^2} \sqrt{1 + p_s^2 + q_s^2}}$$

A second reflectance function, similar to that of materials in the maria of the moon and rocky planets, is given by:

$$\phi_b(i,e,g) = \frac{\rho \cos(i)}{\cos(e)}$$

This reflectance function corresponds to the phenomenological model of a surface which reflects equal amounts of light in all directions. The cosine of the emergent angle accounts for the foreshortening of the surface as seen from the viewer. The corresponding reflectance map is given by:

$$R_b(p,q) = \frac{\varrho(1 + pp_s + qq_s)}{\sqrt{1 + p_s^2 + q_s^2}}$$

It is convenient to represent $R(p,q)$ as a series of iso-brightness contours in gradient space. Figure 3 and Figure 4 illustrate the two simple reflectance maps $R_a(p,q)$ and $R_b(p,q)$, defined above, for the case $p_s = 0.7$, $q_s = 0.3$ and $\varrho = 1$.

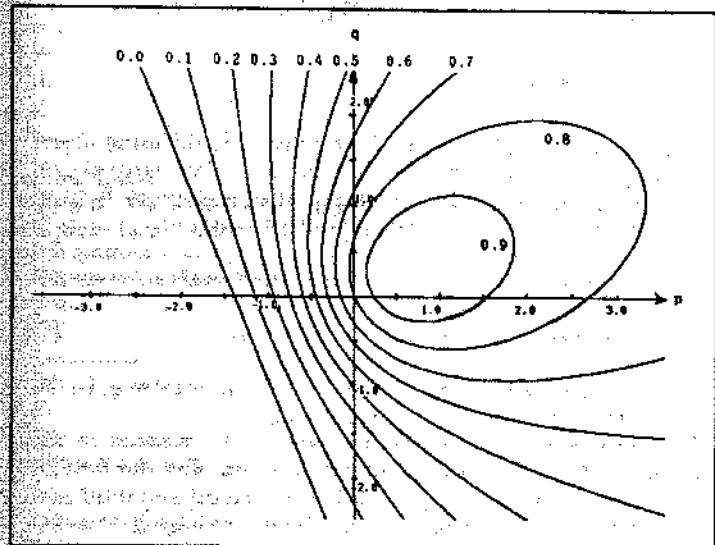


Figure 3. The reflectance map $R_a(p,q)$ for a Lambertian surface illuminated from gradient point $p_s = 0.7$ and $q_s = 0.3$ (with $\varrho = 1.0$). The reflectance map is plotted as a series of contours spaced 0.1 units apart.

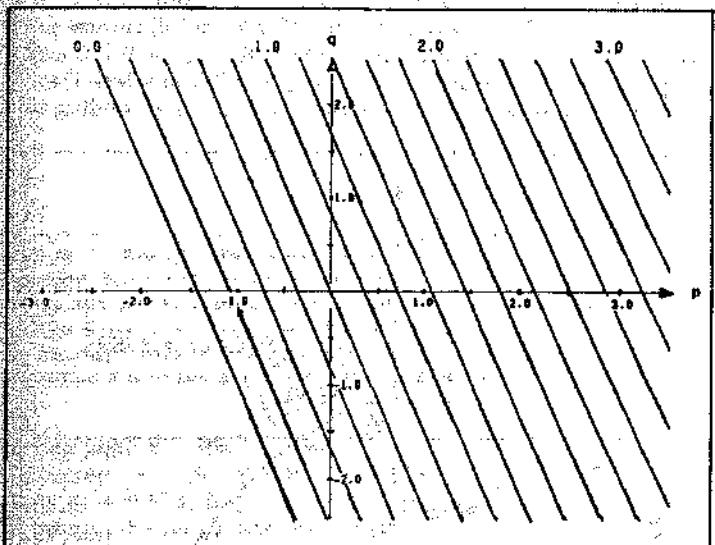


Figure 4. The reflectance map $R_b(p,q)$ for a surface illuminated from gradient point $p_s = 0.7$ and $q_s = 0.3$ (with $\varrho = 1.0$). The reflectance map is plotted as a series of contours spaced 0.2 units apart.

Reflectance map techniques

Using the reflectance map, the basic equation describing the image-forming process can be written as:

$$I(x,y) = R(p,q) \quad (1)$$

One idea is to use Eq. (1) directly to generate shaded images of surfaces. This has obvious utility in computer graphics applications including hill-shading for automated cartography⁵ and video input for a flight simulator.⁶ Synthesized imagery can be registered to real imagery to align images with surface models. This technique has been used to achieve precise alignment of Landsat imagery with digital terrain models.⁷

Equation (1) can also be used in image analysis to determine object shape from image intensity. Equation (1) is a nonlinear first-order partial differential equation. Direct solution is tedious.⁸ More generally, one can think of Eq. (1) as one equation in the two unknowns p and q . Determining object shape from image intensity is difficult because Eq. (1) is underdetermined. In order to calculate object shape, additional assumptions must be invoked.

Recent work has helped to make these assumptions explicit. For certain materials, such as the material of the maria of the moon, special properties of surface reflectance simplify the solution.^{3,8,9} Other methods for determining object shape from image intensity embody assumptions about surface curvature.^{9,10} Simple surfaces have been proposed for use in computer aided design.¹¹ When properties of surface curvature are known *a priori*, they can be exploited in image analysis.¹² This is useful, for example, in industrial inspection since there are often constraints on surface curvature imposed by the drafting techniques available for part design and by the fabrication processes available for part manufacture.¹³

Reflectance map techniques deepen our understanding of what can and cannot be computed directly from image intensity. Photometric stereo is a novel reflectance map technique that uses two or more images to solve Eq. (1) directly.

III. PHOTOMETRIC STEREO

The idea of photometric stereo is to vary the direction of incident illumination between successive views, while holding the viewing direction constant. Suppose two images $I_1(x,y)$ and $I_2(x,y)$ are obtained by varying the direction of incident illumination. Since there has been no change in the imaging geometry, each picture element (x,y) in the two images corresponds to the same object point and hence to the same gradient (p,q) . The effect of varying the direction of incident illumination is to change the reflectance map $R(p,q)$ that characterizes the imaging situation.

Let the reflectance maps corresponding to $I_1(x,y)$ and $I_2(x,y)$ be $R_1(p,q)$ and $R_2(p,q)$ respectively. The two views are characterized by two independent equations:

$$I_1(x,y) = R_1(p,q) \quad (2)$$

$$I_2(x,y) = R_2(p,q) \quad (3)$$

Two reflectance maps $R_1(p,q)$ and $R_2(p,q)$ are required. But, if the phase angle ϱ is the same in both views (i.e., the direction of illumination is rotated about the viewing direction), then the two reflectance maps are rotations of each other.

For reflectance characterized by $R_b(p,q)$ above, Eqs. (2) and (3) are linear equations in p and q . If the reflectance factor ϱ is known, then two views are sufficient to determine surface orientation at each image point, provided the directions of incident illumination are not collinear in azimuth.

In general, Eqs. (2) and (3) are nonlinear so that more than one solution is possible. One idea would be to obtain a third image:

$$I_3(x,y) = R_3(p,q) \quad (4)$$

to overdetermine the solution.

For reflectance characterized by $R_a(p,q)$ above, three views are sufficient to uniquely determine both the surface orientation and the reflectance factor ϱ at each image point, as will now be shown.¹⁴ Let $\underline{I} = [I_1, I_2, I_3]'$ be the column vector of intensity values recorded at a point (x,y) in each of three views (' denotes vector transpose). Further, let

$$\underline{n}_1 = [n_{11}, n_{12}, n_{13}]'$$

$$\underline{n}_2 = [n_{21}, n_{22}, n_{23}]'$$

$$\underline{n}_3 = [n_{31}, n_{32}, n_{33}]'$$

be unit column vectors defining the three directions of incident illumination. Construct the matrix \underline{N} where

$$\underline{N} = \begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{bmatrix}$$

Let $\underline{n} = [n_1, n_2, n_3]'$ be the column vector corresponding to a unit surface normal at (x, y) . Then,

$$\underline{I} = \underline{e} \underline{N} \underline{n}$$

so that,

$$\underline{e} \underline{n} = \underline{N}^{-1} \underline{I}$$

provided the inverse \underline{N}^{-1} exists. This inverse exists if and only if the three vectors n_1 , n_2 and n_3 do not lie in a plane. In this case, the reflectance factor ρ and unit surface normal at (x, y) are given by:

$$\rho = |\underline{N}^{-1} \underline{I}|$$

and

$$\underline{n} = (1/\rho) \underline{N}^{-1} \underline{I}. \quad (5)$$

Unfortunately, since the sun's path across the sky is very nearly planar, this simple solution does not apply to outdoor images taken at different times during the same day.

Even when the simplifications implied by $R_a(p, q)$ and $R_b(p, q)$ above do not hold, photometric stereo is easily implemented. Initial computation is required to determine the reflectance map for each experimental situation. Once calibrated, however, photometric stereo can be reduced to simple table lookup and/or search operations. Photometric stereo is a practical scheme for environments, such as industrial inspection, in which the nature and position of the incident illumination is known or can be controlled.

The multiple images required for photometric stereo can be obtained by explicitly moving a single light source, by using multiple light sources calibrated with respect to each other or by rotating the object surface and imaging hardware together to simulate the effect of moving a single light source. The equivalent of photometric stereo can also be achieved in a single view by using multiple illuminations which can be separated by color.

Applications of photometric stereo

Photometric stereo can be used in two ways. First, photometric stereo is a general technique for determining surface orientation at each image point. For a given image point (x, y) , the equations characterizing each image can be combined to determine the corresponding gradient (p, q) .

Second, photometric stereo is a general technique for determining object points that have a particular surface orientation. This use of photometric stereo corresponds to interpreting the basic image-forming Eq. (1) as one equation in the unknowns x and y . For a given gradient (p, q) , the equations characterizing each image can be combined to determine corresponding object points (x, y) . This second use of photometric stereo is appropriate for the so-called industrial "bin-of-parts" problem. The location in an image of key object points is often sufficient to determine the position and orientation of a known object on a table or conveyor belt so that the object may be grasped by an automatic manipulator.

A particularly useful special case concerns object points whose surface normal directly faces the viewer (i.e., object points with $p = 0$ and $q = 0$). Such points form a unique class of image points whose intensity value is invariant under rotation of the illumina-

tion direction about the viewing direction. Object points with surface normal directly facing the viewer can be located without explicitly determining the reflectance map $R(p, q)$. The value of $R(0, 0)$ is not changed by varying the direction of illumination, provided only that the phase angle g is held constant.

These applications of photometric stereo will now be illustrated using a simple, synthesized example. Consider a sphere of radius r centered at the object space origin. The explicit representation of this object surface, corresponding to the viewing geometry of Figure 2(b), is given by:

$$z = f(x, y) = -\sqrt{r^2 - x^2 - y^2}. \quad (6)$$

The gradient coordinates p and q are determined by differentiating Eq. (6) with respect to x and y . One finds:

$$p = \frac{-x}{z} \text{ and } q = \frac{-y}{z}$$

Suppose that the sphere is made of a perfectly diffusing object material and is illuminated by a single distant point source at gradient point (p_s, q_s) . Then, the reflectance map is given by $R_a(p, q)$ above so that the corresponding synthesized image is:

$$I(x, y) = \begin{cases} 0 & \text{if } x^2 + y^2 > r^2 \\ \max(0, R_a(-x/z, -y/z)) & \text{otherwise} \end{cases} \quad (7)$$

Equation (7) generates image intensities in the range 0 to ρ . In the example below, $r = 60$ and $\rho = 1$.

Multiple images are obtained by varying the position of the light source. Consider three different positions. Let the first be $p_s = 0.7$ and $q_s = 0.3$ as in Figure 3. Let the second and third correspond to rotations of the light source about the viewing direction of -120° and $+120^\circ$ respectively (i.e., $p_s = -0.610$, $q_s = 0.456$ and $p_s = -0.090$, $q_s = -0.756$). Let the three reflectance maps be $R_1(p, q)$, $R_2(p, q)$ and $R_3(p, q)$. The phase angle g is constant in each case. Let the corresponding images generated by Eq. (6) be $I_1(x, y)$, $I_2(x, y)$ and $I_3(x, y)$.

First, consider image point $x = 15$, $y = 20$. Here, $I_1(x, y) = 0.942$, $I_2(x, y) = 0.723$ and $I_3(x, y) = 0.505$. Figure 5 illustrates the reflectance map contours $R_1(p, q) = 0.942$, $R_2(p, q) = 0.723$ and $R_3(p, q) = 0.505$. The point $p = 0.275$, $q = 0.367$ at which these three contours intersect determines the gradient corresponding to

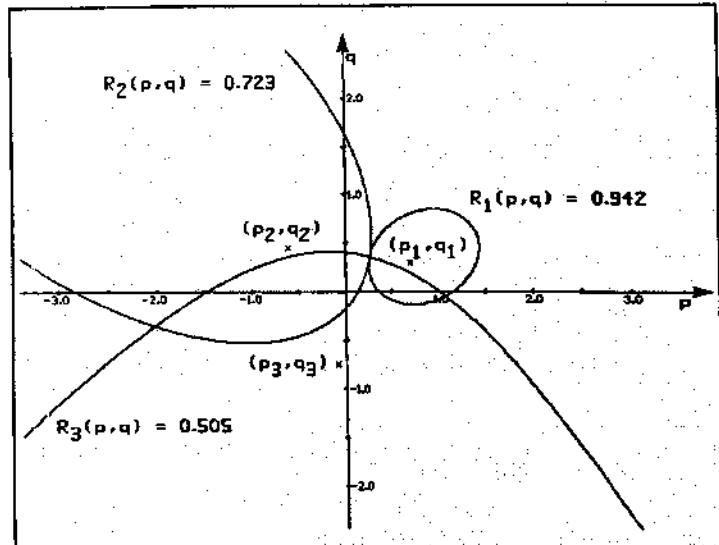


Figure 5. Determining the surface orientation (p, q) at a given image point (x, y) . Three reflectance map contours are intersected where each contour corresponds to the intensity value at (x, y) obtained from three separate images. $I_1(x, y) = 0.942$, $I_2(x, y) = 0.723$ and $I_3(x, y) = 0.505$.

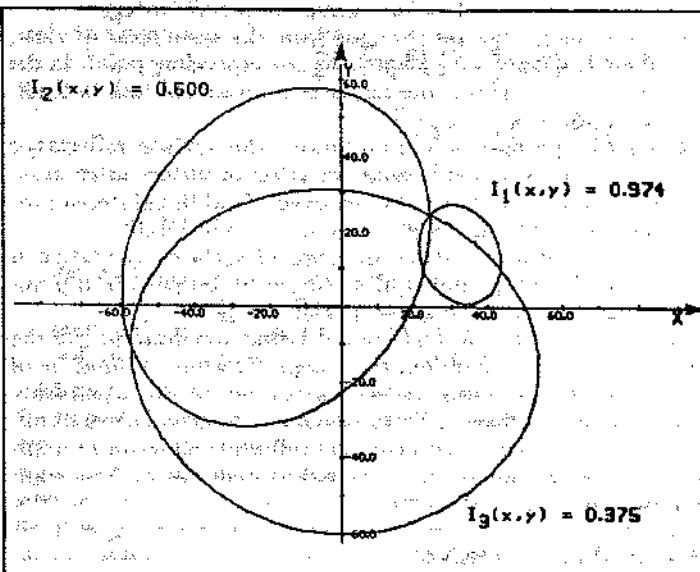


Figure 6. Determining image points (x,y) whose surface orientation is a given (p,q) . Three image intensity contours are intersected where each contour corresponds to the value at (p,q) obtained from three separate reflectance maps: $R_1(p,q) = 0.974$, $R_2(p,q) = 0.600$ and $R_3(p,q) = 0.375$.

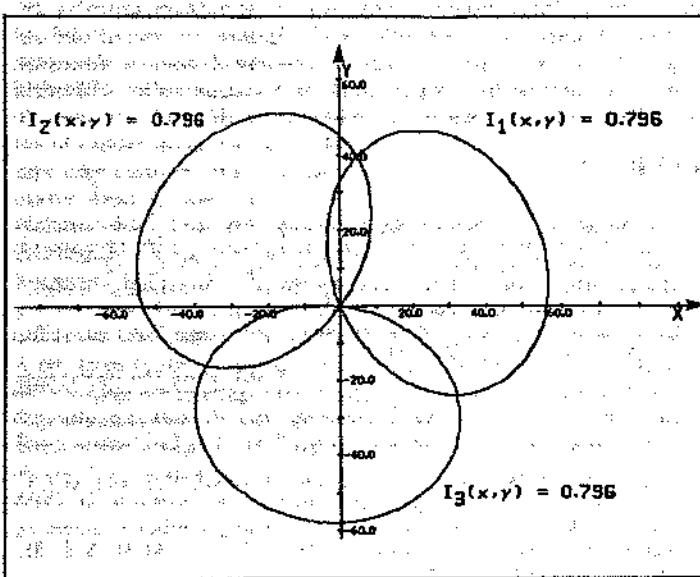


Figure 7. Determining image points (x,y) whose surface normal directly faces the viewer. Three image intensity contours are intersected where each contour corresponds to the value at $(0,0)$ obtained from three separate reflectance maps. Note that the reflectance map value at $(0,0)$ does not change with light source position, provided the phase angle g is held constant.

image point $x = 15$, $y = 20$.

Second, consider gradient point $p = 0.5$, $q = 0.5$. Here, $R_1(p,q) = 0.974$, $R_2(p,q) = 0.600$ and $R_3(p,q) = 0.375$. Figure 6 illustrates the image intensity contours $I_1(x,y) = 0.974$, $I_2(x,y) = 0.600$ and $I_3(x,y) = 0.375$. The point $x = 24.5$, $y = 24.5$ at which these three contours intersect determines an object point whose gradient is $p = 0.5$, $q = 0.5$.

Finally, Figure 7 repeats the example given in Figure 6 but for the case $p = 0$, $q = 0$. Here, $R_1(p,q) = R_2(p,q) = R_3(p,q) = 0.796$. Object points with surface normal directly facing the viewer form a unique class of points whose image intensity is invariant for rotations of the light source about the viewing direction. The point $x = 0$, $y = 0$ at which these three contours intersect determines an object point with surface normal directly

facing the viewer. This result would hold even if the form of $R(p,q)$ is unknown.

Accuracy considerations

Photometric stereo is most accurate in regions of gradient space where the density of reflectance map contours is great and where the contours to be intersected are nearly perpendicular. Several factors influence the density and direction of reflectance map contours. The reflectance properties of the surface material play a role. Figures 3 and 4 illustrate the difference between two idealized materials viewed under identical conditions of illumination. In general, increasing the specular component of reflection will increase the density of contours in one region of gradient space at the expense of other regions. Using extended light sources rather than point sources will alter the shape and distribution of reflectance map contours. Imaging systems can be configured to exploit these facts.¹⁵

For a given surface material, the main determiner of accuracy is the choice of phase angle g . In photometric stereo, there is a trade-off to acknowledge. A large phase angle increases the density of reflectance map contours in illuminated portions of gradient space. At the same time, a large phase angle results in more of gradient space lying in shadow. A practical compromise must be arrived at for each application.

The relative positions of the light sources must also be considered. Figures 8 and 9 give some indication of the trade-off associated with light source position. In each case, reflectance is

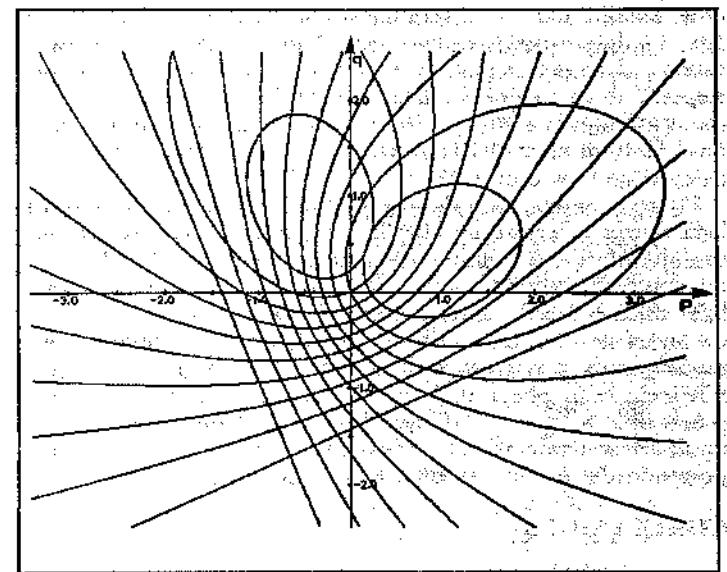


Figure 8. Superimposed reflectance maps $R_1(p,q)$ and $R_2(p,q)$ where $R_1(p,q)$ is $R_a(p,q)$ with $p_s = 0.7$, $q_s = 0.3$, $g = 1$ and $R_2(p,q)$ is $R_a(p,q)$ with $p_s = -0.3$, $q_s = 0.7$, $g = 1$. Each region indicates how an error in intensity measurement determines a corresponding error in the estimation of surface gradient (p,q) .

assumed to be characterized by $R_a(p,q)$ above. Figure 8 considers a two-source configuration in which the light source directions are separated by 90° in azimuth with respect to the viewer. Figure 8 superimposes reflectance map contours, spaced 0.1 units apart, for $R_1(p,q)$ and $R_2(p,q)$ where $R_1(p,q)$ is $R_a(p,q)$ with $p_s = 0.7$, $q_s = 0.3$, $g = 1$ and $R_2(p,q)$ is $R_a(p,q)$ with $p_s = -0.3$, $q_s = 0.7$, $g = 1$. Each region of Figure 8 corresponds to a region of equal measurement error. For example, if $I_1(x,y)$ is determined to lie between 0.4 and 0.5 and $I_2(x,y)$ is determined to lie between 0.5 and 0.6 then surface orientation can be determined to $\pm 6.8^\circ$ of its true value. This corresponds to an area of gradient space in the third quadrant where the error regions are small. Here, a measurement error of 1 gray level in 10 in each of $I_1(x,y)$ and

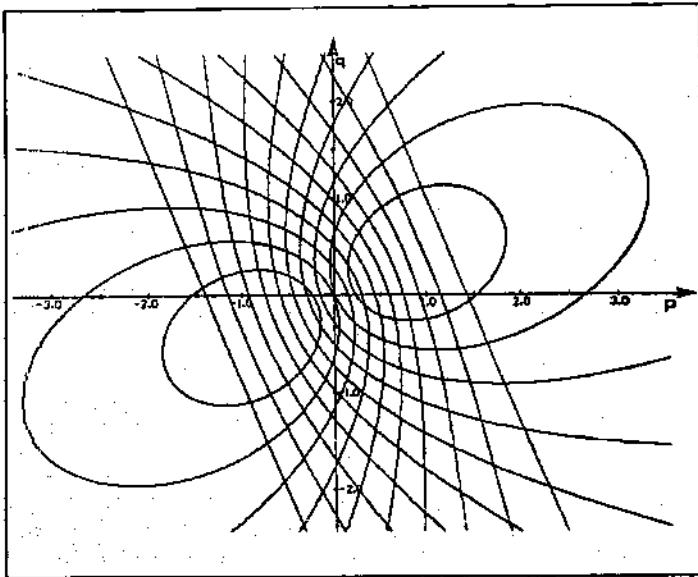


Figure 9. Superimposed reflectance maps $R_1(p,q)$ and $R_2(p,q)$ where $R_1(p,q)$ is $R_a(p,q)$ with $p_s = 0.7$, $q_s = 0.3$, $\rho = 1$ and $R_2(p,q)$ is $R_b(p,q)$ with $p_s = -0.7$, $q_s = -0.3$, $\rho = 1$. Each region indicates how an error in intensity measurement determines a corresponding error in the estimation of surface gradient (p,q) .

$I_2(x,y)$ constrains surface orientation to within $\pm 6.8^\circ$. On the other hand, if $I_1(x,y)$ is determined to lie between 0.9 and 1.0 and $I_2(x,y)$ is determined to lie between 0.5 and 0.6 then surface orientation can be determined to $\pm 25.8^\circ$ of its true value. This corresponds to an area of gradient space in the first quadrant where the error regions are large. Here, a measurement error of 1 gray level in 10 in each of $I_1(x,y)$ and $I_2(x,y)$ only constrains surface orientation to within $\pm 25.8^\circ$.

Figure 9 repeats the example of Figure 8 but with the second light source separated by 180° in azimuth from the first. In this configuration, error regions are smallest in the second and fourth quadrants of gradient space. Combinations using more than two light sources can be arranged to achieve a desired overall accuracy. One idea is to choose four directions of illumination, spaced evenly in azimuth with respect to the viewer and having a relatively large phase angle θ .¹³ In such a configuration, most points of interest are illuminated by at least three independent sources and contours can be selected to intersect which are nearly perpendicular and where error regions are small.

CONCLUSIONS

Surface orientation can be determined from the image intensities obtained under a fixed imaging geometry but with varying lighting conditions. Photometric methods for determining surface orientation can be considered complementary to methods based on the identification of corresponding points in two images taken from different viewpoints:

1. Traditional stereo allows the accurate determination of distances to objects. Photometric stereo is best when the surface gradient is to be found.
2. Traditional stereo works well on rough surfaces with discontinuities in surface orientation. Photometric stereo works best on smooth surfaces with few discontinuities.
3. Traditional stereo works well on textured surfaces with varying surface reflectance. Photometric stereo is best when applied to surfaces with uniform surface properties.

Photometric stereo does have some unique advantages:

1. Since the images are obtained from the same point of view, there is no difficulty identifying corresponding points in the two images. This is the major computational task in traditional stereo.
2. Under appropriate circumstances, the surface reflectance factor can be found because the effect of surface orientation on image intensity can be removed. Traditional stereo provides no such capability.
3. Describing object shape in terms of surface orientation is preferable in a number of situations to description in terms of range or altitude above a reference plane.

Photometric stereo depends on a detailed understanding of the imaging process. In addition, the imaging instrument must be of high caliber so that the gray levels produced can be dependably related to scene radiance. Fortunately, our understanding of image formation and the physics of light reflection has advanced sufficiently, and the quality of imaging devices is now high enough, to make this endeavor feasible.

ACKNOWLEDGMENTS

The author would like to thank Berthold K. P. Horn for his help and guidance. Mike Brady, Anni Bruss, Mark Lavin, Tomas Lozano-Perez, Alan Mackworth, David Marr and Patrick Winston provided useful comments and criticisms.

Work reported herein was conducted while the author was at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defence under Office of Naval Research Contract number N00014-75C-0643.

REFERENCES

1. Nicodemus, F. E., Richmond, J. C. and Hisa, J. J., "Geometrical considerations and nomenclature for reflectance," NBS Monograph 160, National Bureau of Standards, Washington, D. C., 1977.
2. Mackworth, A. K., "Interpreting pictures of polyhedral scenes," *Artificial Intelligence*, Vol. 4, pp. 121-137, 1973.
3. Horn, B. K. P., "Understanding image intensities," *Artificial Intelligence*, Vol. 8, pp. 201-231, 1977.
4. Horn, B. K. P. and Sjoberg, R. W., "Calculating the reflectance map," *Applied Optics*, Vol. 18, No. 11, pp. 1770-1779, 1979.
5. Horn, B. K. P., "Automatic hill-shading using the reflectance map," *Proc. Image Understanding Workshop*, Palo Alto, California, April 1979.
6. Strat, T. M., "Shaded perspective images of terrain," TR-463, M.I.T. A.I. Laboratory, Cambridge, Mass., 1978.
7. Horn, B. K. P. and Bachman, B. L., "Using synthetic images to register real images with surface models," *Comm. ACM*, Vol. 21, No. 11, pp. 914-924, 1978.
8. Horn, B. K. P., "Obtaining shape from shading information," *The Psychology of Computer Vision*, P. H. Winston (ed.), McGraw-Hill, pp. 115-155, 1975.
9. Rindfleisch, T., "Photometric method for lunar topography," *Photogrammetric Engineering*, Vol. 32, pp. 262-276, 1966.
10. Woodham, R. J., "A cooperative algorithm for determining surface orientation from a single view," *Proc. IJCAI-77*, pp. 635-641, Cambridge, Mass., 1977.
11. Huffman, D. A., "Curvature and creases: a primer on paper," *Proc. Conf. Computer Graphics, Pattern Recognition and Data Structures*, IEEE Pub. 75CH0981-1C, pp. 360-370, 1975.
12. Woodham, R. J., "Relating properties of surface curvature to image intensity," *Proc. IJCAI-79*, pp. 971-977, Tokyo, Japan, 1979.
13. Woodham, R. J., "Reflectance map techniques for analyzing surface defects in metal castings," TR-457, M.I.T. A.I. Laboratory, Cambridge, Mass., 1978.
14. Horn, B. K. P., "Three source photometry," (personal communication), 1978.
15. Ikeuchi, K. and Horn, B. K. P., "An application of the photometric stereo method," *Proc. IJCAI-79*, pp. 413-415, Tokyo, Japan, 1979. §

Region-Based Stereo Analysis for Robotic Applications

SURESH B. MARAPANE, STUDENT MEMBER, IEEE, AND MOHAN M. TRIVEDI, SENIOR MEMBER, IEEE

Reprinted from *IEEE Transactions on Systems, Man, and Cybernetics*, Volume 19, Number 6, November/December 1989, pages 1447-1464. Copyright © 1989 by The Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

Abstract — The development of a practical binocular stereo approach for the purpose of extracting depth information is considered. Accurate measurement of depth is required for many robotic tasks such as object recognition, manipulation, assembly, and obstacle avoidance. The ability to produce precise depth measurements over a wide range of distances and the passivity of the approach requiring simple and commonly available sensors, make binocular stereo an attractive approach. Yet rigid input requirements and high computational costs have prevented stereo approaches from being adopted for practical robotic applications. For effective use in this application domain, a stereo system must seek a balance between input flexibility, output sufficiency, and computational complexity. At the heart of a binocular stereo approach lies the task of stereo matching. The importance of semantic content and stability of the matching primitive is emphasized and the use of homogeneous regions as features in stereo matching is introduced. The region-based matching is more accurate than those using edge-based primitives since regions have higher discrimination capability. Also, region-based approaches are more efficient since there are fewer features to be matched. However, while the use of regions makes stereo matching more accurate, reliable, and efficient than edge-based matching, region-based matching processes typically yield coarse disparity maps. The generation of accurate and finer resolution disparity maps (and subsequently depth measurements) can be better accomplished using edge-based techniques. Both regions and edges play important, but somewhat complementary, roles in the binocular stereo process. It is, therefore, critical that an efficient and robust stereo system utilize the most appropriate set of primitives at each stage of the process. A hierarchical stereo approach that exploits and integrates the power of different primitives at appropriate stages of the stereo process is proposed. Several experiments to evaluate the performance of a region-based stereo matcher and a disparity and depth generation module are presented. It is shown that the approach is efficient, robust, and successful in generating depth measurements within ± 5 percent accuracy.

I. INTRODUCTION

MOST of the robotic systems currently in use are restricted in their operations due to either the lack of or the underutilization of external sensory feedback. These systems are inflexible and require a highly structured work environment. A robotic system has to possess sophisticated sensory mechanisms to autonomously perform tasks in a complex unstructured environment. Such sensor-driven robotic systems are more flexible, versatile,

Manuscript received October 30, 1988; revised April 16, 1989. This work was supported by the DOE's University Program in Robotics for Advanced Reactors (Universities of Florida, Michigan, Tennessee, and Texas, and the Oak Ridge National Laboratory) under Grant No. DOE DE-FG02-86NE37968.

The authors are with the Electrical and Computer Engineering Dept., Ferris Hall, University of Tennessee, Knoxville, TN 37996-2100.

IEEE Log Number 8930351.

reliable, dexterous, less expensive, and easier to teach [1], [2].

Advanced robotic systems will utilize many different sensory modalities such as vision, range, force, and touch in their operations. While vision is one of the most important sensory modalities used for sensing the operating environment, three-dimensional range information is also critical for the success of many robotic tasks. Range information is required for obstacle avoidance and navigation, for object recognition and pose determination, and for inspection, manipulation, and assembly of objects. Efficiency, accuracy, resolution, noise immunity, and robustness are some of the parameters that must be addressed in selecting a range-sensing approach. The specific robotic application considered in the present study is that of automatic inspection and manipulation in an "in-plant" environment. In particular, we discuss the development of a binocular stereo approach.

The ability to produce precise depth measurements over a wide range of distances and the passivity of the approach make binocular stereo an attractive tool for range sensing. At the heart of the binocular stereo approach lies the task of stereo matching. This task deals with the problem of identifying for each image feature associated with the projection of a particular surface structure in one image, the corresponding feature in the other image that is the projection of the same surface structure. The ability to solve this correspondence problem accurately, reliably, and efficiently, depends to a great extent on the types of features used in this matching. Most existing stereo approaches use edge-based primitives as features for matching. In this paper, we introduce the use of regions as the matching primitive.

Typically, in an image there are fewer regions than edge pixels (edgels) or edge segments. Regions possess a higher semantic content and, hence, higher discrimination capabilities than edge-based primitives. The reduction of the number of features to be matched, along with the region's high discrimination power, greatly decreases the number of mismatches, false targets, possible and makes the region-based matcher more accurate and reliable. Furthermore, fewer features to be matched makes establishing correspondence quite efficient. The high semantic content of the regions also facilitates the use of a wide range of representational schemes to describe the structural details in images. This allows utilization of both local and global infor-

mation in matching. Thus a region-based stereo matcher will yield a robust, accurate, and more globally consistent solution efficiently.

Yet while regions exhibit many desirable properties, they typically lack the capability to generate an accurate disparity map of fine resolution. Edge pixels on the other hand are better suited for the disparity generation stage. A stereo approach that utilizes a hierarchy of primitives such as regions, edge segments, and edge pixels in stereo matching and disparity generation should be able to address the limitations of purely region-based or edge-based approaches. In such a multilevel, hierarchical stereo system, results of stereo matching at higher levels of the hierarchy can be used for guidance at the lower levels, and disparity maps generated at each level can be fused to obtain an accurate and fine resolution disparity map. Such a system would also provide the capability to selectively analyze regions with varying resolutions. This selective focusing capability would further enhance its performance.

In this paper we address two specific objectives. The first objective is to characterize a robotic application domain and to evaluate the strengths and limitations of the range-sensing approaches utilized in the robotic environment. The second objective is to present details of a region-based stereo matcher and its associated disparity-depth generation module. This efficient region-based approach should form the highest level of the hierarchical stereo system. Our final goal is to build a complete and effective hierarchical stereo system suitable for use in a robotic environment. The development of a hierarchical stereo system is a complex and challenging task. It requires selection of appropriate primitives for each level and design of various methodologies to integrate the results from these levels. Those design issues are not addressed in depth in this paper, however.

The organization of the paper is as follows. First, characteristics of the robotic environment are discussed along with a brief review of some of the well established range-sensing techniques. This discussion is followed by a detailed description of the binocular stereo approach. Finally the region-based stereo matcher and a series of experiments conducted to systematically evaluate the performance of the region-based approach are presented.

II. RANGE SENSING FOR ROBOTIC APPLICATIONS

In this section we describe important characteristics of the robotic application domain. A brief review of some of the popular range-sensing approaches is also presented.

A. Characteristics of the Application Domain

An intelligent robotic system may utilize depth information and associated 3-D cues in performing one or more of the following tasks:

- 1) object recognition,
- 2) object position and orientation determination,
- 3) object surface or status inspection,

- 4) object manipulation or assembly, and
- 5) obstacle avoidance and navigation.

Tasks 1)-4) can be grouped in the general category of object recognition, inspection, and manipulation, whereas item 5) can be identified as the mobility and navigation in unstructured environments. Obviously the type and nature of the range information required are task dependent. For example object recognition and inspection may require a detailed 3-D surface profile of an object whereas in obstacle avoidance and navigation tasks it may be sufficient to detect the presence or absence of an object (unrecognized). It is important to note that the overall environment in which object recognition, inspection and manipulation tasks are performed is generally quite different from that of mobility and navigation tasks. Typically for object inspection and manipulation tasks, the scene containing the work space is imaged from close distance. The work space usually consists of man-made objects having smooth and finished surfaces that lack rich, highly textured surface details. Furthermore it may not be possible or practical to utilize active illumination sources like textured light patterns [3], because this might interfere and obstruct human operators, who are generally present in the "in-plant" environments. A range-sensing system for these tasks is often required to provide, accurately and efficiently, a complete depth map of the workspace for the entire field of view of the sensor. On the other hand a range-sensing system for mobility and navigation tasks may function in outdoor environments typically characterized as having rich, textured surface details. For these tasks, only 3-D cues for resolving the volume occupancy type of decision need to be extracted [4].

In developing practical range-sensing systems for robotic applications, several, often competing, design parameters need consideration. The following is a list of these parameters:

- efficiency—primarily in speed with which depth information is generated,
- accuracy,
- resolution—spatial as well as of the depth measurements produced,
- input flexibility—should tolerate minor variations in the imaging process,
- noise immunity,
- measurement mode—to derive either the absolute or relative depths,
- reliability,
- repeatability,
- robustness—in handling variations caused by factors such as illumination variations, viewing geometry, work space composition, etc.,
- complexity, and
- cost.

An effective 3-D information extraction system must perform well under these often competing criteria and generate depth information required for a given robotic

task. Thus, a balance must be sought between input flexibility, output sufficiency and computational cost [4].

B. Range-Sensing Techniques

Range-finding techniques can be loosely classified as either active or passive. Active techniques utilize artificial sources of energy to illuminate the workspace, but passive techniques do not require such energy sources. Popular active techniques include: contrived lighting approaches and direct range finders based on time-of-flight measurements. Common examples of passive techniques include: stereo, both binocular and photometric; shape from shading; shape from texture; and focusing methods. Passive methods are preferred in many application areas because they do not use an artificial energy source. These application areas include environments where active illumination is impractical or hazardous. A brief discussion of some popular active and passive techniques is given next. A comprehensive review of range sensing and sensors can be found in [5].

The contrived-lighting approach involves illuminating the scene with a controlled lighting pattern and interpreting the projection of the pattern to derive a depth profile of the scene [6], [7]. Data acquisition is slow in these types of approaches. Such active illumination can also be disadvantageous in an outdoor or hostile environment. This method may also fail because of the specular reflectivity of the objects appearing in the scene. The main advantage of this approach is its simplicity and low cost. In time-of-flight range finders, distance is estimated by the elapsed time between the transmission and the reception of an ultrasonic or laser signal. Ultrasonic range finders lack good depth resolution and are useful for detecting obstacles in navigation tasks. Range finders that use lasers have better resolution than ultrasonic range finders. The major disadvantage of laser ranging techniques is their high cost. Also using lasers in an uncontrolled environment could be hazardous to humans. The advantage of time-of-flight techniques are the following: range measurement is directly available, it produces a dense range map, no (or very little) image analysis is required, and registration of range and intensity images is simple. Another simple, active depth-sensing technique is based upon triangulation and utilizes a narrow light beam to illuminate the scene one spot at a time [5]. This approach also suffers from many of the disadvantages associated with other contrived-lighting approaches.

Passive techniques for range sensing typically require simpler and less expensive setup than do active approaches. The binocular stereo approach falls into this category and will be discussed in the next section. Here we briefly discuss some of the other passive techniques. In the shape from shading approach, surface orientation is derived from the gray-level intensity values, using information about the light source and the surface reflectivity of objects [8]. One of the disadvantages of this type of approach is that it requires many photometric assumptions.

Shape has also been derived from texture gradients [5], [9]. However such methods require scenes to be highly and uniformly textured. Distance can also be derived from focusing a camera until objects appear sharp [5]. The difficulty in measuring sharpness in visually homogeneous regions prohibits these techniques from producing range measurements for such regions. Also, accuracy of this method is reduced as the range increases.

Two different classes of stereo techniques have been used for deriving range information: photometric stereo and binocular stereo. Of these, photometric stereo has been used to compute orientation of surfaces [8], [10], and it seems to work well on surfaces with uniform reflectance functions. This approach, however, has difficulty when the surface is covered with high contrast markings [3]. Binocular stereo on the other hand relies heavily on the presence of surface detail and has been found to operate flexibly under a wide range of conditions without requiring strong photometric assumptions.

III. RANGE-SENSING BY BINOCULAR STEREO

Binocular stereo can produce precise depth measurements over a wide range of distances, which makes it one of the most attractive passive range-finding methods. Yet the high computational cost of the algorithms has prevented it from being widely used in robotic environments. It is however recognized by many in the computer vision and robotics research communities to offer the most promise for the future [11].

If a 3-D object point is sensed from two different locations, the relative positions of its image differ in the two views. In the binocular stereo approach one measures this difference (disparity) in the relative positions in the images to compute depth information. Typically the binocular stereo approach is characterized by the following four steps.

- Step 1) An image feature of interest, corresponding to a surface point, is located in one image.
- Step 2) The other image of the stereo image pair, is examined to identify the unique image feature corresponding to the projection of the same surface point.
- Step 3) The disparity in the locations is measured.
- Step 4) Disparity value is converted into actual range information using geometric transformations.

Of the four steps, Step 2), which deals with the stereo matching or the correspondence problem, lies at the heart of the binocular stereo approach.

To deal with the correspondence problem, and its concomitant problem of avoiding false targets in determining the correct correspondence or match, it is vital to focus on appropriate representations or primitives, to be matched and on the matching constraints and rules matching. The success and efficiency of a binocular stereo approach is dependent on both the nature of the primitive used and the matching technique itself. The issues related to the

selection of an appropriate matching primitive and a description of some of the common constraints and rules that can be utilized for making the matching process computationally tractable are discussed in the following. A brief discussion of some of the popular matching techniques is also presented.

A. Role of Matching Primitives

The comparison of image brightness (gray-level) between left and right images is a weak indication of the correspondence, because imaging geometry, camera characteristics, and noise make the image elements differ in their gray-level values. Furthermore featureless areas of nearly homogeneous brightness values in images make the matching of gray levels unsuitable for solving correspondences. Accordingly most computational stereo algorithms include some form of primitive extraction phase to derive, from the images, features or measurements that can be used for effective stereo matching.

Two important properties sought in matching primitive are its invariance to small variations in image viewpoint, (referred to as stability), and its descriptive power to resolve ambiguities. Selection of an appropriate matching primitive is influenced by the following characteristics of the primitive [12]:

- dimensionality (point like vs. edge like),
- size (spatial frequency),
- contrast,
- semantic content,
- density of occurrence (sparse vs. dense),
- ease in extracting attributes, and
- distinguishability/uniqueness.

Most stereo algorithms use low-level primitives, i.e., those that do not require sophisticated semantic analysis in their extraction. This may be partially attributed to the fact that there is a trade-off between the complexity of the monocular analysis used to extract the features to be matched and the complexity of the process utilized in matching them. Furthermore investigations into human stereopsis have suggested that stereopsis occurs very early in the visual process rather than following a detailed analysis of each monocular image separately [13]. The stereopsis experiments by Julesz on random dot stereograms, substantiates this theory [14], [15].

Popularly used matching primitives include: sign of the Laplacian of the Gaussian: i.e., $\nabla^2 G$ [3]; zero crossings of $\nabla^2 G$ [13], [16]; edges [17] and linked edge segments [18], [19]. These types of primitives are relatively sparse in images and thus require more careful and more explicit matching rules to eliminate false targets. In an efficient and accurate stereo system the matching primitive needs to be easy to extract without much image analysis, yet should possess adequate semantic information. They should not be very dense in occurrence but should be sufficient for useful interpretation. Furthermore the matching technique itself should be designed to exploit the characteristics of the primitives.

B. Constraints and Rules for Stereo Matching

Similarity is the guiding principal for solving the correspondence problem. Derivation of a matching primitive that contains adequate power to resolve ambiguities and is truly invariant with respect to the viewing geometries, is a difficult task. Hence, in general, for a particular image feature in one image, there will be many candidate matches in the corresponding image. Matching rules derived from the constraints underlying the physical environment and imaging are used to restrict this pool of candidate matches.

Some of the common constraints incorporated in stereo algorithms, as identified by Nishihara and Poggio, [4] are the following.

- 1) *Surface continuity constraint*: Assumes that the physical world is composed of surfaces that are almost continuous everywhere. This suggests that disparities ought to vary smoothly.
- 2) *Surface uniqueness constraint*: Relates to the fact that the imaged surfaces, for the most part, are opaque, allowing us to assume that the image element recorded by a camera corresponds to a unique point lying physically on the surface of an object. Thus correspondence should be unique.
- 3) *General position constraint*: Relates to the observation that certain events occur quite infrequently, in a statistical sense, to rule out false correspondences. Both surface continuity and surface uniqueness constraints were formulated and used by Marr and Poggio [13], while Arnold and Binford [20], [21] have utilized general position constraints.

Nishihara and Poggio have identified two broad categories of matching rules in stereo algorithms [4]. Most of the existing algorithms seem to use at least one rule from each class. These categories are as follows: spatial-domain rules and gradient-limit rules. We discuss them in detail next.

1) *Spatial-domain rules*: This class of rules is based on the surface continuity and the general position assumptions of the matching environment. Examples include the following.

Area statistics: Matching primitives collected over an image measurement patch are compared across images to obtain a single similarity measure. Correlation of image features is an example of this class. This rule typically implies a strong continuity assumption, because it imposes approximately constant disparity over the patch. Nishihara's Prism stereo algorithm [3] belongs to the class of algorithms that uses this type of matching rule.

Contour statistics: In this the comparison is restricted along a contour. The assumption of physical surface continuity is made weaker by assuming disparity to be smooth along contours but allowing them to change abruptly across contours. Figural continuity, as suggested by Mayhew and Frisby [22] and used in Grimson's stereo algorithm [16] is an example of this type. Similar contour based constraints have also been used by Arnold and Binford [20].

2) *Gradient limit rules*: This class of rules is based on the manner in which images are manifested. The following are examples of rules from this class.

Ordering constraints: Impose the restriction that along epipolar lines, the matched primitives must occur in the same order. This is equivalent to the assumption that imaged surfaces are not transparent and are continuous. This type of constraints have been used by Arnold [21] and also by Baker and Binford [17].

Disparity gradient limits: Restrict the maximum disparity gradient allowed between matched primitives. Pollard, Mayhew, and Frisby have suggested that for most natural scene surfaces the disparity gradient between correct matches is usually less than one [23]. They have proposed a stereo algorithm based upon this observation.

Coarse-to-fine analysis: In this analysis disparity information obtained at a coarser scale is used to limit the search domain for the matching of finer scale primitives. This is used with scale specific matching primitives. Many including Marr and Poggio [13], Grimson [16], and Nishihara [3] have used this strategy for stereo matching.

C Edge-Based Stereo Approaches

Most stereo approaches use matching primitives based on gray-level discontinuities or edges. Stereo-matching algorithms that use edges are usually more efficient than those using more densely occurring primitives. However because of the sparse and irregularly distributed nature of these features the edge-based matching approaches produce a sparse depth map. Such methods must be augmented by an interpolation step if a complete depth map of the scene is desired. They also require carefully selected and more explicit matching rules to eliminate false targets. Explicit global consistency mechanisms are also incorporated in these algorithms for satisfying constraints such as surface continuity, figural continuity, and constraints due to ordering and disparity gradients. Examples of edge-based matching algorithms include those by Marr and Poggio [13], [24], Grimson [16], Arnold [20], [21], Baker and Binford [17], and Medioni and Nevatia [18]. While Nishihara's Prism system does not use edges as matching primitives, his representation, sign of the $\nabla^2 G$, is still derived based on gray-level discontinuities [3]. In this subsection we briefly review and analyze the abovementioned algorithms in the context of their utility in a robotic environment and present some enhancements recently proposed by Hoff and Ahuja [25] and Mohan *et al.* [26] to improve the performance of edge-based techniques.

Marr and Poggio use the zero crossings of the circularly symmetric Laplacian of Gaussian, $\nabla^2 G$ operator as the matching primitive and features are extracted at several different spatial frequencies, or scales (coarse-to-fine), by convolving the image with different size masks. Given a set of these zero crossing representations at different scales for each of the images, the matching process includes an iterative examination of coarse-to-fine scale features. The matches at coarse scales are used to constrain the matching of finer details. The sparse features from coarse spatial

filtering reduces search space and makes matching easier. Matching is performed along horizontal epipolar lines, and the search range for matching is constrained by the size of the convolution mask in use. A match is detected if the zero crossings have the same contrast sign and roughly the same orientation. In cases where the region under consideration does not lie within the current disparity range examined by the matcher, an area statistic based on the continuity assumption is used to distinguish correct matches from random ones. Finally the sparse disparities obtained from the coarser filters are used to realign the images and bring the finer representations within the range of the matcher. This coarse-to-fine control strategy allows matching of very dense zero crossing descriptions with a greatly reduced false target problem. While the overall performance of the algorithm is demonstrated on number of natural and random dot images the computational complexity of the algorithm makes it unsuitable for use in an "in-plant" environment. The algorithm relies heavily on rich surface detail and performs poorly in the absence of such detail. Matching will also fail if repeated or uniform detail is present. And, finally the stringent epipolar matching geometry makes the method extremely sensitive to small vertical misalignments, an effect difficult to arrest in imaging in a robotic environment.

While the use of continuity constraints for eliminating unacceptable candidate matches in the zero crossings algorithm was sound, the difficulties in the implementation of it, by means of statistical measures computed over a region led to the development of Grimson's algorithm. In this rather than imposing a condition of disparity continuity over an area in the image it requires a continuity along a contour in the filtered image. This is a contour-based analogy to the regional check used by the previous algorithm and is essentially the figural continuity suggested by Mayhew and Frisby [22]. This algorithm also relaxed the assumption of epipolar-matching geometry slightly at the expense of a heavy computational burden. The computational times of order of an hour are reported for processing 512×512 images with 100-pixel disparity range on a specialized MIT Lisp machine [3]. Such long processing times prohibit this algorithm from being used in a robotic environment, because depth extraction is just one of the many functions a sensor-driven robot must perform. The algorithm will perform poorly in the absence of rich detail or in the presence of repetitive or periodic detail. Performance will also degrade rapidly for images with vertical misalignments in excess of one or two epipolar lines.

Nishihara's Prism system uses a correlation based approach to stereo matching using the sign of the $\nabla^2 G$ convolution as the matching primitive rather than the zero crossings of the $\nabla^2 G$ operator. In order to deal with visually homogeneous regions within the field of view of the camera, the scene is illuminated by projecting an unstructured texture pattern using a slide projector. The Prism was designed to be used in a robotic environment to rapidly detect obstacles and to determine their rough extents and heights. It produces a coarse surface elevation map over the entire camera field as an output. It uses a

near/far matching strategy similar to the coarse-to-fine approach, where matching at a finer scale is loosely coupled with coarser results guiding the search at the next finer scale. While it performs efficiently to produce a 36×26 disparity array over the entire field of view of the camera, it utilizes specialized hardware to perform the computations. Furthermore illuminating the scene with a textured light pattern may not be possible or practical for many inspection and manipulation tasks, because this might interfere with and obstruct human operators, who are generally present in the robot's environment.

Arnold's stereo algorithm matches edges using epipolar geometry. In this implementation, edges are extracted interactively by superimposing line drawings on the gray-scale images. Rather than matching edges in isolation using only local information, he uses several constraints based upon global information to reduce ambiguity and to arrive at a globally consistent interpretation. Constraints used include the following: occlusion constraints, ordering constraints, edge angle and edge interval constraints, and constraints based on the edge continuity in the context of adjacent epipolar lines. Matching is performed by a dynamic programming algorithm developed by Viterbi [27], which is modified to incorporate these constraints. The modified Viterbi algorithm finds the best match between two given sequences of edge points of two epipolar lines from the left and right images. The cost evaluation function used for scoring utilizes the constraints on the edge angle and edge interval [20]. While the goal of the Viterbi algorithm is to find an optimal match, it is noted that a globally optimal match might, indeed, be suboptimal in the limited context of a single epipolar line. Hence the modified Viterbi algorithm is extended to avoid the premature discarding of suboptimal matches. This extended version maintains a pool of suboptimal matches, which is later filtered by an iterative process that enforces consistency across adjacent epipolar lines. This algorithm too suffers from high computational complexity to be effective in a robotic environment.

Baker and Binford also use the modified Viterbi algorithm in their edge and intensity-based stereo matcher. The matching primitives used in their algorithm are linked edges and ordering constraints are utilized to match them. Matching is restricted to epipolar lines and proceeds in a coarse-to-fine manner. However, a cooperative process discards, based upon the connectivity of edges across adjacent epipolar lines, correspondences that violate global continuity assumptions. Following edge-based matching, a further matching phase is performed on the intensity values from the intervals between paired edges, again using the modified Viterbi algorithm. The result is a complete depth map over the entire scene. Heavy dependence on linked linear edges and epipolar matching may make this approach too fragile to use in a robotic environment.

The algorithm developed by Medioni and Nevatia uses straight line segments extracted by a linear feature extractor [28] as the primitive of stereo matching. Important characteristics of their segment-based stereo approach in-

clude the following: they use line segments to make inter-line connectivity implicit in their matcher, one segment is allowed to be matched with more than one segment in the other image to deal with fragmentation that may occur during extraction of these features, and matches are sought over the total extent of epipolar lines in which a given segment appears rather than one epipolar line at a time. The matches are chosen based on an evaluation function that uses a disparity continuity constraint, but does not adhere to an ordering constraint. Their results clearly demonstrate the power of choosing primitives that are semantically richer than individual edge points. They have suggested that some higher-level monocular cues might be required, in addition to line segments, to solve correspondences in complex scenes. Computational times of several minutes are reported for their experiments.

Recently proposed enhancements of stereo approaches include attempts to detect and correct errors committed by stereo matching [26] and integration of feature matching with disparity generation [25]. While accuracy of the results has been improved, both approaches have resulted in a heavy increase in computational time. In addition to the heavy computational requirements of the edge-based approaches, there are several other major difficulties that these approaches encounter in robotic environments. These include issues related to noise immunity, accuracy, and robustness.

IV. REGION-BASED ANALYSIS FOR A HIERARCHICAL STEREO SYSTEM

A practical stereo system for robotic applications should produce a depth map of sufficient detail to accomplish the prescribed robotic task efficiently. The system must also exhibit a high-noise tolerance level for its inputs. It is critical for such a system to employ an accurate, reliable, and efficient module for stereo matching. The choice of the matching primitives will greatly influence the performance of such a matching module. In this section we will introduce the use of homogeneous regions as features in stereo matching. Region-based stereo matching is typically more accurate, reliable, and efficient than edge-based matching. Yet a region-based matcher lacks the capability to generate an accurate disparity map of fine resolution. Generation of an accurate and fine resolution disparity map (and subsequently depth) can be better accomplished using edge-based techniques. Consequently regions and edges both play important and somewhat complementary roles in the binocular stereo process. Therefore it is critical that an efficient and robust stereo system utilizes the most appropriate set of primitives at each stage of the process. It is believed that a hierarchical, multilevel stereo approach that exploits and integrates the power of different primitives at the appropriate stages of the computational process, will be most useful in building an effective stereo system. In such a hierarchical stereo system results of stereo matching from the higher levels of the hierarchy can be used for guidance at the lower level, and disparity maps

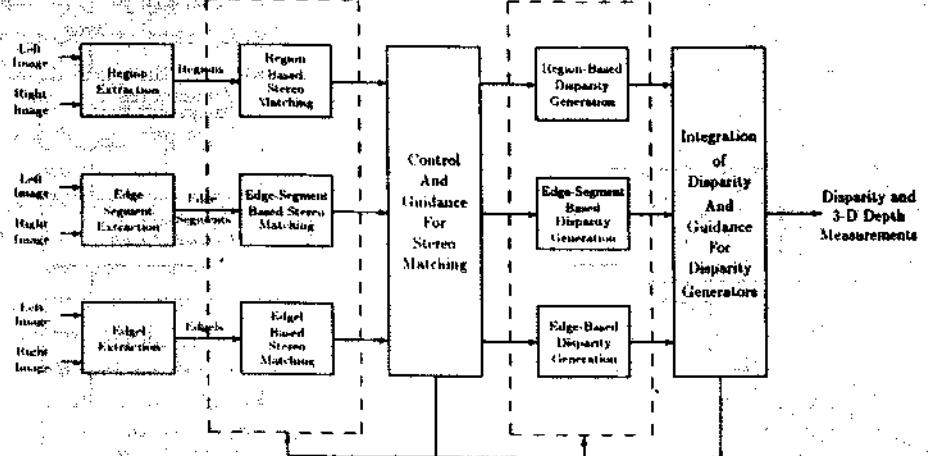


Fig. 1. Multilevel hierarchical stereo system.

generated at each level can be fused to obtain an accurate and fine resolution disparity map.

A schematic of a hierarchical stereo system is presented in Fig. 1. The proposed multilevel system incorporates hierarchical stereo matching as well as hierarchical disparity generation. A system that utilized hierarchy of primitives to solve only the matching problem was presented by Lim and Binsford [19]. In our approach we have addressed both matching and disparity generation in a hierarchical framework. The system utilizes primitives from three hierarchical levels. At the highest level are region primitives, followed by edge segments, and at the lowest level are the single pixel-edge elements, edgels. Matching and disparity generation can be performed at three levels, and lower-level processes can utilize results of the higher levels for guidance. This system offers many advantages over those that use only a single primitive for stereo matching and disparity generation. The multilevel hierarchy will greatly reduce mismatches and inconsistencies and, thereby, produce an accurate, globally consistent, and reliable output. Control and guidance mechanisms provide the capability to selectively analyze regions with varying resolution. This selective focusing capability would further enhance its performance by concentrating on critical areas, eliminating the need for detailed examination of the entire image [3]. Note that the complete stereo problem is solved entirely at each level. Obviously their computational requirements differ and the results derived at different levels would differ in accuracy as well as resolution. Multilevel processing also enables the system to generate outputs of only sufficient detail required for a particular task, and further reduces demands on resources. Such a system will exhibit a high level of adaptability to different robotic tasks, requiring depth information in varying degrees of detail and resolution.

Development of a complete hierarchical stereo system is a complex and challenging task. It requires careful and detailed conceptualization, design, and evaluation of each individual component making up the complete system. Our final objective is to develop such a system. In order to pursue this objective, we have attempted to systematically

conceptualize, design, and test each component separately. Even though development is carried out separately, the components will eventually have to be integrated.

The primary focus of this section is to present the detailed development and evaluation of region-based stereo matching module and disparity generation module associated with the highest level of a hierarchical stereo system. We begin by presenting the rationale for the use of homogeneous regions as primitives for matching and then describe the region-based stereo matcher and disparity generator. Finally with the help of a series of experiments, we evaluate the robustness, accuracy, and efficiency of these modules.

A. Use of Regions as Matching Primitives

As described in Section III-A, two of the main properties sought in an effective stereo-matching primitive are its stability and its descriptive capability to resolve ambiguities. In image analysis there are basically two complementary approaches for extracting low-level information from images. The first is based on the property of dissimilarity, i.e., edge-based, and the second is based on the utilization of the property of similarity, i.e., region-based [29]. Primitives extracted from either of these approaches can be utilized in a stereo system. The higher semantic content of a region makes it a good candidate for a matching primitive. Also regions will be more stable than those primitives that depend on local gray-level discontinuities such as edges [29], [30]. Thus region-based primitives are more tolerant to noise than edge-based primitives. Furthermore in an image there will be typically fewer regions than edges or zero crossings of the Laplacian of a Gaussian operator. Therefore establishing image correspondences will be more efficient using regions as features, because the number of candidates to be evaluated is greatly reduced. This reduction in the number features along with the region-based primitive's high discrimination power reduces the number of mismatches (false targets) and, therefore, increases the accuracy of matching.

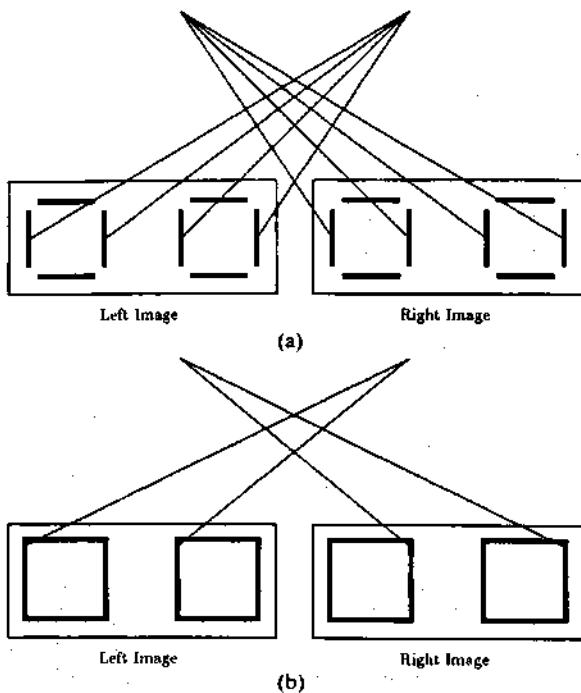


Fig. 2. Resolution of ambiguity in stereo correspondence. (a) Shows various combinations that need to be resolved if edge-segments are used as features. (b) Shows situation when regions are used as features.

The improved efficiency and the reduction in the number of mismatches using regions as features in stereo matching can be illustrated with the help of a simple example. Fig. 2(a) shows the horizontal and vertical edges extracted in the stereo image pair of a scene containing two similar square objects. If edge segments are used as features, an attempt to match the four vertical segments in the left image with the similar four segments in the right (Fig. 2(a)) will yield 24 possible combinations. Also if one observes the epipolar constraints, an attempt to match the top two horizontal segments gives rise to two possibilities. The same can be stated for the two bottom segments as well. Thus segment-based matching between the eight segments appearing in each image will lead to a total of $24 \times 2 \times 2 = 96$ combinations, of which only one combination is correct. If individual edge pixels rather than segments are used as features, the total number of combinations will be greatly increased. On the other hand if regions are used as features, then matching requires solving correspondence among the two square regions, as shown in Fig. 2(b). Therefore when using regions as features there will be fewer features to be matched, and the higher semantic content and reduced number of features will also decrease the number of false targets. Another advantage of using regions in stereo matching is that they make some of the matching constraints (described in Section III-B) implicit or easier to incorporate. Finally the high semantic content of the regions also makes it possible to use a wide range of region representation and description schemes that, in turn, allow the use of a variety of matching techniques. It is therefore believed that region-based stereo will yield a reliable, accurate, and more

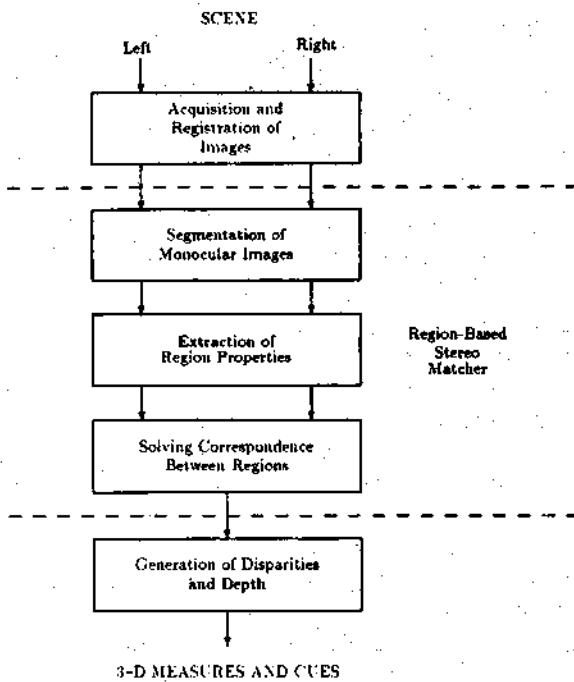


Fig. 3. Region-based analysis for stereo matching and disparity generation.

globally consistent solution efficiently satisfying the constraints of an effective stereo system for robotic applications.

B. Design of a Region-Based Stereo Matcher

The region-based component of a hierarchical stereo system consists of two modules. The region-based matching module solves the correspondence problem, and the second module generates the disparities. The processing sequence of the region-based component is shown in Fig. 3. It includes the following steps.

- Step 1) Acquisition and registration of the stereo image pair.
- Step 2) Segmentation of the two monocular images.
- Step 3) Extraction of the region properties.
- Step 4) Solving correspondence between regions.
- Step 5) Generation of disparities.

The matching module includes processing Steps 2), 3), and 4), whereas Step 5) corresponds to the disparity generation module. We now present a detailed discussion of these steps.

Two types of imaging geometries have been commonly used in stereo-image acquisition. In type one, the cameras are oriented such that their optical axes converge at a point called the fixation point. The disparities between the image features in this geometry can be either positive or negative. The sign of the disparity depends on the location of the physical surface point (giving rise to the image feature) with respect to the fixation point. For a particular image feature \mathcal{F} in one image, its corresponding feature in the other image, may lie to the left or right of \mathcal{F} . Thus search for correspondence needs to be carried out on both

sides of \mathcal{F} . The second type of camera geometry, known as the parallel axis geometry, employs two cameras whose optical axes are parallel. This geometry can be thought of as having the fixation point at infinity. Here all disparities have the same sign and, hence, the search for correspondence is restricted to only one side. As compared to the convergent axes geometry, parallel axis geometry yields less overlap in the left and right images of the scene being imaged. It is also desirable to maintain the epipolar lines along the horizontal image scan lines in both parallel and convergent axes geometries. Thus registration of left and right images may be required to bring the images into sufficient vertical and rotational alignment in order to satisfy input requirements of the matching module.

The processing step following image acquisition and registration deals with the segmentation of each of the two monocular images separately. Image segmentation refers to the partitioning of an image into meaningful regions. It is important to note that meaningful regions may not necessarily correspond to any physical objects appearing in the scene. Formally given a definition of "uniformity" a segmentation is a partitioning of an image into connected subsets, each of which is uniform, but such that no union of adjacent subsets is uniform [31]. There are two basic approaches to segmentation: the first is based on edge or boundary formation, and the second is based on region formation [29]. Typical region-forming approaches utilize uniformity (or homogeneity) primitives based on such image properties as gray levels, multispectral or color measurements, and texture. There are numerous techniques proposed for segmentation, and a good comprehensive review can be found in reference [29]. For the proposed region-based stereo approach, the choice of a segmentation algorithm is arbitrary. However, the algorithm should be robust and capture the uniformity or homogeneity properties of the region accurately.

In Step 3), the segmented homogeneous regions are further processed to extract various properties of these regions. These properties may include spectral and spatial as well as relational properties. Typical spectral properties may include measurements derived from gray levels recorded in single or multiple channels. Both boundary and region descriptors may be used to describe the spatial properties of regions. Popular boundary descriptors include length, diameter, curvature, and chain codes. Regions may be described by area, perimeter, compactness, major and minor axes, length and width, aspect ratio, moments, texture, and topological descriptors such as Euler numbers. Relational descriptors can be used to describe the structural relationships existing between elements of interest, in this case, the regions. These relationships can be described using relational graphs. The availability of such a wide array of descriptors allows utilization of rich descriptive vocabulary when using regions as features.

Finally in Step 4) correspondence will be resolved based on the similarity between the candidate regions. Regions satisfying a variety of constraints or rules, such as those

described in Section II-B, are selected as candidate regions. Solving for correspondence can be accomplished through a variety of matching techniques including, but not limited to, minimum distance classifiers based on a similarity metric, optimization methods, and graph theoretical approaches. Distance-based matching is implicit in many edge-based stereo algorithms that match individual edge pixels based on the similarity of such properties as contrast and orientation of candidates. Typical optimization methods, such as dynamic programming, relaxation techniques, and simulated annealing have been used in solving stereo correspondence [21], [32]–[34]. Recently graph-matching, using a graph isomorphism algorithm, has also been used in stereo vision [35]. The choice of a matching technique is dependent on the properties and representation schemes used in Step 3) for describing the regions, as certain representations support specific matching techniques better than others.

C. Implementation of the Region-Based Modules

In order to verify the performance of the region-based stereo approach, we implemented a region-based matching module and a disparity generation module. Our implementation uses a single camera mounted on a Cincinnati Milacron T^3 -726 industrial robot. Stereo image pairs are acquired by moving the robot arm to two locations using parallel-axis geometry with horizontal epipolar lines. The use of a single robot-mounted camera provides a flexible image acquisition system with a variable baseline. Experiments performed indicate that the robot arm could be positioned accurately to guarantee that the acquired images are in sufficient vertical and rotational alignment to satisfy the input requirements of the stereo matcher. Thus no image registration is required.

Image segmentation is accomplished by a region-growing algorithm, which employs a homogeneity criterion based on the mean and the standard deviation of the gray-level values of a region. In addition to segmenting images into regions (or blobs), the segmentor also extracts various spectral and spatial domain properties for each of those segmented regions [36]. These properties include: mean gray level, size (area), minimum enclosing rectangle, centroid, perimeter length, principal axis (PA), width along the PA, height perpendicular to the PA in pixels, and width-to-height (aspect) ratio. Constraints on the size of regions are used to discard small noise regions and large uniform background regions.

Solving correspondences between blobs in the left and right images requires a metric to measure similarity (or dissimilarity). For comparing similarities/dissimilarities between blobs, we have chosen to describe each blob by six region descriptors. The six features chosen as attributes of a blob are: mean gray level, area, perimeter, width, length, and the aspect ratio. These features, except for aspect ratio, are normalized in both left and right images to facilitate matching. The current implementation uses Euclidean distance in the six-dimensional measurement

space as the metric of dissimilarity between blobs. Thus the correspondence for a blob among several candidate blobs is found in the measurement space using the minimum distance rule. Candidates are chosen based on epipolar constraints similar to those used by Medioni and Nevatia in their edge-segment based stereo matcher [18], as follows. For each blob, R , in the segmented right image, only those blobs in the left image that lie between the epipolar extent of R and are within the maximum disparity range of the matcher are chosen as the candidates for matching. Of the pool of candidates, the nearest (or least dissimilar) blob, in terms of minimum distance in the measurement space, is selected, and correspondence is established if this minimum distance is less than a preset threshold. No match is reported for blob R , if no blobs in the left image lie within the epipolar extent of R , or no blobs in the left image that lie within the epipolar extent of R are within the preset distance threshold in the measurement space.

Once correspondence is established between a blob in the left image and a blob in the right image, disparity between them is computed to be the displacement (in pixels), along the epipolar line, required to produce the maximum overlap in their spatial extent. Specifically the blob in the left image is translated along the epipolar line towards the left, and the overlap area between the left and right blobs is computed. The overlap area is normalized by dividing it by the area of the larger of the two blobs. The displacement at which this normalized overlap area is a maximum is considered to be the disparity between the two blobs. If this maximum normalized overlap is greater than a preset threshold, the displacement (in pixels) that the left blob was translated is accepted to be the disparity and is assigned to the region of overlap between the two blobs. The range in which a maximum overlap is sought can be computed by using the minimum depth expected in the operating environment and the imaging parameters. Finally the disparities are converted to actual depth measurements, using the imaging and camera parameters. The equations for this conversion are described in Kim and Aggarwal [33].

D. Experimental Evaluations of the Region-Based Analysis Modules

The performance of the region-based stereo component was evaluated by conducting a series of experiments. No specialized lighting was utilized in conducting these experiments, and scenes analyzed were comparable to the types of scenes reported in current stereo literature. The experiments were designed to evaluate specifically the following.

- 1) Stability of the regions as matching features.
- 2) Discriminatory power of the measures derived from regions.
- 3) Accuracy of region-based matching.
- 4) Efficiency of region-based matching.
- 5) Accuracy of disparities and depth measurements derived by the region-based approach.

The first experiment consists of a simple scene containing two objects and only two depth levels. Figs. 4 (a)-(b) show a pair of 256×256 stereo images. Of the two objects, the cylindrical jar is taller than the plier. The results of segmentation of the stereo image pair are shown in Figs. 4(c)-(d). Six normalized descriptors were extracted from each blob in left and right segmented images. Of the five blobs found in the segmented left image, blob 1 corresponds to the plier and blob 4 corresponds to the jar. Similarly, in the segmented right image, blob 1 is the plier and blob 3 is the jar. None of the remaining blobs correspond to any physical object and they are not stable across the images. This demonstrates that regions manifested due to the physical composition of the scene tend to be more stable than those regions that arise due to photometric effects. Similar observations are also made in edge-based stereo systems [12].

The dissimilarities between blobs in the right and left segmented images are tabulated in Table I. Though all possible distances are listed in the table, not all entries need to be computed, because the epipolar constraint eliminates several comparisons. The relatively small dissimilarities, as opposed to other entries in the table, between the left blob 1 and the right blob 1, and the left blob 4 and the right blob 3, indicate the discriminatory capability of regions as matching primitives. The highlighted entries denote the correct match among the candidates. The computed correspondence and the disparity measures are shown in Table II. The minimum distance rule is able to correctly establish correspondences. Disparities indicate that the cylindrical object is indeed closer to the viewer than the plier. A 3-D plot of the resulting disparity map is shown in Fig. 5.

For the second experiment, we utilize a scene comprised of a number of industrial parts. They include a rectangular box, a circular jar, a valve, and a plier. The scene has four distinct depth levels. The 256×256 stereo image pair, acquired from a distance of about 1.3 feet, is shown in Figs. 6 (a)-(b). The results of segmentation are shown in Figs. 6(c)-(d). Normalized descriptors for each segmented blob in the left and right images were calculated. Pairwise dissimilarities between the blobs are shown in Table III. Dissimilarities indicate that regions possess sufficient information to identify corresponding blobs. The correspondences established, based upon the minimum distance rule and the disparity obtained using normalized overlap ratio, are shown in Table IV. Once again, the correspondence is solved correctly, and disparities agree with the actual depth levels. The disparity map generated by the region-based approach is shown in Fig. 7. These disparities are then converted to actual depth. The computed distance, measured distance, and percentage error for each of the objects is tabulated in Table V. The results are quite accurate.

The third experiment consists of a scene containing doughnut-shaped objects (rings) on a table top. A pair of 256×256 stereo images, acquired from about 1.3 feet from the table top, is shown in Figs. 8(a)-(b). The smaller ring with lighter shade is supported by two darker rings. These

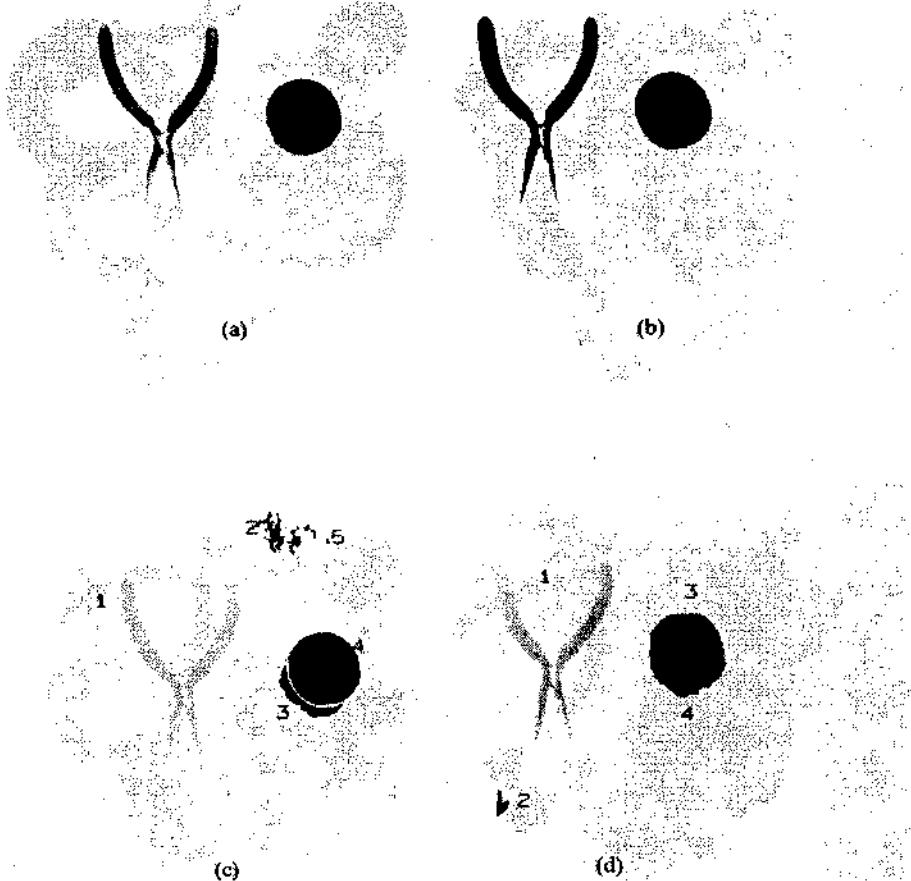


Fig. 4. Simple scene with two objects. The 256×256 stereo image pair is shown in (a) left and (b) right. Results of segmentation are shown in (c) and (d).

TABLE I
SIMPLE SCENE: SIMILARITY MEASURES BETWEEN DETECTED BLOBS

Blob ID Left Image	Blob ID Right Image			
	1	2	3	4
1	0.07	1.82	1.05	1.44
2	1.64	0.58	1.06	0.46
3	1.26	0.81	0.73	0.41
4	1.06	1.24	0.03	0.96
5	1.46	0.82	0.98	0.56

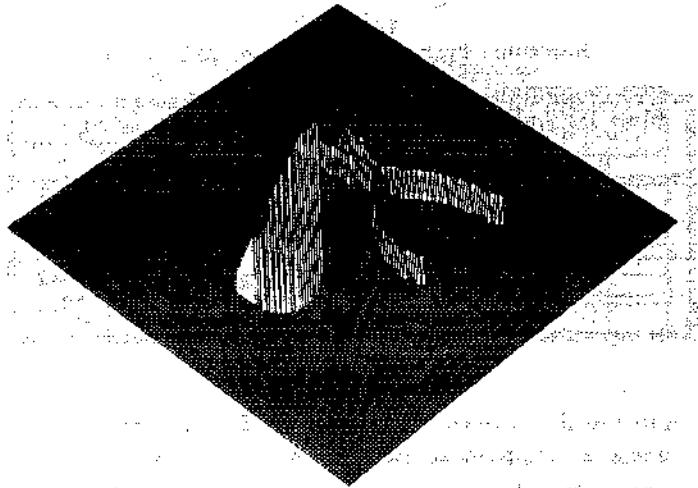


Fig. 5. 3-D disparity map for simple scene.

TABLE II
SIMPLE SCENE: CORRESPONDENCES AND DISPARITIES
BETWEEN DETECTED BLOBS

Blob ID Left Image	Blob ID Right Image	Similarity Measure	Overlap Ratio	Disparity (in pixels)
1	1	0.07	0.90	41
4	3	0.03	0.90	46

rings are of same height. The fourth ring is much taller (about three times) than the other rings. Thus this scene contains four different depth levels. The results of segmentation are shown in Figs. 8(c)-(d). The correspondences and disparities computed are shown in Table VI. The table indicates that correspondences were established accurately

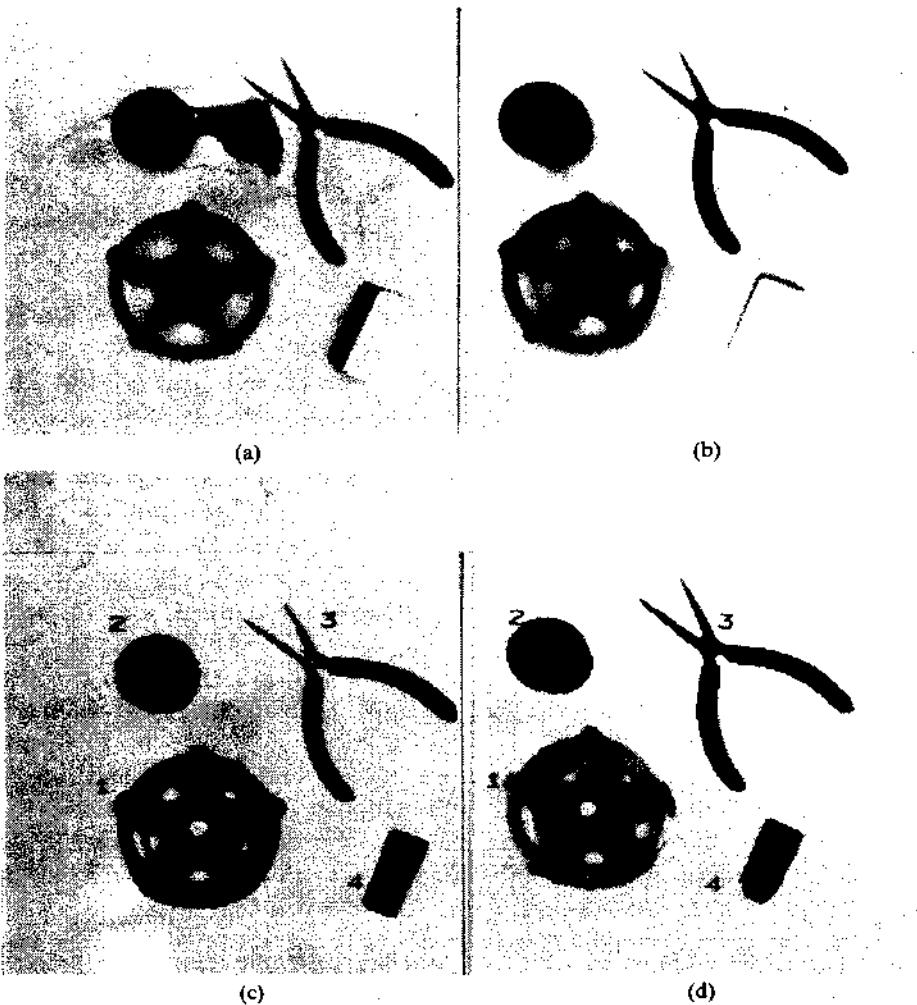


Fig. 6. Scene composed of industrial parts: The 256×256 stereo image pair is shown in (a) left and (b) right. Results of segmentation are shown in (c) and (d).

TABLE III
INDUSTRIAL PARTS: SIMILARITY MEASURES BETWEEN
DETECTED BLOBS

Blob ID Left Image	Blob ID Right Image			
	1	2	3	4
1	0.10	1.02	0.95	1.35
2	0.97	0.09	1.07	0.77
3	0.84	1.10	0.35	1.42
4	1.37	0.87	1.32	0.11

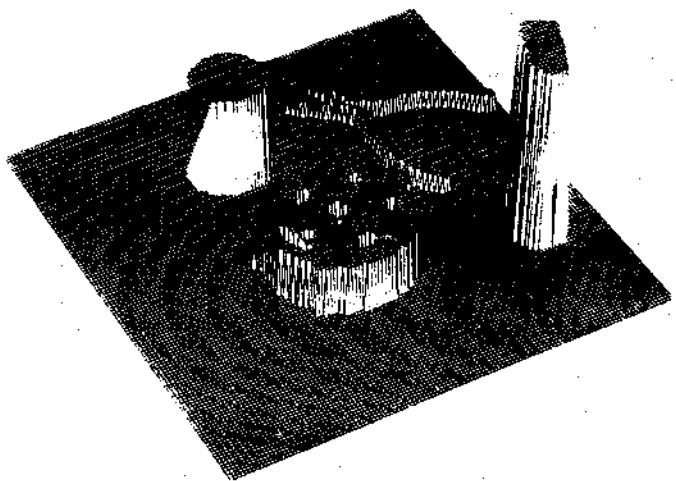


Fig. 7. 3-D disparity map for scene composed of industrial parts.

TABLE IV
INDUSTRIAL PARTS: CORRESPONDENCES AND DISPARITIES

Blob ID Left Image	Blob ID Right Image	Similarity Measure	Overlap Ratio	Disparity (in pixels)
1	1	0.10	0.91	61
2	2	0.09	0.91	64
3	3	0.35	0.77	59
4	4	0.11	0.87	69

for all of the blobs that were matched. One of the dark rings that supports the smaller ring is erroneously assigned the same disparity as the smaller ring. Disparities assigned to all other rings were consistent with their heights (Fig. 9). An analysis of depth measurements is tabulated in Table VII.

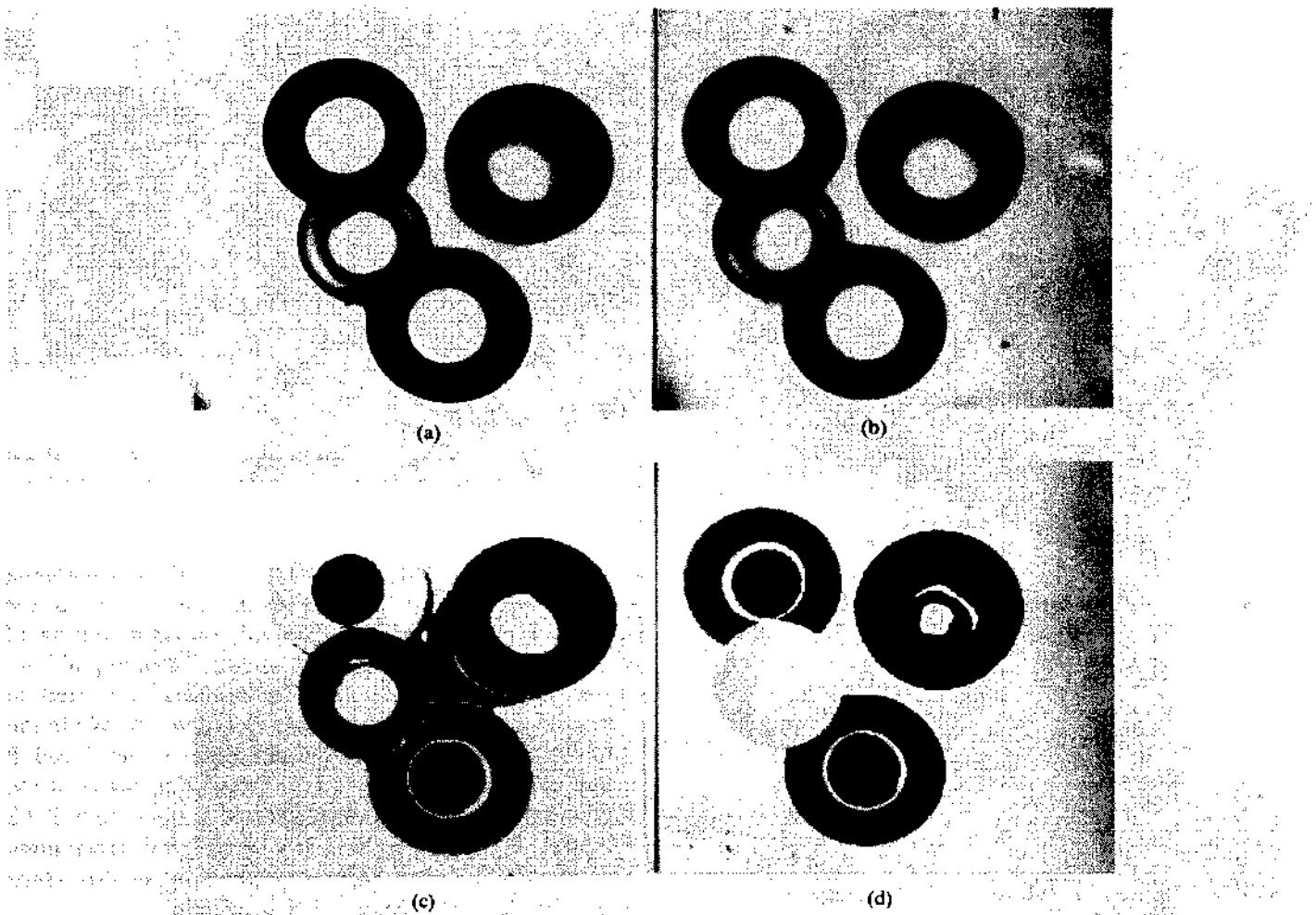


Fig. 8. Scene composed of four rings. The 256×256 stereo image pair is shown in (a) left and (b) right. Results of segmentation are shown in (c) and (d).

TABLE V

INDUSTRIAL PARTS: COMPARISON OF COMPUTED AND MEASURED DISTANCES

Object	Actual Distance	Computed Distance	% Error
Plier	17.25"	17.80"	-3.2%
Valve	16.25"	17.22"	-5.9%
Jar	15.75"	16.41"	-4.2%
Box	15.625"	15.23"	2.5%

The fourth scene contains a test panel, used in our testbed, for robotics experiments. The panel contains typical instruments commonly found in an industrial environment (Fig. 10). These include analog meters, digital meters, valves, a slider control, push-button switches, etc. The input stereo image pair and the segmented images are shown in Fig. 11. This scene was imaged from about 2.5 feet from the panel. The panel contains many objects and several of them are quite similar in appearance. Also there are distinct depth levels associated with these objects. A total of 17 regions in the left image and 21 in the right image were extracted. Matching results indicate that correspondences were established correctly for most of the blobs associated with various objects. A total of 12 blobs

TABLE VI

RINGS: CORRESPONDENCES AND DISPARITIES

Blob ID Left Image	Blob ID Right Image	Similarity Measure	Overlap Ratio	Disparity (in pixels)
1	2	0.04	0.94	33
2	1	0.15	0.87	34
3	3	0.05	0.92	32
4	4	0.04	0.93	34
5	5	0.03	0.95	32
7	6	0.04	0.93	37

were matched, and all of them were correct. All but one object mounted on the panel were successfully matched. Two identical LCD meters at the bottom of the panel seem to confuse the matcher, as there are no ordering constraints in the current implementation. Thus only one LCD was correctly matched. This experiment clearly demonstrates the robustness of the stereo matcher. The depth map generated by the region-based approach is shown in Fig. 12. Finally the computed distances, measured distances and percentage error for selected objects are presented in Table VIII. The disparities generated provided depth measurements within about ± 2.5 percent of the actual depths.

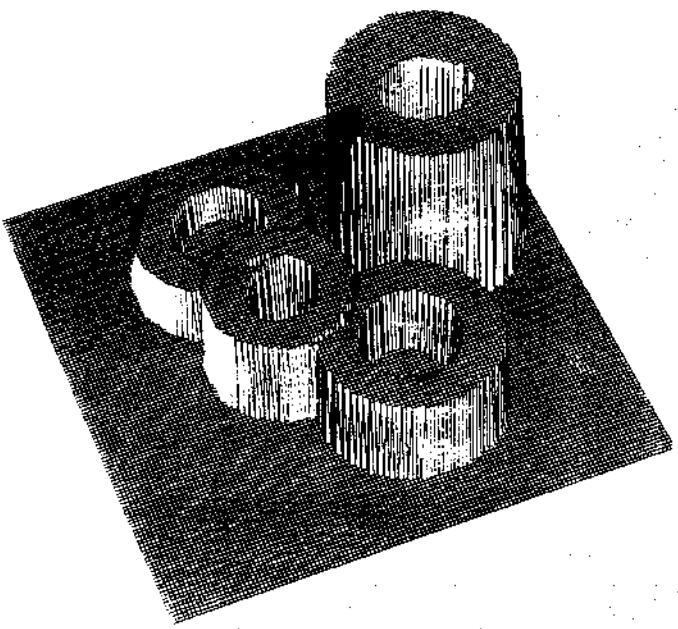


Fig. 9. 3-D disparity map for scene composed of four rings.

TABLE VII
RINGS: COMPARISON OF COMPUTED
AND MEASURED DISTANCES

Object	Actual Distance	Computed Distance	% Error
Table Top	17.50"	17.91"	-2.3%
Two Dark Rings	16.625"	17.38" & 16.86"	+4.5% & -1.4%
Small Ring	16.00"	16.86"	+5.4%
Single Ring	15.00"	15.19"	-3.3%

The fifth and the final experiment analyzes a scene of a circuit board with numerous integrated circuit chips (ICs). Figs. 13 (a)-(b) show a pair of 256×256 stereo images, which were acquired from a distance of about 2.2 feet. This experiment poses many challenges to the stereo matcher. First IC's are mounted very densely on the circuit board, which makes region extraction difficult. Second there are many similar objects that may confuse the stereo matcher. And finally while there are only a few depth levels in the scene, depth variation among them is quite small. For example the height difference between the larger IC's and the smaller IC's is only approximately 0.25 in. Thus the stereo system would be required to resolve depths to 1/4 of an inch for objects lying about 2 ft away. The result of segmentation on the stereo pair is shown in Figs. 13(c)-(d). There are 94 regions extracted from the left image and 104 regions from the right. While not all IC's are extracted as individual regions (e.g., notice that in the segmented left image the cluster of large IC's closer to the right edge of the board was extracted as one composite region), this effect appears consistently in both images. The region-based matcher found correspondences for 40 blobs and all of them were correct. While matching accuracy was perfect, slightly less than half of the extracted regions were matched. The disparity map for the scene is shown in Fig. 14. It indicates that all larger chips were

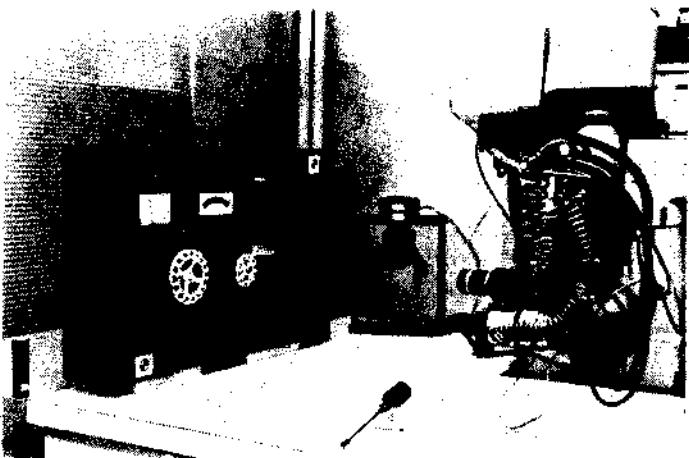


Fig. 10. Experimental set-up showing robot mounted camera and test panel with variety of displays, meters, valves, controls, and switches.

mapped correctly, but the method had difficulty mapping some of the smaller chips. This is partly due to the compact layout of the circuit board. Dense mounting of the chips makes it difficult to extract "meaningful" regions. The actual depth measurements are compared to measured depth and are tabulated in Table IX. While the distance measurements for the composite regions *A* and *B* in the segmented images were quite accurate, many of the smaller chips were assigned distances ranging from 27.02 inches to 25.57 inches. The actual measured distance however was 26.75 inches. Thus the error for smaller chips ranged from -1.0 percent to 4.4 percent.

Matching accuracies of the region-based matcher for the aforementioned experiments are summarized in Table X. In each of the experiments, segmentation performed quite well, and regions associated with the physical objects or their parts were successfully extracted. The six simple and easy-to-extract descriptors proved sufficient for measuring region similarities. Thus the minimum distance rule was able to solve correspondences satisfactorily. Table X shows that the region-based matcher performed perfectly in each experiment. The circuit board experiment shows that while the accuracy of matching remains perfect, small regions may be difficult to match. It should, however, be noted that a high degree of accuracy is more important than a high number of matches. Thus the matching results indicate a high degree of confidence. These experiments also showed that correct matches yield very high overlap ratios. The final analysis of depth measurements indicates that the disparities generated, based upon the normalized overlap ratios, were adequate to provide reasonably high accuracies (± 5 percent) in deriving the distance measures. The accuracies of depth measurements for all the experiments are summarized in Table XI.

The disparities generated and the distances computed are generally quite satisfactory; however small regions and perspective effects in viewing make the assignment of disparity to overlapping areas based upon normalized overlap ratio, limited in accuracy. Therefore, improved methods may be needed to generate finer disparity maps.

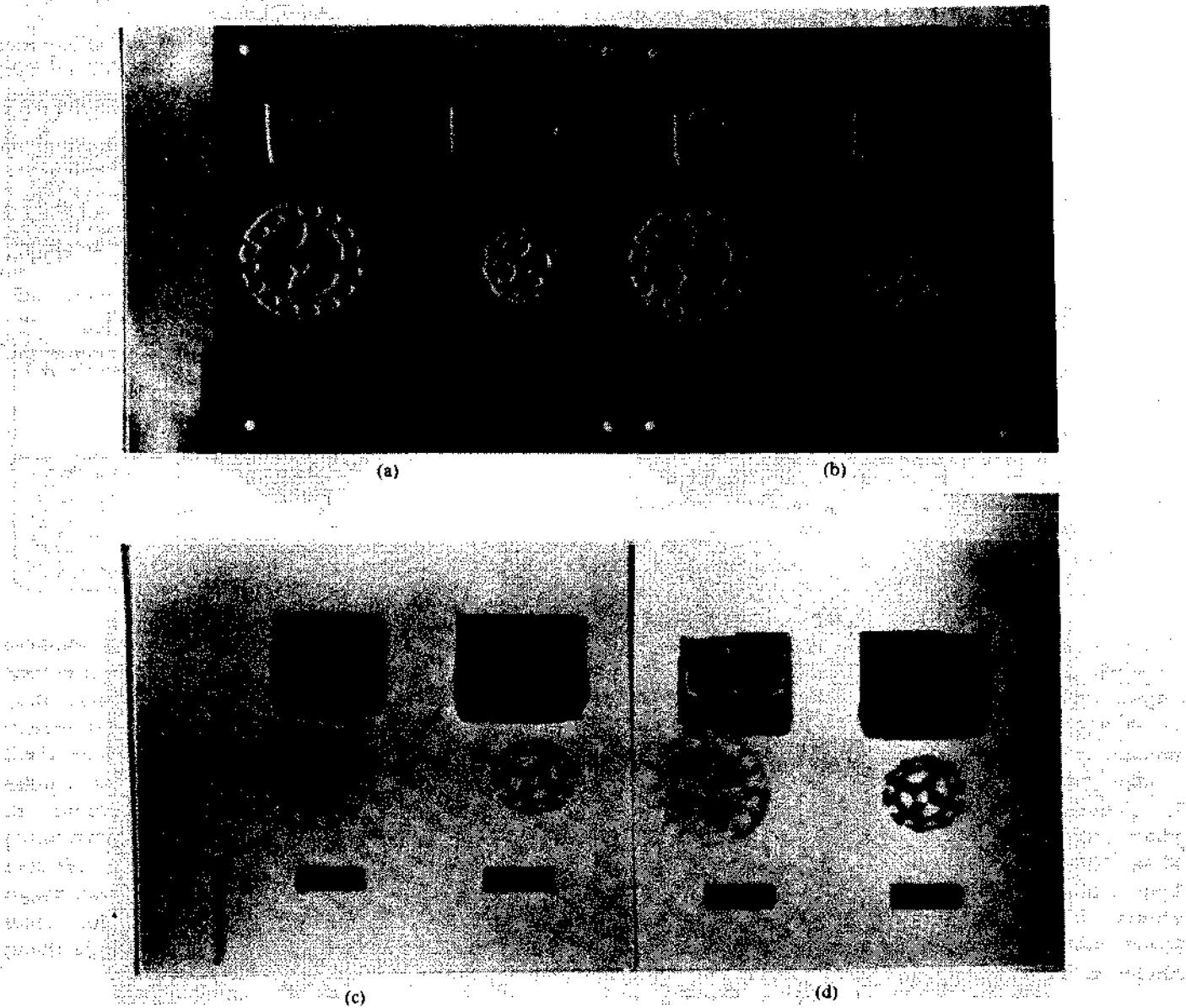


Fig. 11. Scene composed of test panel. The 256×256 stereo image pair is shown in (a) left and (b) right. Results of segmentation are shown in (c) and (d).

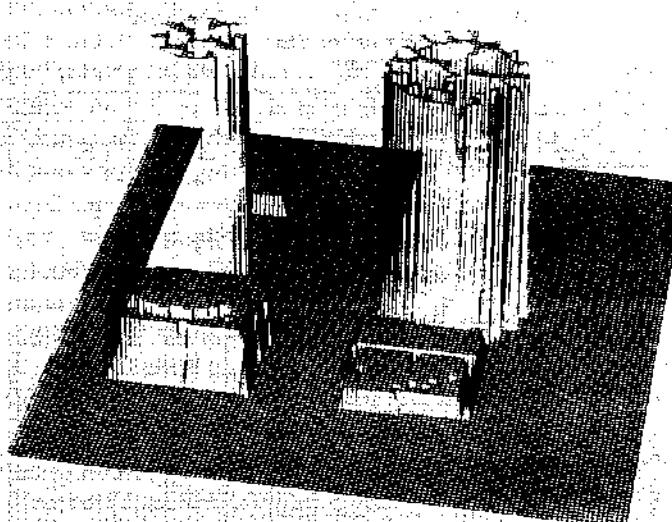


Fig. 12. 3-D disparity map for test panel scene.

TABLE VIII
PANEL: COMPARISON OF COMPUTED AND MEASURED DISTANCES

Object	Actual Distance	Computed Distance	% Error
LCD	31.25"	31.44"	0.6%
Large Valve	27.125"	26.95"	0.6%
Small Valve	27.325"	26.95"	1.4%
Right Meter Base	30.875"	30.8"	0.2%
Left Meter Base	30.875"	31.4"	1.7%

A multilevel, hierarchical disparity generation module, based on matching results of both edge features as well as regions, seemingly offers attractive features.

Finally a practical stereo system must also address the issue of efficiency. A multiuser VAX 11/785 computing environment was used in the aforementioned experiments. The computational times for the experiments are summarized in Table XII. Processing time of about a minute for these experiments without any optimized or specialized

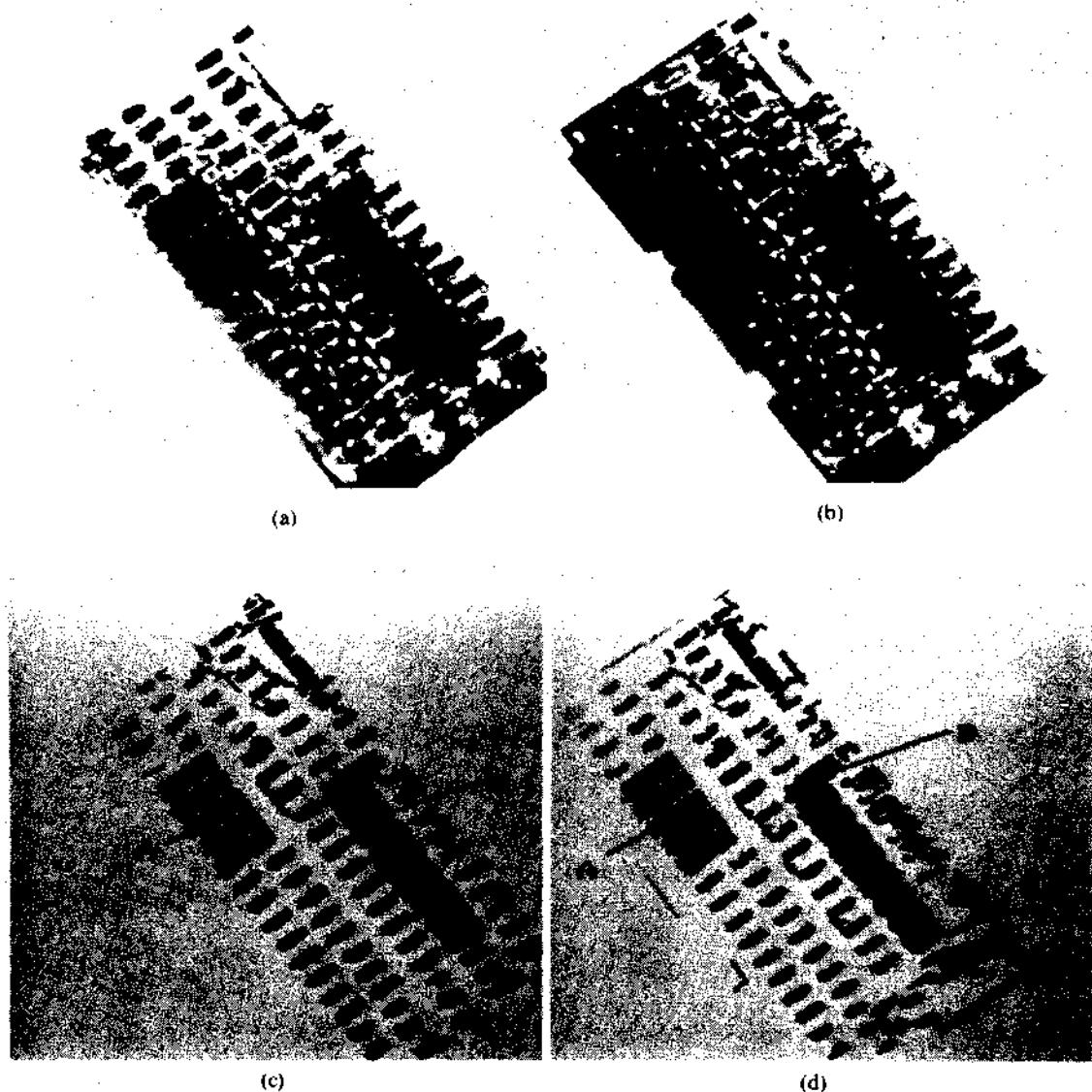


Fig. 13. Scene containing circuit board. The 256×256 stereo image pair is shown in (a) left and (b) right. Results of segmentation are shown in (c) and (d).

TABLE IX
CIRCUIT BOARD: COMPARISON OF COMPUTED
AND MEASURED DISTANCES

Object	Actual Distance	Computed Distance	% Error
Region A	26.50"	26.28"	0.8%
Region B	26.50"	27.02"	-1.9%
Small Chips (see text)	26.75"	27.03" ~ 25.57"	-1.0% ~ -4.1%

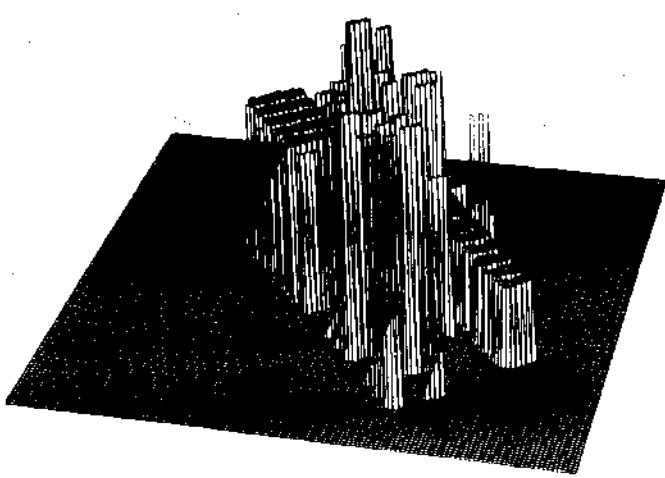


Fig. 14. 3-D disparity map for circuit board scene.

TABLE X
ACCURACY OF REGION-BASED
STEREO MATCHING

Experiment	No. of Blobs in Left Image	No. of Blobs in Right Image	Number of Blobs Matched	Number of Matching Errors Committed	Accuracy Of Matching (%)
Simple scene	5	4	2	None	100
Industrial parts	4	4	4	None	100
Rings	7	7	6	None	100
Panel	17	21	12	None	100
Circuit board	94	104	40	None	100

TABLE XI
ACCURACY OF DEPTH MEASUREMENTS

Experiment	Approximate Imaging Distance (feet)	Absolute %Error in Depth (Max.-Min.)
Industrial parts	1.2	6.0% ~ 2.6%
Rings	1.3	5.1% ~ 1.1%
Panel	2.5	1.8% ~ 0.3%
Circuit board	2.2	4.4% ~ 0.8%

TABLE XII
EFFICIENCY OF REGION-BASED STEREO MODULE

Experiment	No. of Blobs in Left Image	No. of Blobs in Right Image	Maximum Disparity	Processing Time (Seconds)
Simple scene	5	4	50	55.87
Industrial parts	1	1	75	71.53
Rings	7	7	40	86.8
Panel	17	21	75	84.60
Circuit board	94	104	45	85.14

modules, supports the promise of a region-based stereo approach for on-line robotic tasks. The scenes analyzed are quite comparable to the types of scenes appearing in current stereo literature, but we are unable to quantitatively analyze the accuracy of the derived measurements against others, because only a few studies have presented data on actual depth measurements. Also different imaging parameters would make such a comparison difficult. However the accuracy of about ± 5 percent achieved by the region-based approach, within a fraction of the time that most existing stereo algorithms require, demonstrates the practical utility of the region-based approach.

V. CONCLUSION

In this paper we have discussed issues related to the design and development of a binocular stereo approach for a robot vision system. We presented the characteristics of the application domain and identified the needs and goals of a practical depth extraction system for a vision system for robotic applications. The binocular stereo approach offers an attractive, passive means for deriving detailed depth maps for a scene without using specialized illumination or energy sources. Yet high computational costs and rigid input requirements have prevented stereo approaches from being adopted for practical robotic applications. For effective use in this application domain, a stereo system must seek a balance between input flexibility, output sufficiency, and computational complexity. At the heart of a binocular stereo approach lies the task of stereo matching. In this paper we emphasized the importance of semantic content and stability of the primitive used in this matching and introduce the use of homogeneous regions as features in stereo matching. Considering the fewer number of features and the higher discrimination power of the primitive, the region-based matcher is more efficient and accurate

than those using edge-based primitives. A region-based matching technique can utilize both local and global information and thus yield a more globally consistent solution. However while the use of regions makes stereo matching more accurate, reliable, and efficient than edge-based matching, region based matching processes typically yield coarse disparity maps. The generation of accurate and finer resolution disparity maps (and subsequently depth measurements) can be better accomplished using edge-based techniques. Both regions and edges play important, but somewhat complementary, roles in the binocular stereo process. It is therefore critical that an efficient and robust stereo system utilize the most appropriate set of primitives at each stage of the process. In this paper a hierarchical stereo approach that exploits and integrates the power of different primitives at appropriate stages of the stereo process is proposed. We presented several experiments to evaluate the performance of a region-based stereo matcher and a straightforward disparity and depth generation module. It is shown that the approach is efficient, robust, and successful in generating depth measurements within ± 5 percent accuracy.

An effective stereo system for robotic applications should include a component that relies on rich surface texture as well as one that relies on the uniformity of surface details. While more research is required before such a system can be engineered, the direction suggested in this research offers promise. For dealing with typical robotic scenes, a multilevel, hierarchical system, utilizing a hierarchy of primitives appears to be suitable. In such a system, results of stereo matching at higher levels of the hierarchy can be used for guidance at the lower levels, and disparity maps generated at each level can be fused to obtain an accurate and fine resolution disparity map. Such a system would also provide the capability to selectively analyze regions with varying resolutions.

ACKNOWLEDGMENT

The software developed by Chu Xin Chen for segmentation was useful in our studies. Final manuscript preparation was done by Mrs. Janet Smith. We also thank the reviewers for many insightful and valuable comments.

REFERENCES

- [1] A. C. Kak and J. S. Albus, *Handbook of Industrial Robotics*, ch. on "Sensors for intelligent robots." New York: John Wiley, 1985.
- [2] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. New York: McGraw-Hill, 1987.
- [3] H. K. Nishihara, "Practical real-time imaging stereo matcher," *Optical Engineering*, vol. 23, pp. 536-545, Sept. 1984.
- [4] H. K. Nishihara and T. Poggio, "Stereo vision for robotics," in *First Int. Symp. Robotics Res.*, IEEE Press, 1983, pp. 489-505.
- [5] R. A. Jarvis, "A perspective on range finding techniques for computer vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 122-139, Mar. 1983.
- [6] A. C. Kak, *Handbook of Industrial Robotics*, ch. on "Depth perception for robots." New York: John Wiley, 1985.

- [7] G. Stockman, S. Chen, G. Hu, and N. Shrikhande, "Recognition of rigid objects using structured light," in *Proc. 1987 IEEE Int. Conf. Syst. Man Cybern.*, Aug. 1987, pp. 877-883.
- [8] B. K. P. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.
- [9] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [10] K. Ikeuchi, "Determining surface orientations of specular surfaces by using the photometric stereo method," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, pp. 661-669, Nov. 1981.
- [11] R. C. Jain and A. K. Jain, "Review of the NSF range sensing workshop," in *Comput. Vision Pattern Recognition Conf.*, Ann Arbor, MI, May 1988.
- [12] S. T. Barnard and M. A. Fischler, "Computational stereo," *Computing Surveys*, vol. 14, pp. 553-572, Dec. 1982.
- [13] D. Marr and T. Poggio, "A computational theory of human stereo vision," *Proc. Royal Soc. London*, vol. B 204, 1979, pp. 301-328.
- [14] B. Julesz, "Binocular depth perception of computer-generated patterns," *Bell System Tech. J.*, vol. 39, pp. 1125-1162, 1960.
- [15] ———, *Foundation of Cyclopean Perception*. Chicago, IL: Univ. of Chicago Press, 1971.
- [16] W. E. L. Grimson, "Computational experiments with a feature-based stereo algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 17-34, Jan. 1985.
- [17] H. H. Baker and T. O. Binford, "Depth from edge and intensity-based stereo," in *Proc. 7th Int. Joint Conf. AI*, 1981, pp. 631-636.
- [18] G. Medioni and R. Nevatia, "Segment-based stereo matching," *Comput. Vision, Graphics Image Processing*, vol. 31, pp. 2-18, July 1985.
- [19] H. S. Lim and T. O. Binford, "Stereo correspondence: A hierarchical approach," in *DARPA Image Understanding Workshop*, Los Angeles, CA, 1987.
- [20] R. D. Arnold and T. O. Binford, "Geometric constraints in stereo vision," in *Image Processing for Missile Guidance*, pp. 281-292, SPIE, 1980.
- [21] R. D. Arnold, *Automated Stereo Perception*. Ph.D. dissertation, Dept. of Computer Science, Stanford University, CA, 1983.
- [22] J. E. W. Mayhew and J. P. Frisby, "Psychophysical and computational studies towards a theory of human stereopsis," *Artificial Intell.*, vol. 17, pp. 349-385, 1981.
- [23] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby, "PMF: A stereo correspondence algorithm using a disparity gradient limit," *Perception*, vol. 14, pp. 449-470, 1981.
- [24] W. E. L. Grimson, *From Images to Surfaces: A Computational Study of the Human Early Visual System*. Cambridge, MA: MIT Press, 1981.
- [25] W. Hoff and N. Ahuja, "Surface from stereo: Integrating feature matching, disparity estimation, and contour detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-11, pp. 121-136, Feb. 1989.
- [26] R. Mohan, G. Medioni, and R. Nevatia, "Stereo error detection, correction, and evaluation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-11, pp. 113-120, Feb. 1989.
- [27] G. D. Forney Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, Mar. 1973, pp. 268-278.
- [28] R. Nevatia and K. R. Babu, "Linear feature extraction and description," *Comput. Vision Graphics Image Processing*, vol. 13, pp. 257-269, June 1980.
- [29] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, vol. 2. Orlando, FL: Academic Press, 1982.
- [30] R. Nevatia, *Machine Perception*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [31] T. Pavlidis, *Structural Pattern Recognition*. New York: Springer, 1977.
- [32] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 139-154, Mar. 1985.
- [33] Y. C. Kim and J. K. Aggarwal, "Positioning three-dimensional objects using stereo images," *IEEE J. Robotics Automation*, vol. RA-3, pp. 361-373, Aug. 1987.
- [34] S. T. Barnard, "A stochastic approach to stereo vision," Tech. Rep. 373, SRI International, Menlo Park, CA 94025, Apr. 1986.
- [35] N. M. Nasrabadi, Y. Liu, and J. Chiang, "Stereo vision correspondence using a multichannel graph matching technique," in *Proc. 1988 IEEE Int. Conf. Robotics Automation*, Apr. 1988, pp. 1804-1809.
- [36] M. M. Trivedi, C. Chen, and S. B. Marapane, "Vision system for robotic inspections," *Computer*, vol. 22, no. 6, pp. 91-97, June 1989. Special issue on Autonomous Intelligent Machines.



Suresh B. Marapane is a doctoral student in electrical and computer engineering at the University of Tennessee, Knoxville. He earned the B.S. and M.S. degrees, both in electrical and computer engineering from Louisiana State University.

His research interests include computer vision, robotics, and concurrent computing. He is a member of Phi Kappa Phi and Eta Kappa Nu, and a student member of SPIE.

Mohan M. Trivedi (S'76-M'79-SM'86) for photograph and biography please see page 1335 of this TRANSACTIONS.

Shape from Texture: Integrating Texture-Element Extraction and Surface Estimation

DOROTHEA BLOSTEIN, MEMBER, IEEE, AND NARENDRA AHUJA, SENIOR MEMBER, IEEE

Abstract—A perspective view of a slanted textured surface shows systematic changes in the density, area, and aspect-ratio of texture elements. These apparent changes in texture element properties can be analyzed to recover information about the physical layout of the scene. However, in practice it is difficult to identify texture elements, especially in images where the texture elements are partially occluded or are themselves textured at a finer scale. To solve this problem, it is necessary to integrate the extraction of texture elements with the recognition of scene layout. We present a method for identifying texture elements while simultaneously recovering the orientation of textured surfaces. A multiscale region detector, based on measurements in a ∇^2G (Laplacian-of-Gaussian) scale-space, is used to construct a set of candidate texture elements. True texture elements are selected from the set of candidate texture elements by finding the planar surface that best predicts the observed areas of the candidate texture elements. Results are shown for a variety of natural textures, including waves, flowers, rocks, clouds, and dirt clods.

Index Terms—Integration, multiscale structure, natural textures, perspective view, region detection, shape from texture, surface orientation, texture elements, texture gradients, texture homogeneity, three-dimensional vision.

I. INTRODUCTION

TEXTURE variation due to projective distortion provides important cues for recovering the three-dimensional structure of the surfaces visible in an image [11]. A uniformly-textured surface undergoes two types of projective distortions during the imaging process. Firstly, an increase in the distance from the surface to the viewer causes a uniform compression of increasingly large areas of surface onto a fixed area of image. Secondly, foreshortening (due to the angle between the surface and the image plane) causes an anisotropic compression of the texture. These texture variations provide information about the relative distances and orientations of the textured surfaces in an image.

A primary goal of the work reported in this paper is to demonstrate the feasibility of extracting useful measures of texture gradients from images of natural scenes. A ma-

Manuscript received September 15, 1988; revised January 30, 1989. Recommended for acceptance by A. K. Jain. This work was supported by the Air Force Office of Scientific Research under Grant AFOSR 86-0009 and by the Eastman Kodak Company.

D. Blostein was with the Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801. She is now with the Department of Computing and Information Science, Queen's University, Kingston, Ont. K7L 3N6, Canada.

N. Ahuja is with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801.

IEEE Log Number 8929956.

ajor challenge in texture analysis is to identify texture scale consistently. Natural surfaces exhibit a rich hierarchy of textures and subtextures. All texture measurements are prone to distortion due to the presence of subtexture, since the imaging process captures more subtexture details for close texture samples than for distant ones. As discussed in Section II-A-1, existing shape-from-texture algorithms do not address the problem of scale. The algorithms presented here provide good surface-orientation estimates even in the face of significant sub- and supertexture.

A. Texels and Texture Gradients

The term *texel*, short for *texture element*, denotes the repetitive unit of which a texture is composed. "Texel" refers to the physical texture element in the real world as well as to the appearance of the texture element in the image. In cases where the distinction must be made, we use the phrases *physical texel* versus *image texel*. Distance and foreshortening changes alter the image texel, but not the physical texel.

Projective distortion affects many texture features, and hence gives rise to a variety of texture gradients. Consider first the idealized texture of Fig. 1(a); a planar surface covered with nonoverlapping circular disks of constant size. The disks project as ellipses in the image. The major axis of each ellipse is perpendicular to the tilt,¹ whereas the minor axis is parallel with the tilt. Scanning the image from bottom to top (in the direction of tilt), the apparent size of the major axes decreases linearly, due to increasing distance from the viewer (the *perspective gradient*). However, the apparent size of the minor axes decreases quadratically: in addition to the distance scaling, the minor axes are foreshortened. Thus the eccentricity of the ellipses increases in the tilt direction (the *aspect-ratio gradient*). Similarly, the area of the ellipses decreases fastest in the direction of tilt (the *area gradient*). This is accompanied by an increase in the density of the ellipses (the *density gradient*). In this idealized texture, the uniformity in the size, shape and placement of the texture elements leads to pronounced texture gradients.

¹We express surface orientation in terms of two angles, *slant* and *tilt* [23]. Slant, ranging from 0° to 90° , is the angle between the surface and the image plane. Tilt, ranging from 0° to 360° , is the direction in which the surface normal projects in the image; a tilt of 0° indicates that distance to the viewed surface increases fastest toward the right side of the image. The synthetic textures shown in Fig. 1(a) illustrate the definition of slant and tilt.

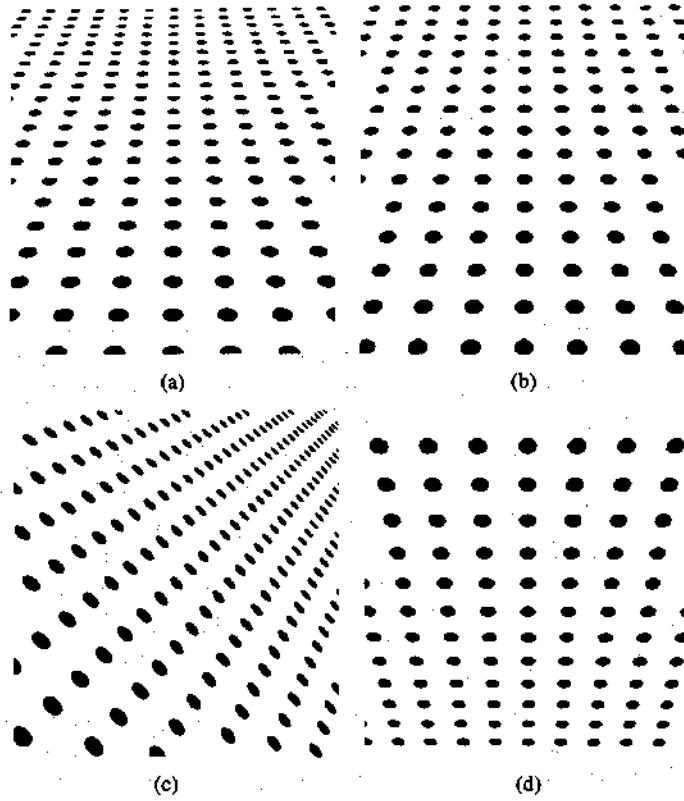


Fig. 1. Synthetic textures illustrating various slants and tilts. Slant is the angle between the textured surface and the image plane. Tilt is the direction in which the surface normal projects in the image. (a) Slant 60°, tilt 90°. (b) Slant 50°, tilt 90°. (c) Slant 60°, tilt 45°. (d) Slant 45°, tilt 270°.

Natural textures are much less regular than the idealized texture of Fig. 1(a); therefore the texture gradients are not as easily observed. Natural textures display considerable variability of texel size, shape, coloration and density. Physical texels are typically three-dimensional, unlike the disks portrayed in Fig. 1(a). Highlights, shadows, and occlusions between texels result. Also, physical texels have a complex structure. In contrast to a uniform synthetic disk, a physical texel changes in appearance as the distance to the camera decreases: more subtexture is visible for the nearby texels than for the distant texels. Supertexture regions arise when small image texels, corresponding to distant physical texels, blur into larger regions of relatively uniform gray level. These factors make it difficult to identify texture elements and extract texture gradients from real images.

The importance of various texture gradients has been studied extensively in the psychology literature (see, for example, [7], [19], [21], [22]). Vickers [24] was among the first to advocate an approach involving accumulation of evidence from multiple texture gradients. Cutting and Millard [9] attempt to quantify the relative importance of various texture gradients. They test human subjects on synthetically generated textures, which are designed to contain only a subset of the normally-occurring texture gradients. Experimental results show that for slant judgments of flat surfaces the perspective and density gradients are more important than the aspect-ratio gradient,

whereas in the perception of curved surfaces the aspect-ratio gradient is dominant, with perspective and density gradients having little impact.

II. THE INFERENCE OF SURFACE SHAPE FROM TEXTURE GRADIENTS

We now turn to a discussion of the basic requirements for a system that infers surface-shape from texture gradients. In Section II-A we argue that correct measurement of texture gradients requires explicit identification of image texels, especially when textures show three-dimensional relief, when texels exhibit significant subtexture, or when it is unknown *a priori* which texture gradients carry surface-shape information. In Section II-B we address the problem of texel identification. Texture elements cannot be identified in isolated image areas since texels are defined only by the repetitive nature of the texture as a whole. Therefore, the identification of texture elements is best done in parallel with the estimation of the shape of the textured surface.

A. The Importance of Texel Identification

The extraction of texels is an essential step in measuring texture gradients, because it permits correct analysis of textures containing subtexture. Explicit texel identification also provides the basis for a unified treatment of the various texture gradients (area gradient, density gradient, aspect-ratio gradient) that may be present in an image. Previous researchers have avoided texel identification because it is quite difficult to do in real images.² Instead, indirect methods are used to estimate texel features. We give below several examples of such methods, and indicate why these methods may give erroneous results.

1) Previous Work: Most previous shape-from-texture algorithms use indirect methods to estimate texel features, by making some assumptions about the nature of texture elements. For example, texel density may be estimated by measuring edge density, under the assumption that all detected edges correspond to the borders of texture elements [1]³, [2], [14], [20]. Alternatively, texture elements may be assumed to have uniform edge direction histograms; surface orientation can then be estimated from any deviations from isotropy observed in the distribution of edge directions [10], [13], [26]. However, the directional-isotropy assumption is very restrictive; for example, it does not hold true in images containing elongated texels such as waves. Texture coarseness and directionality may be characterized using Fourier domain measurements [3], ignoring the effect of super- and subtextures. Various researchers [12], [15], [16] have developed

²Ohta *et al.* [18] use the observed areas of pairs of texels to obtain vanishing points. However, the method has been tested only on synthetic texture images. The problem of extracting texels from natural images is not addressed.

³The theoretical analysis in this paper is based on texel area. In application to real images, edge-density is used instead, under the assumption that the edge detector finds texel boundaries.

algorithms to analyze textures containing parallel and perpendicular lines. Most natural textures are too irregular to be analyzed in this way.

All of these methods may encounter problems when applied to complex natural textures seen under natural lighting conditions. Since texels are not identified and explicitly dealt with, it becomes difficult to distinguish between responses due to texels and those due to other image features, such as subtexture. It appears to be necessary to recognize the texture elements before the various measures can be computed as intended.

Consider, for example, methods based on measuring edge density. If these algorithms are applied to edges produced by an edge-detector, the measurements are made inaccurate by contributions from subtexture and supertexture edges. Fig. 2(a)-(c), which shows the response of an edge detector to several texture images,⁴ illustrates that it would be incorrect to interpret all of the detected edges as boundaries of texture-elements. Additional edges result from subtexture; these edges are not artifacts of this particular edge detector, since they are clearly present in the original images. Many natural textures have a hierarchical physical structure that causes observed edge density to be nearly constant throughout the image: edges from subtexture and subsubtexture are observed to whatever detail the camera resolution permits.

In order to measure edge-density as intended by [2] and [14], it is necessary to eliminate subtexture edges. This cannot simply be done by applying a global threshold, since the contrast of texels far from the camera is comparable to the contrast of subtexture features in the foreground. Aloimonos [2] distinguishes between a "strong segmentation" (finding texels) and a "weak segmentation" (finding edges, where the edges are supposed to be texel boundaries), and states that weak segmentation is easy to obtain (apply any general-purpose edge detector). We argue that correct weak segmentation is not possible without simultaneously performing a strong segmentation: in order to eliminate all edges except those that arise from texel boundaries, one has to in effect identify the texels.

2) Multiple Texture Gradients: Explicit texel identification offers an additional advantage: texels provide a unifying framework for examination of the various texture gradients (such as gradients of apparent texel area, aspect ratio, density etc.) that may be present in an image. A given image may exhibit a combination of texture gradients. In general, the accuracy of the surface information obtainable from these gradients varies from image to image.⁵ Since it is not known in advance which texture gra-

dients are useful for determining the three-dimensional orientation of surfaces, a shape-from-texture system should evaluate the information content of different types of gradients in a given image, and use an appropriate mix of these gradients for surface estimation.

B. Integration of Texel Identification and Surface-Shape Estimation

Texel identification is difficult because texels have tremendously varied shapes, sizes and gray-level characteristics. A texel cannot be identified in isolation, since texels are only defined by the repetitive nature of the texture as a whole. In order to determine if an image region is a texel, it is necessary to test if the region has properties consistent with the properties of many other image texels, i.e. whether the image region is part of a *texture field*.

We use the term *texture field* (or *field of texels*) to denote a collection of image texels that exhibit one or more consistent texture gradients. Consistency is defined with respect to a perspective view of a given surface. It is not uncommon for a single image to contain several texture fields. First, many images are composed of closely associated bright and dark texture fields which arise from lighting effects. For example, the aerial view of houses in Fig. 5(a) contains a field of bright texels composed of the houses and a field of dark texels composed of the shadows cast by the houses. Second, it is possible for physically separated textured surfaces to be spatially interleaved in an image. This is strikingly illustrated by the birds-over-water image shown in Fig. 7(a). Finally, the same physical surface may contain different texture fields: an aerial view of a residential neighborhood shows one texture field consisting of houses and another texture field consisting of trees.

Texels can only be identified in the context of a texture field, where consistent texture gradients must exist across the whole field. The consistency of a texture gradient can only be evaluated for a particular surface shape and orientation. Thus, texel identification must be combined with surface estimation.

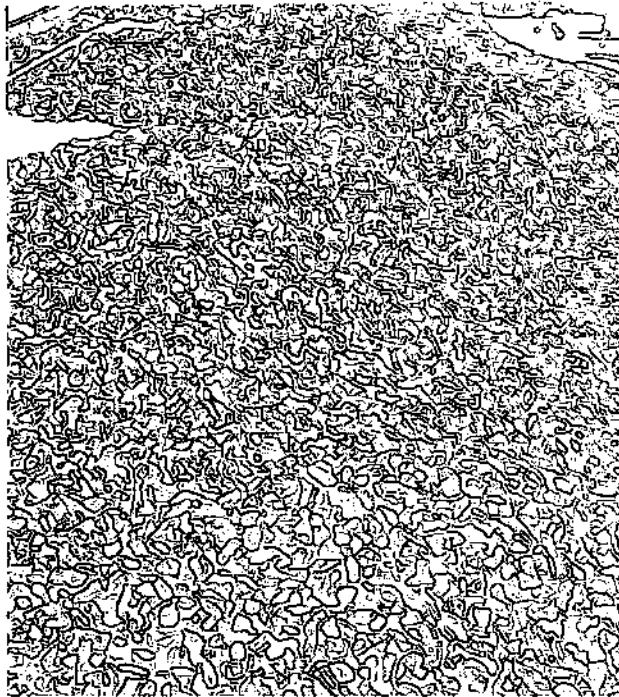
C. Overview

Motivated by the above discussion, we now summarize the requirements for an ideal shape-from-texture algorithm, and the extent to which our work meets these requirements. Many open problems remain.

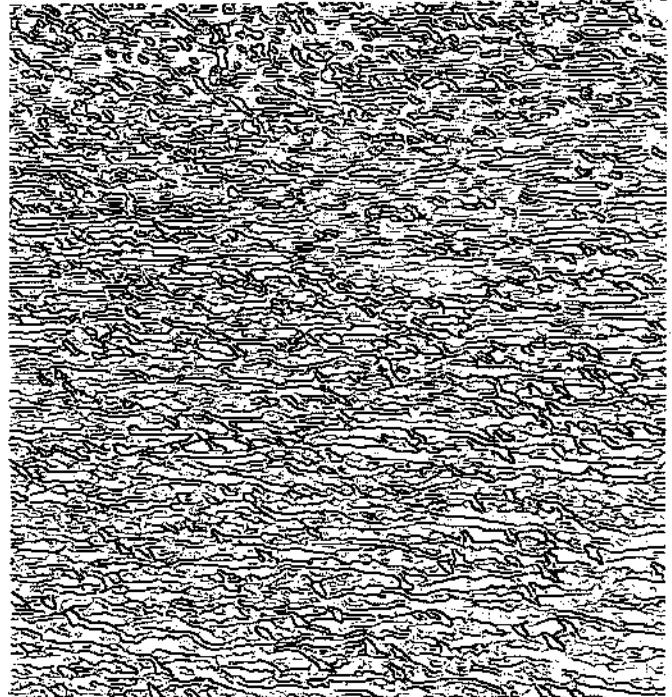
⁴We use an edge operator described by Nevatia and Babu [17]. Six 5-by-5 edge masks at different orientations are used; the mask giving the highest output at each pixel is recorded. The edges are thinned by suppressing non-maxima perpendicular to the edge directions.

⁵This may be illustrated by the following examples. It is common for physical texels to be fairly uniform in size and shape, but for the gaps between the texels to be much less uniform [Figs. 7(a), 19(a), 23(a), and 25(a)]. In these images, it is more accurate to infer a three-dimensional

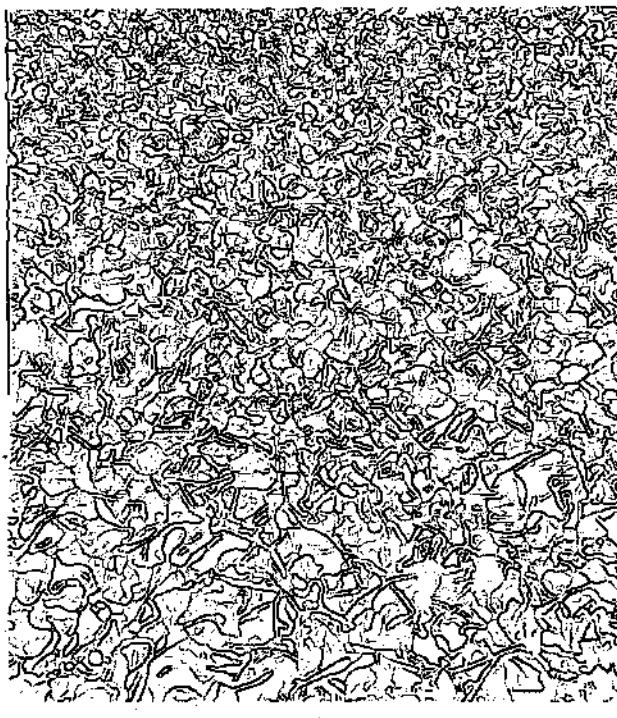
surface from the area and aspect-ratio gradients than from the density gradient or the gradient of spacings between texels. As a second example, the potential accuracy of the aspect-ratio gradient is higher in textures where the physical texels are separated by gaps than in textures where the physical texels overlap and occlude one another [the lily pads in Fig. 25(a) show a much better aspect-ratio gradient than do the rocks in Fig. 3(a)]. Thirdly, for the water hyacinths of Fig. 29(a), the random three-dimensional arrangement of the leaves makes the aspect-ratio gradient very weak, while the area gradient is still quite significant. Finally, in images with partial occlusions [Figs. 13(a) and 15(a)], the perspective gradient (length of the unforeshortened texel dimension) is more accurate than the area gradient: if only part of a texel is occluded, the apparent texel area is decreased, whereas the complete unforeshortened dimension may remain in view.



(a)



(b)



(c)

Fig. 2. Edges extracted from three texture images. Only a subset of the detected edges are boundaries of texture elements. If edge density is to be effective in capturing the texture gradient, all edges that do not correspond to texel boundaries must be removed. Such edge removal cannot be accomplished without, in effect, performing an identification of texture elements. (a) Edges from the rock-pile image shown in Fig. 3(a). (b) Edges from the image of birds flying over water, shown in Fig. 7(a). (c) Edges from the water hyacinths image shown in Fig. 29(a).

Texel Identification: As discussed in Section II-A, texel identification is important for correct shape-from-texture analysis. In general, physical texels can give rise to complex gray-level patterns. However, it is difficult to test for

repetitive patterns of arbitrary gray-level configuration. In our implementation, we restrict image texels to be regions that have small gray-level variation relative to a neighborhood of their size. Under this restriction, a physical

texel can give rise to several image texels: often the physical repetitive unit of a texture contains both bright and dark regions [for example, the sunflowers in Fig. 15(a)]. We treat the bright and dark image texels as two texture fields, which we analyze separately.

Texture Gradients: A shape-from-texture system should test each image for the presence of various texture gradients (area gradient, aspect-ratio gradient, density gradient), and combine these various sources of information to produce a surface estimation. In our current implementation, the only texture gradient we test for is a gradient in texel area.⁶ However, we do independent analyses of the area-gradients in positive-contrast and negative-contrast image regions; these two types of regions may correspond to foreground and background, or to different portions of the physical texels. Some images contain bright texels on dark backgrounds, other images contain dark texels on bright backgrounds, and yet other images have no visible "background" region because texels are densely placed. Measurements of foreground regions are often more accurate than measurements of background regions, since texel area tends to be less variable than texel spacing.

Surface Estimation: Ideally, a system tests for texture gradients produced by a variety of surface shapes (planar, cylindrical, spherical etc.), and is able to locate discontinuities in depth and surface orientation. Much work remains to be done in this area. We restrict ourselves to fitting a single planar surface to the entire image. This is a common restriction in current implementations of shape-from-texture algorithms, although a theoretical treatment of the nonplanar case has been performed by [1], and has been tested on synthetic images.

Three-Dimensional Texel Effects: For accurate image analysis, it is necessary to model the effects of three-dimensional relief on observed texture gradients. We do not address this problem in our current implementation: the equations we use for expected area-gradients are derived under the assumption that texels do not have three-dimensional relief, i.e., that they are "painted" on the textured surface. For textures with relief, this assumption results in a redefinition of texels: ideally, only those parts of physical texels that are parallel to the underlying surface are recognized as defining a texture field. The detected consistency is reduced if the texels have parts that are not parallel to the surface. If the texel relief is regular (so that the texels are mutually parallel), the detected gradient may still be significant and nearly correct recovery of surface orientation may be possible. For example, we obtain satisfactory results on the sunflower image of Fig. 15, where the texels are not parallel to the surface.

Although some theoretical treatment of the foreshortening of textures with relief exists (e.g., [15]), no one has

⁶Our extraction of texel shape is not accurate enough to permit useful measures of aspect-ratio. We also cannot measure texel density accurately because we do not extract all of the texels. Ongoing research into improved texel-extraction will permit the analysis of several texture gradients.

addressed how in-plane texels could be distinguished from out-of-plane texels in real images.

Multiple Textures: A general shape-from-texture system must be able to handle images containing multiple textures; the system must perform texture segmentation as well as shape-from-texture estimation. To solve this problem it is necessary to separate texture variations due to distance and foreshortening effects from texture variations due to a boundary between different physical textures. Our current implementation does not address this problem: each of our images contains only a single texture. Ongoing research is aimed at extending the method to apply to images containing multiple textures.

The rest of the paper describes our two-step algorithm for texel identification and surface estimation. In the first step, we use a multiscale region detector to construct a set of candidate texels (Section III). In the second step we use surface-fitting to identify the true texels from among the candidates, while simultaneously constructing an approximation to the shape of the textured surface (Section IV). The second step thus enforces perspective viewing constraints to select texels. Section V presents results for a variety of images of textured natural scenes.

III. IDENTIFYING CANDIDATE TEXELS: MULTISCALE REGION DETECTION

We now turn to a description of the multiscale region detector used to construct the set of candidate texels. The set of candidate texels includes all image regions that have small gray-level variation relative to a neighborhood of their size. These image regions may be of any shape and size, and they may be nested, since there is no *a priori* way to distinguish texture regions from subtexture and super-texture regions.

To simplify the problem of extracting regions of arbitrary shapes and sizes, we assume that each region can be represented as a union of overlapping circular disks. Large disks define the rough shape of a region, with overlapping smaller disks capturing finer shape details such as protrusions and concavities. In Section III-A we derive a method of extracting all circular image regions of relatively uniform gray level. Section III-B discusses how sets of overlapping disks are used to form candidate texels.

The region detector is based on the image response to convolution with $\nabla^2 G$ filters over a range of scales. Related work includes [27] (a scale-space representation of $\nabla^2 G$ zero-crossings) and [8] (a representation of $\nabla^2 G$ peaks and ridges over a range of scales⁷). We find circular image regions of uniform gray level by convolving the image with $\nabla^2 G$ masks over a range of scales, and comparing the convolution output to that expected for an ideal circular disk of constant gray level. Here we present a brief summary of the region detection algorithm; a more detailed discussion may be found in [6].

⁷Crowley and Parker use a difference-of-Gaussian operator, which is a discrete approximation to $(\partial/\partial\sigma)G$ and hence to $\nabla^2 G$. By the diffusion equation, $\nabla^2 G = (1/\sigma)(\partial/\partial\sigma)G$.

A. A Closed Form Expression for the $\nabla^2 G$ Response of a Disk

The algorithm for uniform-region extraction is based on calculations of the $\nabla^2 G$ and $(\partial/\partial\sigma)\nabla^2 G$ responses of a disk image. Given a function $I(x, y)$ which specifies the intensity of an image, the $\nabla^2 G$ response of this image at (x, y) is given by the following convolution:

$$\begin{aligned} \nabla^2 G(x, y) * I(x, y) &= \iint_{-\infty}^{+\infty} \frac{2\sigma^2 - (u^2 + v^2)}{\sigma^4} e^{-(u^2+v^2)/2\sigma^2} \\ &\cdot I(x-u, y-v) du dv. \end{aligned} \quad (1)$$

Mathematical analysis of the response of the $\nabla^2 G$ filter to most images is difficult because the convolution integrals of (1) do not have closed form solutions. However, a closed-form solution can be derived for the center point of a circular disk of constant intensity. We analyze the $\nabla^2 G$ response at the center of an ideal circular disk in the continuous domain; to generate the $\nabla^2 G$ convolution of digitized images, we sample the $\nabla^2 G$ filter values and perform a discrete convolution. The image of a disk of diameter D and contrast C is defined by

$$\text{disk image: } I(x, y) = \begin{cases} C & \text{if } x^2 + y^2 \leq D^2/4 \\ 0 & \text{elsewhere.} \end{cases} \quad (2)$$

Using this definition of $I(x, y)$ in (1), and setting x and y to zero, we find [6] that at the disk center

$$\nabla^2 G \text{ response} = \frac{\pi CD^2}{2\sigma^2} e^{-D^2/8\sigma^2} \quad (3)$$

$$\frac{\partial}{\partial\sigma} \nabla^2 G \text{ response} = \frac{\pi CD^2}{2} \left(\frac{D^2}{4\sigma^3} - \frac{2}{\sigma^3} \right) e^{-D^2/8\sigma^2}. \quad (4)$$

Dividing these expressions, we solve for the diameter D and contrast C of the disk:

$$\begin{aligned} D &= 2\sigma \sqrt{\sigma \left(\frac{\partial}{\partial\sigma} \nabla^2 G * I \right) / (\nabla^2 G * I) + 2} \\ C &= \frac{2\sigma^2}{\pi D^2} e^{D^2/8\sigma^2} (\nabla^2 G * I) \end{aligned} \quad (5)$$

where the convolutions are evaluated at the center of the disk.

B. Extracting Candidate Texels in Real Images

We construct an approximation of image texels by first fitting disks to uniform image regions, and then forming unions of connected disks. (An alternative approach is presented by [25]. They extract texture elements by convolving the image with a $\nabla^2 G$ filter and then selecting components of above-threshold pixels that have suitable geometrical properties, such as compactness.) For the first step, we use (5) to estimate disk diameter and disk contrast from the $\nabla^2 G * I$ and $(\partial/\partial\sigma)\nabla^2 G * I$ values at the

center of a region. Disks are fit at the extrema of the $\nabla^2 G * I$ images. The disks fit to local maxima have positive contrast (regions brighter than the surround), whereas the disks fit to local minima have negative contrast (regions darker than the surround).

We use a range of filter sizes. For a region R in image I , local extrema in $\nabla^2 G * I$ occur at the center of R when the $\nabla^2 G$ filter size approximately matches the region-diameter. Thus, to fit disks as accurately as possible, we accept a disk only if the computed diameter D is close to the $\nabla^2 G$ filter size used to detect the disk.

Parts (b) of Figs. 3-30 illustrate the result of this disk-fitting for the positive-contrast and negative-contrast regions of each image. Implementation details for the disk-fitting are as follows.

1) Compute the convolutions $\nabla^2 G * I$ and $(\partial/\partial\sigma)\nabla^2 G * I$ for the following six σ values: $\sqrt{2}$, $2\sqrt{2}$, $3\sqrt{2}$, $4\sqrt{2}$, $5\sqrt{2}$, and $6\sqrt{2}$. (The center lobes of the six $\nabla^2 G$ filters have diameters of 4, 8, 12, 16, 20, and 24 pixels, respectively.) To compute $\nabla^2 G * I$ for a particular σ value, the image is convolved with a mask whose coefficients are taken from

$$\frac{2\sigma^2 - r^2}{\sigma^4} e^{-r^2/2\sigma^2}$$

To compute $(\partial/\partial\sigma)\nabla^2 G * I$ for a particular σ value, the image is convolved with a mask whose coefficients are taken from

$$\frac{6r^2\sigma^2 - r^4 - 4\sigma^4}{\sigma^7} e^{-r^2/2\sigma^2}.$$

2) Mark the locations where disks will be fit. To analyze the positive-contrast regions of the original image, mark all local maxima in the $\nabla^2 G * I$ images. To analyze the negative-contrast regions of the original image, mark all local minima in the $\nabla^2 G * I$ images. Local maxima and minima are computed relative to a 3×3 neighborhood.

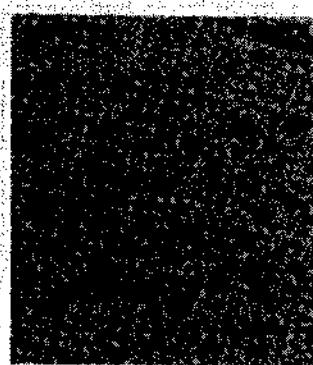
3) At each marked location, use the measured $\nabla^2 G * I$ and $(\partial/\partial\sigma)\nabla^2 G * I$ values in (5) to compute a disk diameter and disk contrast. Accept the disk only if $w - 2 \leq D \leq w + 2$, where $w = 2\sqrt{2}\sigma$ is the diameter of the center lobe of the $\nabla^2 G$ filter. Form a single set of disks by taking the union of the disks detected at the various filter sizes.

After the disk-fitting is completed, we form unions of overlapping disks to construct candidate texels. Overlapping disks form concavities. There is no *a priori* way to tell whether a set of disks should be split at a concavity or not: some concavities arise at the border between two neighboring texels; at other times the concavities are part of the shape of an individual texel. Thus both possibilities are included in the list of candidate texels. The implementation details are as follows:

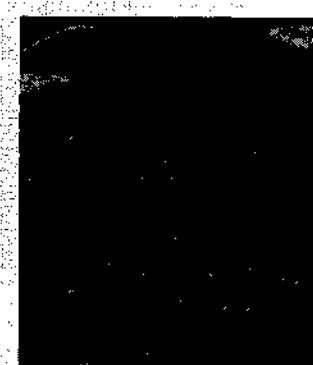
To form the list of candidate texels, extract all subsets of disks that are spatially connected and contain no concavities greater than 90° . If a concavity is in



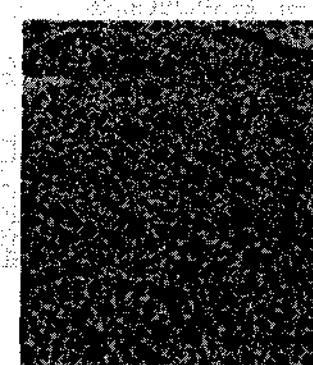
(a)



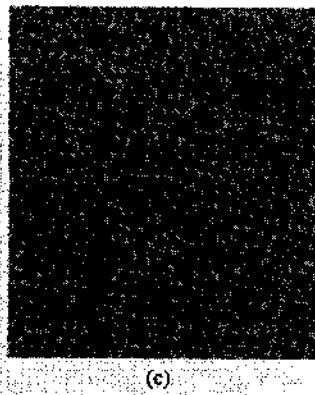
(b)



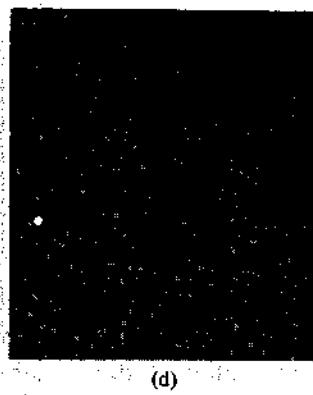
(c)



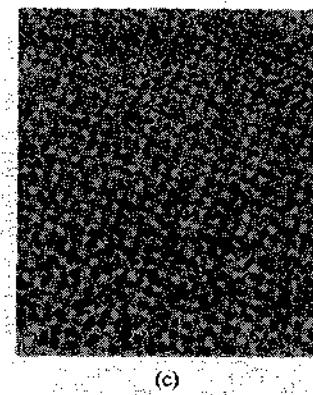
(d)



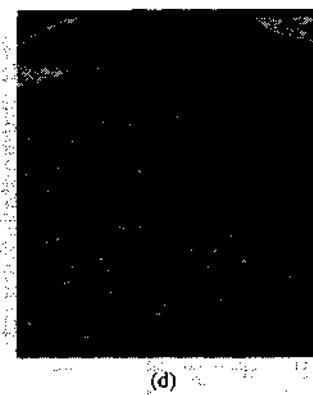
(e)



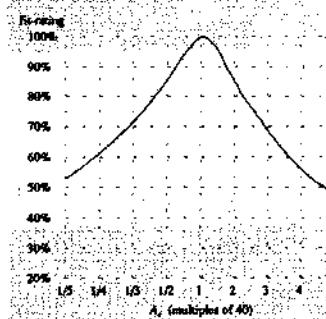
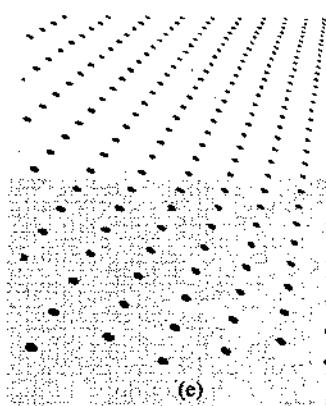
(f)



(g)

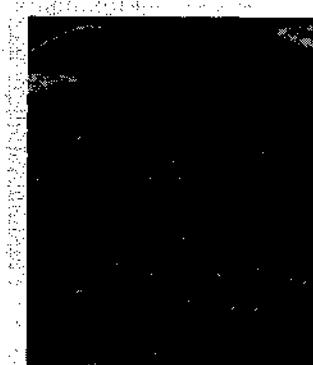


(h)

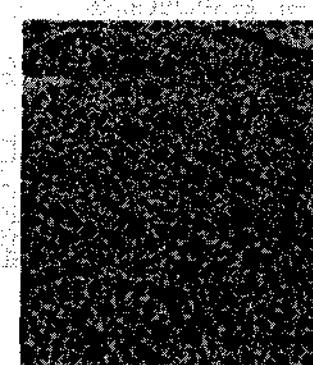


(g)

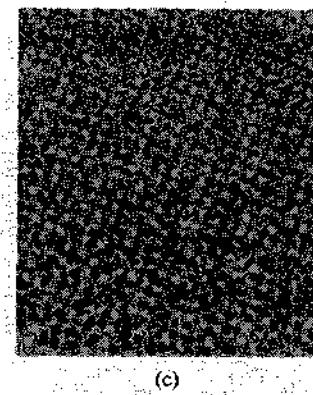
Fig. 3. (a) A rock pile. (b) Disks corresponding to positive-contrast regions of relatively uniform gray level. Disks are shown with a darkness proportional to the contrast of the region. (c) Extracted texels. These are all regions (sets of overlapping disks) having area within a factor of two of the area expected by the best planar fit ($A_c = 40$, slant 62.5° , tilt 65°). The texels that fit the plane most closely are printed darkest. (d) The texels superimposed on a dark reproduction of the original. (e) Synthetic image to illustrate the planar fit $A_c = 40$, slant 62.5° , tilt 65° . (f) and (g) Rating of various possible planar fits. In (f) slant and tilt are varied while A_c is constant at 40. In (g) A_c is varied while slant and tilt are constant at 62.5° and 65° , respectively.



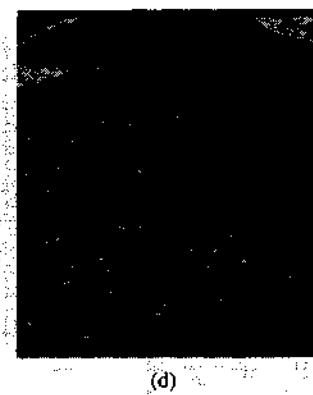
(a)



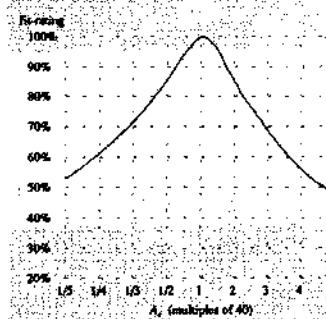
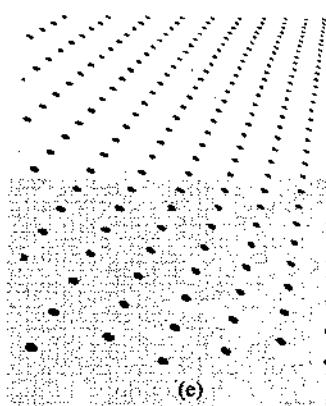
(b)



(c)

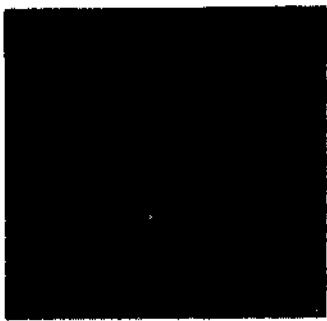


(d)

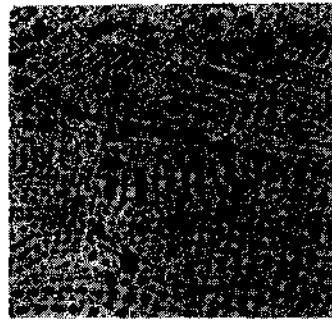


(g)

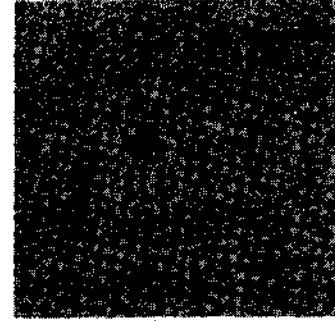
Fig. 4. (a) A rock pile. (b) Disks corresponding to negative-contrast regions of relatively uniform gray level. Disks are shown with a darkness proportional to the contrast of the region. (c) Extracted texels. These are all regions (sets of overlapping disks) having area within a factor of two of the area expected by the best planar fit ($A_c = 40$, slant 60° , tilt 75°). The texels that fit the plane most closely are printed darkest. (d) The texels superimposed on a bright reproduction of the original. (e) Synthetic image to illustrate the planar fit $A_c = 40$, slant 60° , tilt 75° . (f) and (g) Rating of various possible planar fits. In (f) slant and tilt are varied while A_c is constant at 40. In (g) A_c is varied while slant and tilt are constant at 60° and 75° , respectively.



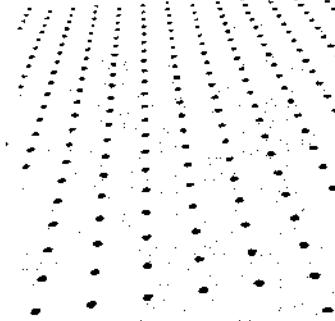
(a)



(b)

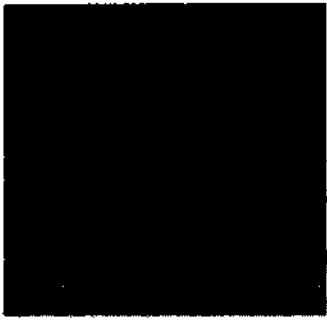


(c)

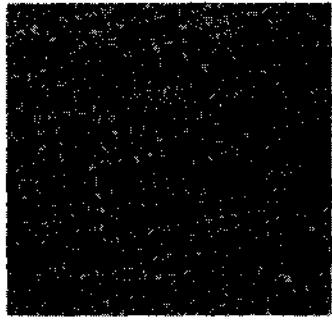


(d)

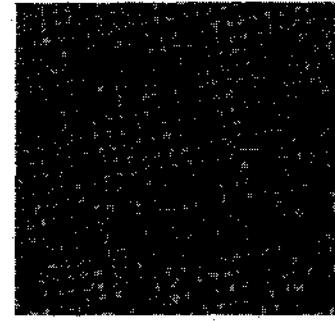
Fig. 5. Aerial view of Levittown, PA: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



(a)



(b)



(c)

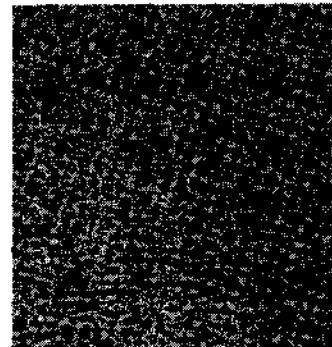


(d)

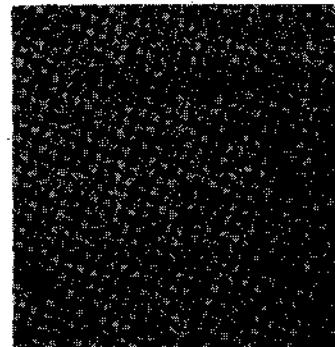
Fig. 6. Aerial view of Levittown, PA: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



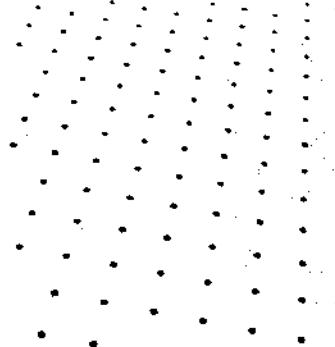
(a)



(b)

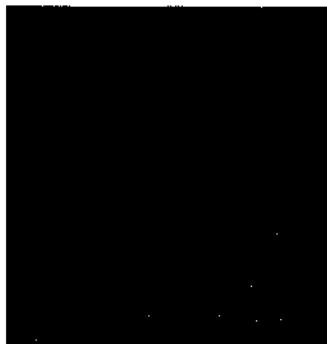


(c)

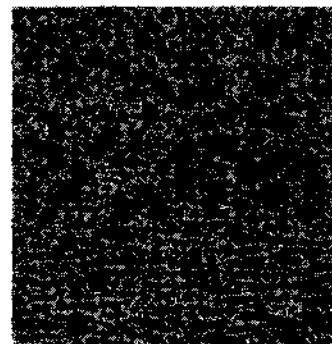


(d)

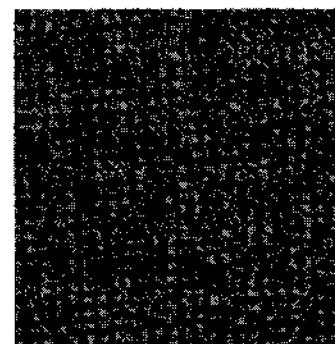
Fig. 7. Snow geese over Back Bay: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



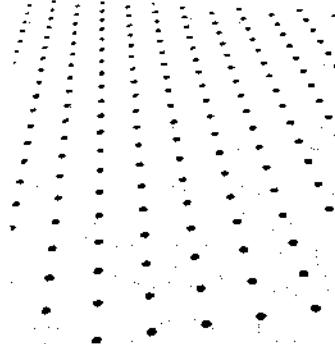
(a)



(b)



(c)

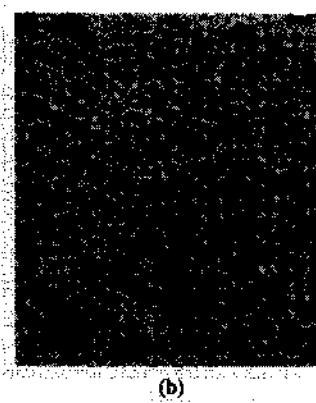


(d)

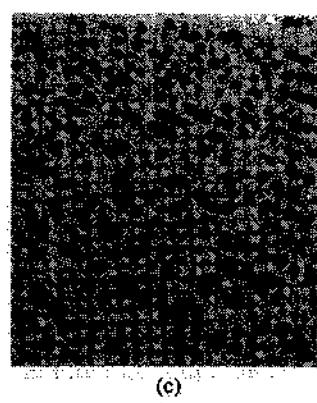
Fig. 8. Snow geese over Back Bay: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



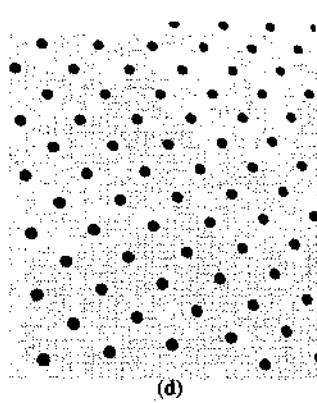
(a)



(b)



(c)



(d)

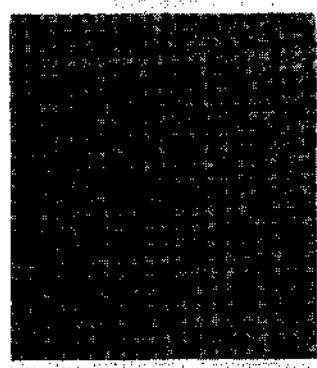
Fig. 9. Prayer at a mosque: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



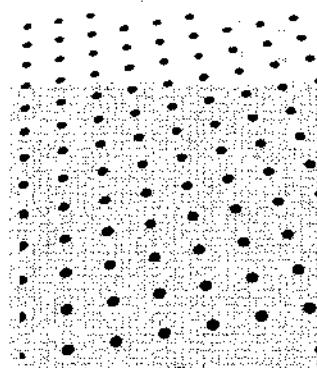
(a)



(b)



(c)

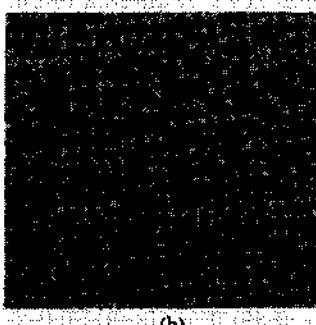


(d)

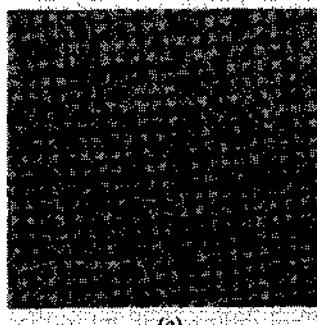
Fig. 10. Prayer at a mosque: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



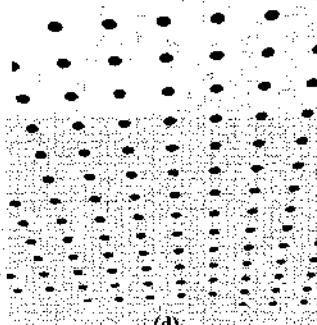
(a)



(b)

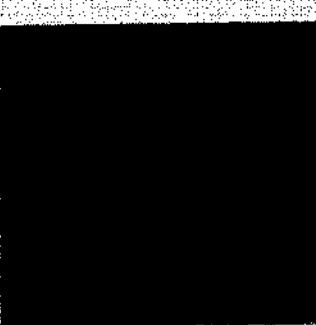


(c)

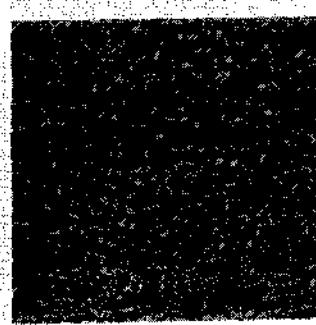


(d)

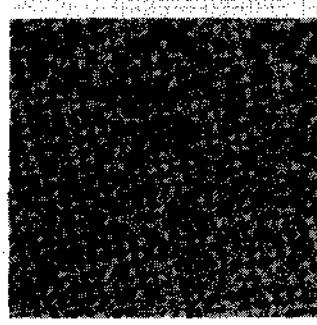
Fig. 11. Fleecy clouds: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



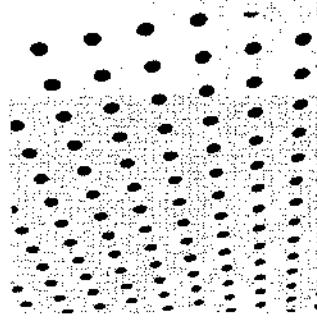
(a)



(b)



(c)

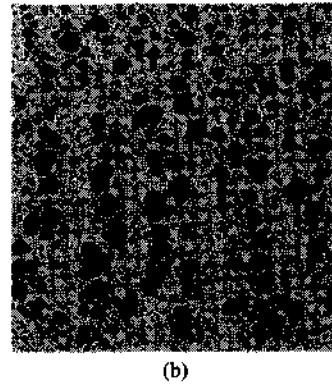


(d)

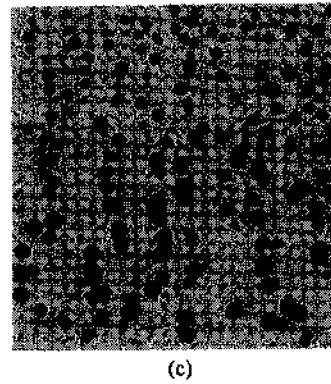
Fig. 12. Fleecy clouds: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



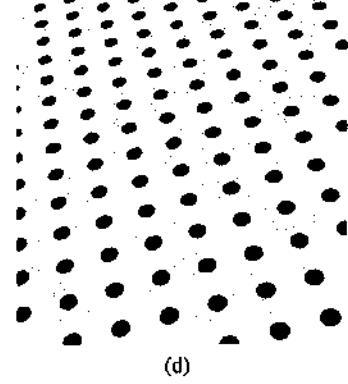
(a)



(b)



(c)

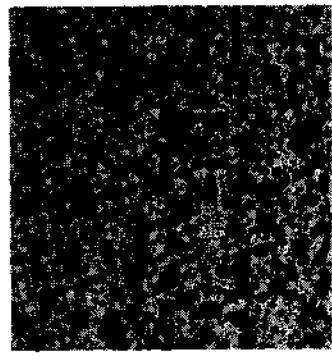


(d)

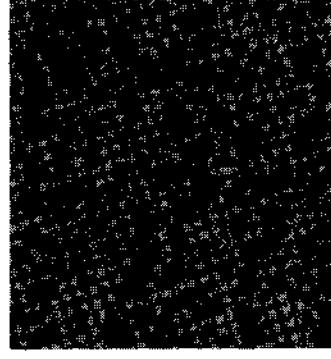
Fig. 13. Audience at a 3-D movie: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



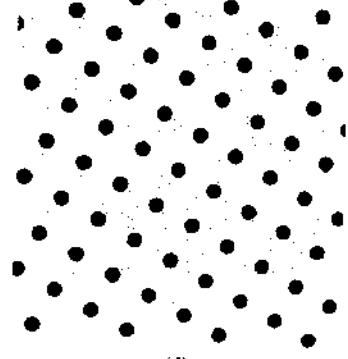
(a)



(b)



(c)

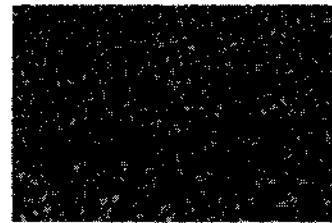


(d)

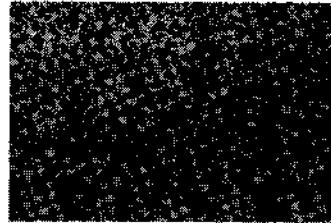
Fig. 14. Audience at a 3-D movie: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



(a)



(b)



(c)

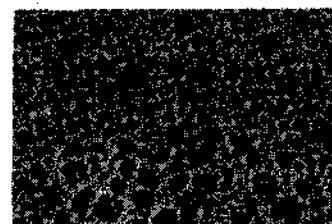


(d)

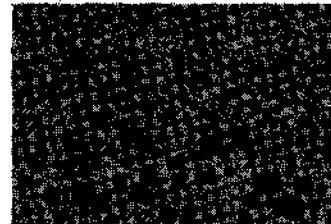
Fig. 15. Sunflowers: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



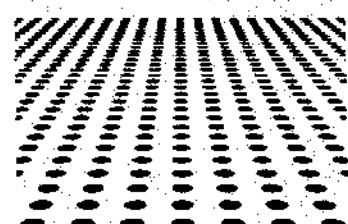
(a)



(b)



(c)

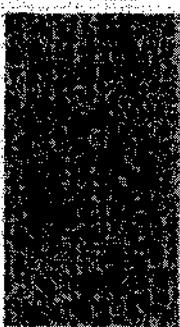


(d)

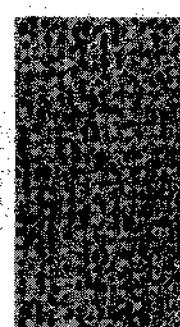
Fig. 16. Sunflowers: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



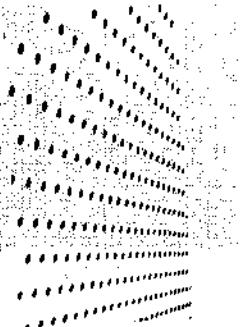
(a)



(b)

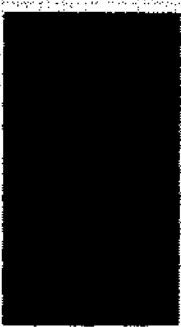


(c)

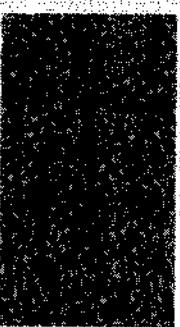


(d)

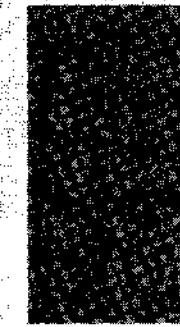
Fig. 17. Tree trunk: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



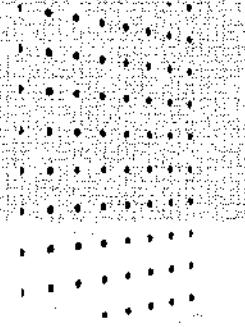
(a)



(b)

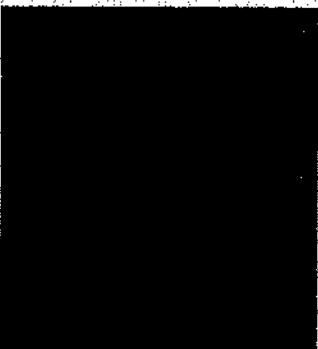


(c)

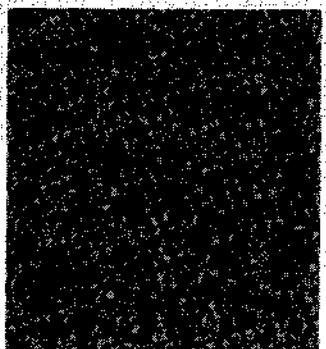


(d)

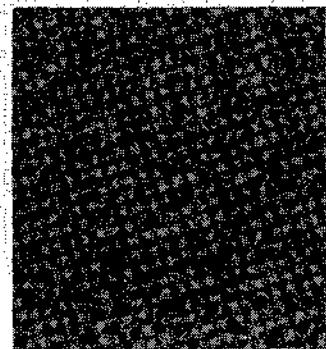
Fig. 18. Tree trunk: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



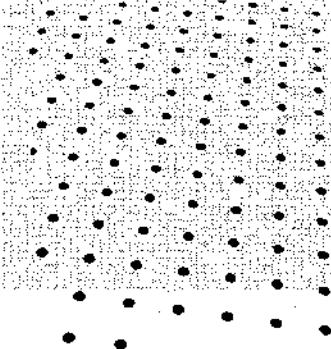
(a)



(b)

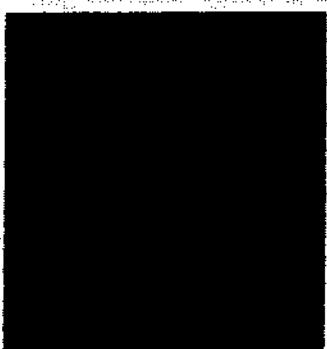


(c)

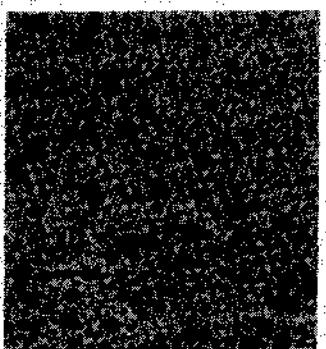


(d)

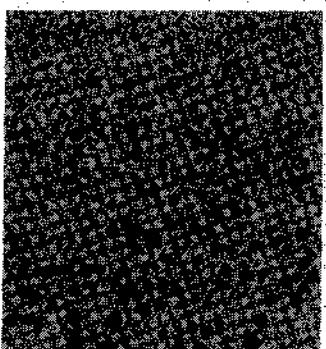
Fig. 19. Bathers on the Ganges: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



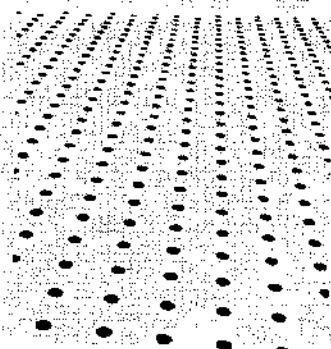
(a)



(b)



(c)

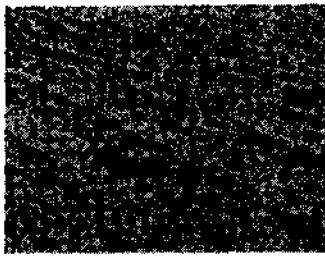


(d)

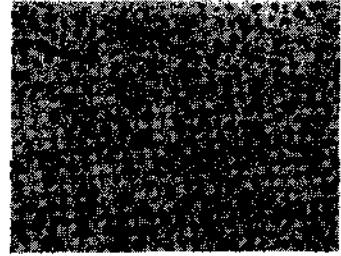
Fig. 20. Bathers on the Ganges: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



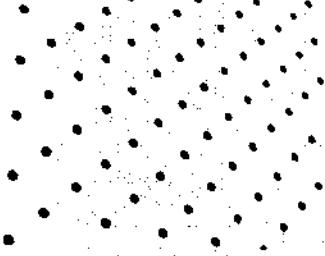
(a)



(b)

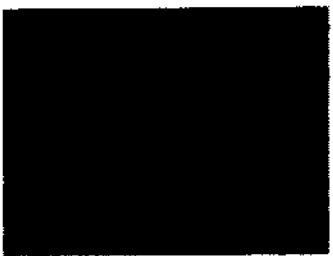


(c)

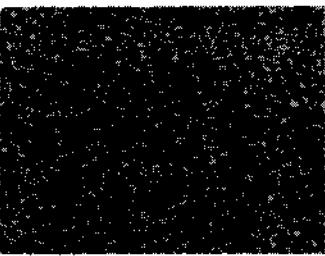


(d)

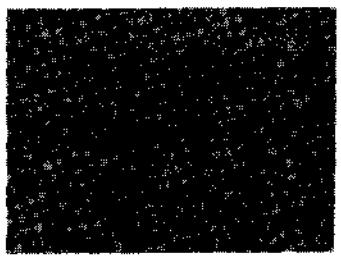
Fig. 21. A plowed field: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



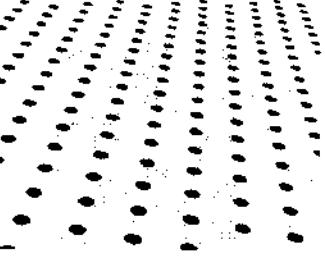
(a)



(b)



(c)

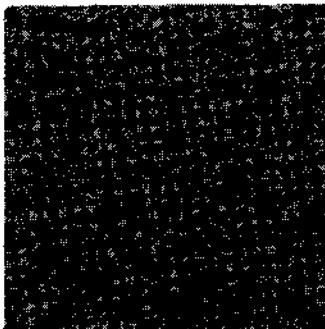


(d)

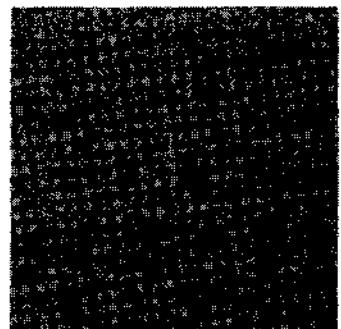
Fig. 22. A plowed field: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



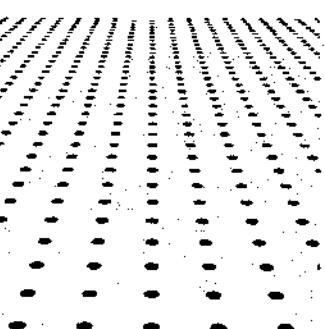
(a)



(b)



(c)

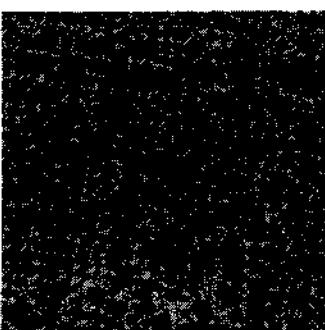


(d)

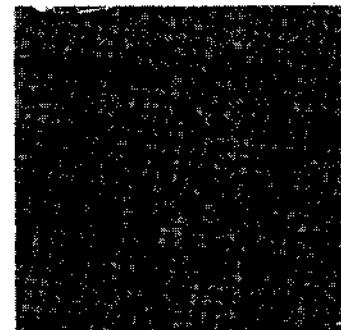
Fig. 23. A field of flowers: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



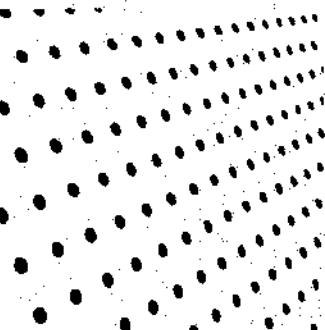
(a)



(b)



(c)



(d)

Fig. 24. A field of flowers: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.

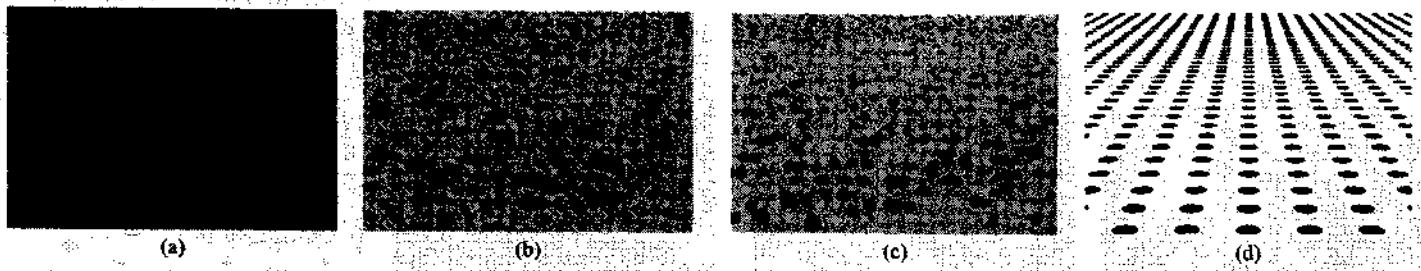


Fig. 25. Water lilies: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.

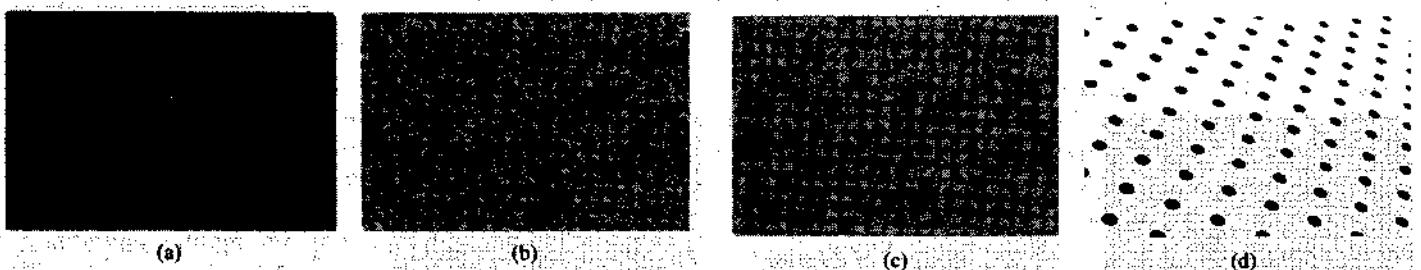


Fig. 26. Water lilies: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.

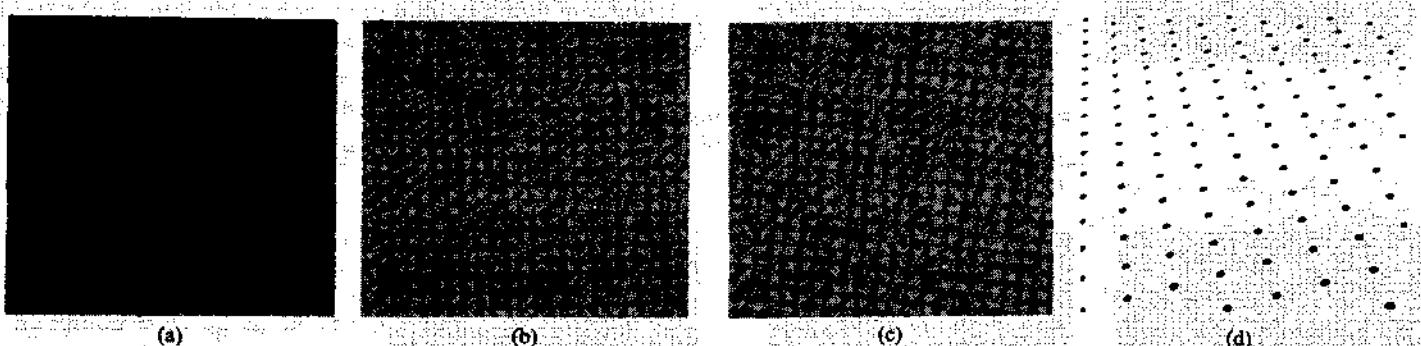


Fig. 27. Ripple marks in a shallow sea: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.

the range 50° – 90° , use the disks to form three candidate texels⁸: one large region consisting of all the disks, and two smaller regions resulting from splitting the large region at the concavity.⁹ Mark mutual exclusion between candidate texels that share a disk: at most one of them can contribute support to a planar fit and be chosen as a true texture element.

⁸The particular values 50° and 90° are not critical; we have found that the range 50° – 90° is large enough to capture the regions of interest and yet small enough to prevent a combinatorial explosion in the number of candidate texels generated.

⁹Region splitting is implemented as follows. We begin with a set P of overlapping disks, which together cover an image region R . The largest concavity in R is found by computing the angles formed by every pair of neighboring disks on the border of R . Suppose that X and Y are two neighboring disks on the border of R , and that they form a concavity that causes a split into smaller, more convex regions. The concavity is split by 1) removing X from P and repeating the above process, and then 2) removing Y from P and repeating the above process.

IV. SURFACE ESTIMATION AND TEXEL IDENTIFICATION

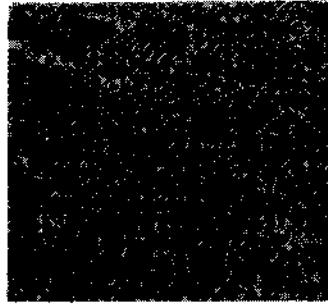
Our goal in analyzing image texture is to find a spatial layout of homogeneously-textured surfaces that could result in the given image texture. We do this by testing many spatial layouts and choosing the one that is the most consistent with a maximal subset of the candidate texels. The surface parameters are determined at the same time that the true texels are chosen from among the candidates.

A. The Expected Distribution of Texel Areas for a Planar Surface

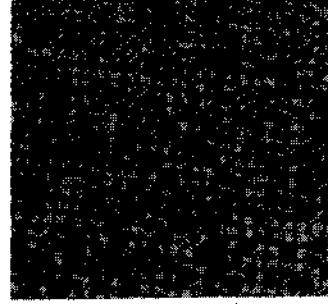
The current implementation is restricted to fitting a single planar surface to the image, based on the observed areas of the candidate texels. In order to find a planar fit to the candidate texels, we need to know the distribution of texel areas that occurs in an image of an idealized textured plane. To derive this relationship, we assume a



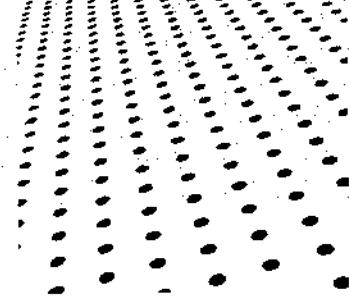
(a)



(b)



(c)

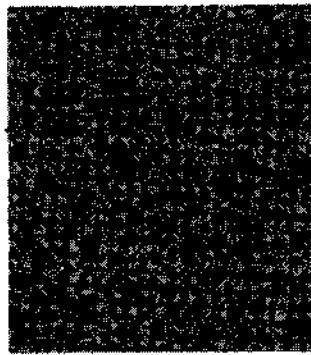


(d)

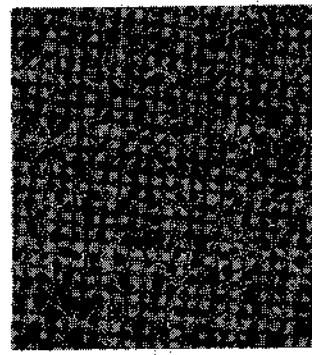
Fig. 28. Ripple marks in a shallow sea: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



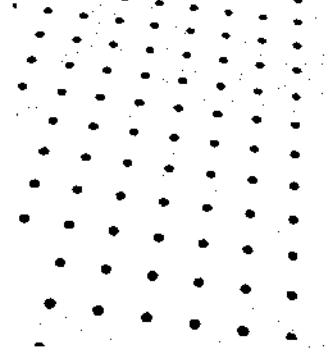
(a)



(b)



(c)

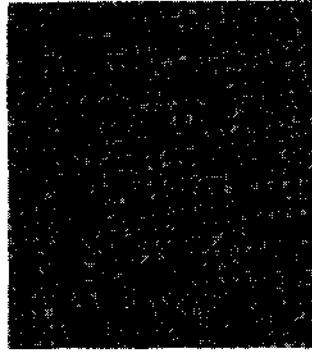


(d)

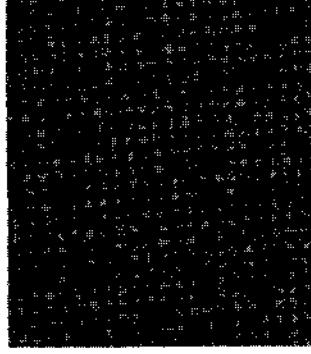
Fig. 29. Water hyacinths: positive contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.



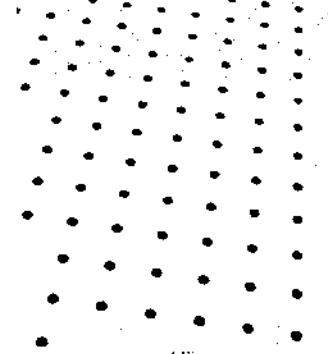
(a)



(b)



(c)



(d)

Fig. 30. Water hyacinths: negative contrast texture. (a) Original image, (b) regions detected, (c) texels extracted, and (d) synthetic display of recovered orientation.

planar textured surface covered with identical texels, where the texels show no three-dimensional relief (the texels are "painted" on the surface). Natural textures are typically more complicated: they are composed of highly variable texels that show three-dimensional relief. Our experiments show that the equations derived from consideration of idealized textures are useful for analyzing a variety of natural textures as well (Section V).

We derive two expressions to describe the size of image texels. The first expression characterizes the foreshortened image-texel dimension F_i ; this is the texel dimension

parallel to the tilt (Fig. 31). The second expression characterizes the unforeshortened image-texel dimension U_i ; this is the texel dimension perpendicular to the tilt. Combining these we obtain an expression for A_i , the expected image-texel area.

As illustrated in Fig. 31, an image location is specified by (X, Y) , in a normalized coordinate system which does not depend on the number of pixels in the image: X and Y are zero at the image center, and are -1 or 1 at the image border. (For notational simplicity, we are assuming square images.)

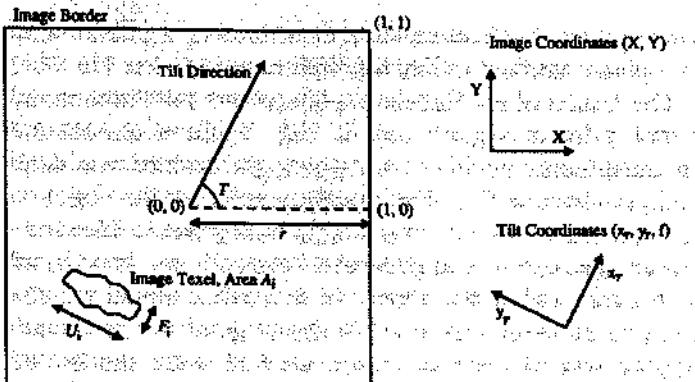


Fig. 31. Image and tilt coordinate systems.

The camera parameters we use are the focal length f and the physical width of the film r . The final expressions use only the ratio r/f , which is a measure of the field-of-view of the camera lens. Telephoto lenses have a low value of r/f , whereas fish-eye lenses have a high value of r/f .

The slant and tilt of the textured plane are denoted by S and T , respectively. Simple derivations may be obtained by defining a *tilt coordinate system*, (x_T, y_T, z_T) , with origin at the focal point, the z_T axis perpendicular to the image plane, and the x_T axis in the tilt direction. The view direction is along the positive z axis; thus this is a left-handed coordinate system. A point (X, Y) in image coordinates is transformed to tilt coordinates by

$$x_T = X \cos T + Y \sin T$$

$$y_T = X \sin T + Y \cos T$$

$$z_T = f$$

Shown in Fig. 32 is the x_T-z_T plane, which is perpendicular to both the image plane and the textured plane. The angle $\theta = \tan^{-1}(x_T(r/f))$ for an image location with tilt coordinates (x_T, y_T, f) ; given raw image coordinates (X, Y) ,

$$\theta = \tan^{-1}((X \cos T + Y \sin T)(r/f)). \quad (6)$$

From the geometry in Fig. 32 we derive that [4]

$$F_t = F_p \frac{f}{g} \cos S (1 - \tan \theta \tan S)^2.$$

The only approximation made is that θ does not change significantly across the texel. To eliminate the dependence on the surface-depth g , we calculate F_t in terms of F_c , the foreshortened dimension of a texel at the image center:

$$F_t = F_c (1 - \tan \theta \tan S)^2. \quad (7)$$

Similarly, we derive that U_t , the unforeshortened image-texel dimension, is related to U_c , the unforeshortened dimension of a texel at the image center by

$$U_t = U_c (1 - \tan \theta \tan S). \quad (8)$$

If the image texel has a compact shape, the area A_i of the image texel is proportional to the product of F_t and U_t ,

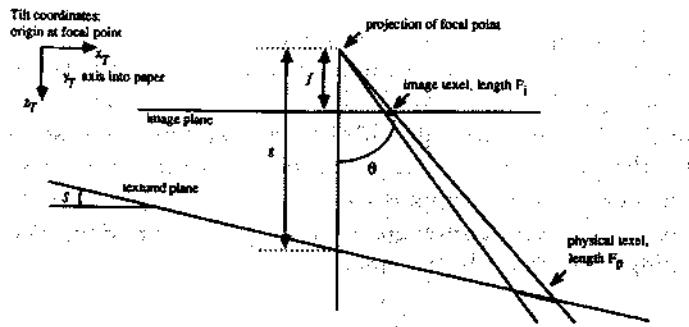


Fig. 32. The x_T-z_T plane. Since y_T is constant, both the image plane and the textured plane are perpendicular to the drawing.

(For example, if the physical texel is a circle, the image texel is an ellipse with $A_i = (\pi/4)F_t U_t$; if the physical texel is a rectangle, the image texel is a parallelogram with $A_i = F_t U_t$.) Thus, $A_i = kF_t U_t$, where k is a constant of proportionality which depends upon the texel shape. Therefore A_i , the area of a texel at location (X, Y) in the image¹⁰ is related to A_c , the area of a texel at the image center by

$$A_i = A_c (1 - \tan \theta \tan S)^3. \quad (9)$$

Substituting θ from (6) into (9), we see that the following values are needed to predict the texel area anywhere in the image:

- A_c , the area that would be measured for a texel located at the center of the image.
- S and T , the slant and tilt of the textured plane.
- The ratio r/f , which is related to the field-of-view of the camera lens.

In our work we assume that the field-of-view is known. The other three quantities (A_c, S, T) form the parameter space we search to find the best planar fit for a given texture image.

B. Fitting a Planar Surface to the Candidate Texels

Having extracted candidate texels from an image of a textured surface, we find the orientation of the textured plane that best agrees with the observed areas of the candidate texels. A planar surface is characterized by the triple (A_c, S, T) , where A_c is the texel area expected in the image center, S is the slant, and T is the tilt. In order to find the best planar fit for the image texture, we discretize the possible values of A_c, S , and T , and evaluate each possible planar fit. For each choice of (A_c, S, T) , (9) gives the expected texel area at each image location. These expected areas are compared to the region areas actually occurring in the image, and a fit-rating is computed for the plane. The plane that receives the highest fit-rating is selected as the estimate of the textured surface. The candidate texels that support the best planar fit are interpreted as true image texels.

¹⁰As pointed out by a reviewer, this result can also be derived from equations (2-1) and (3-2) in [18]; they also use the approximation that θ is constant across the texel.

The rating of a planar fit is computed by summing contributions from all the candidate texels. If a texel at location (X, Y) has an area which is close to the ideal area in (9), then the texel provides a large amount of support for the planar fit. As the actual texel area deviates from the ideal area, the support for the planar fit decreases; we use an exponentially-decreasing weighting function. The rating of a planar fit is computed as

$$\text{fit rating} = \sum_{\text{all regions}} (\text{region area}) \cdot |\text{region contrast}| e^{-(\text{region-fit})^2/4}$$

$$\text{where region-fit} = \frac{\max(\text{expected area, actual area})}{\min(\text{expected area, actual area})} \quad (10)$$

The region-fit is 2.0 for a candidate texel that is either half as big or twice as big as the size predicted by the planar fit.

We begin with a coarse fit, in which the (A_c, S, T) space is searched at sparse locations: A_c (in units of pixels) takes on the values $\{10, 20, 40, 80, 160, 320, 640\}$, S takes on the values $\{0^\circ, 5^\circ, 10^\circ, \dots, 70^\circ, 75^\circ, 80^\circ\}$, and T takes on the values $\{0^\circ, 20^\circ, 40^\circ, \dots, 300^\circ, 320^\circ, 340^\circ\}$. To refine the planar fit, a more detailed search of the (A_c, S, T) space is done in the neighborhood of the best plane from the coarse fit: S is changed in increments of 2.5° , T in increments of 5° , and A_c in increments of less than 25 percent. As illustrated in parts (e) of Figs. 3 and 4 the fit-rating values change smoothly as a function of A_c , slant, and tilt. The plane that receives the highest fit-rating is selected as the best estimate of the textured surface. True image texels are those regions that have an area close to the area expected by the best planar fit.

V. APPLICATION OF THE ALGORITHM TO REAL IMAGES

We have conducted experiments with a variety of images of natural textures, having different mixes of texel shapes, number of fields, types of gradients, tilt directions, and three-dimensional texel effects. The results of the performance of the algorithm on a large variety of textures should help in judging the strengths, weaknesses, and generality of the algorithm and its current implementation. Part (a) of Figs. 3-30 show 14 of the images we have used in our experiments. A few of the images are photographs of outdoor scenes taken by one of the authors in Urbana, Illinois. The rest are illustrations in books (see [4] for references), which we have rephotographed. All of these images are digitized off of the photographic negatives using a drum scanner. The images are 512 by 512 pixels; the image sizes in the figures vary because image borders have been trimmed. All of the images are processed the same way; the method has no parameters that need to be tuned to particular images. As was described in Sections III and IV, the processing of an image is divided into three main phases: fit disks to the uniform im-

age regions, construct candidate texels from the disks, and fit a planar surface to the candidate texels.

The results of each phase are illustrated for one texture, a rock pile, in Figs. 3 and 4. Fig. 3 shows the results obtained for the positive contrast (bright) texture over dark background, and Fig. 4 shows the results for the negative contrast (dark) texture over bright background. The original image is shown in part (a) of each figure. Part (b) of each figure shows the extracted disks that model the regions of uniform gray level in the original image. Overlapping sets of these disks are used to make the list of candidate texels. It is impossible to display all the disks in a single image, since many disks are spatially contained in larger disks. This spatial containment typically means that either 1) the large disk is part of a texture element and the small disks are subtexture, or 2) the small disks are texels and the large disk is supertexture. In case 1) the large disk usually has higher contrast than the smaller disks, whereas in case 2) the smaller disks usually have higher contrast than the large disk. Wherever disks overlap, our figures shows the disk of higher contrast. Therefore most subtexture disks in part (b) are not visible: they are covered by a larger, higher-contrast disk corresponding to part of a texture element.

The parameters of the best planar fit are illustrated by the synthetic texture images in part (e) of the figures. The detected texels are shown in parts (c) and (d): these are all candidate texels having area within a factor of two of the area expected by the best planar fit.

Parts (f) and (g) illustrate the change of fit-rating as a function of A_c , slant, and tilt. The height fields in part (f) of each figure show fit-rating as a function of slant and tilt, with A_c fixed at the value that produces the best planar fit for the texture in question.¹¹ The height fields flatten out near the back because tilt becomes less important as slant decreases; tilt is irrelevant when the slant is zero. The graphs in part (g) of each figure show fit-rating as a function of A_c , with slant and tilt fixed at the values that produce the best planar fit for the texture in question.

Figs. 5-30 illustrate selected results for 13 additional images of natural textures. The results obtained for each image are illustrated in two successive figures. The first figure shows the results for the positive-contrast texture, and the second figure shows the results for the negative-contrast texture. Parts (a), (b), and (c) of Figs. 5-30 are analogous to the corresponding parts of Figs. 3 and 4, whereas part (d) of Figs. 5-30 is analogous to part (e) of Figs. 3 and 4. For brevity, the details shown in parts (d), (f), and (g) of Figs. 3 and 4 are not repeated for the textures in Figs. 5-30.

The shape of the fit-rating peak is related to the properties of the image texture. A sharp fit-rating peak indicates that the texels have small size variance. This observation is supported by the fit-rating plots for the aerial view of houses (Figs. 5 and 6) and by the field of sun-

¹¹In these height fields, the fit-rating values have been squared for display purposes.

flowers (Figs. 15 and 16), although, for brevity, these plots are not shown in the figures. If the texel sizes have larger variance, as for the clouds (Figs. 11 and 12) and the rock pile (Figs. 3 and 4), then the peak is much broader. (In the rock-pile image, the nonplanarity of the original textured surface also contributes to the broadness of the fit-rating peak.) The texels shown in part (c) of the figures are those candidate texels having area within a factor of two of the area expected by the planar fit. Using this same factor of two for all images causes incomplete extraction of texels in images where texel size is highly variable. More complete texel extraction can be achieved by adjusting the criteria for choosing texels from the set of candidate texels; the criteria should vary as a function of the broadness of the fit-rating peak in (A_c , S , T) space.

The accuracy of the results may be illustrated in two ways. First, the reader can compare his perception of the textured surfaces (part (a) of Figs. 3–30) with the planar surface fitted by the program. Agreement with human perception is quite good for many of the images. Second, since the processing of the positive-contrast and negative-contrast regions is performed totally independently, the agreement between the slants and tilts obtained by the two analyses strengthens the confidence in the results. (Note that the A_c parameters are not expected to be similar for the positive-contrast and negative-contrast regions—the positive-contrast and negative-contrast regions may be of very different sizes.) However, the two analyses may not always lead to the same estimates of slant and tilt, because a texture may not be homogeneous in both texel size and texel separation. Thus, an agreement among multiple analyses (such as the two discussed here) should not be required; instead, a method of automatically assessing the accuracies of the results obtained by different analyses, and selecting and integrating the pertinent analyses must be devised. Work is underway to address this problem.

Table I summarizes the planar fits obtained for all images. These fits use slants that are multiples of 2.5° and tilts that are multiples of 5° . The slant and tilt values computed from the positive-contrast and negative-contrast regions are frequently within 15° of each other. For reference, a 30° difference in tilt is equal to the angular distance between adjacent numbers on a clock face. A 30° difference in slant, on the other hand, is a more serious error. In many of those images that have a large discrepancy between the two planar fits, attributes of the original texture lead us to expect the fits to differ in accuracy. We have identified four reasons for the observed discrepancies. In the field of flowers (Fig. 23) and the water lilies (Fig. 25), the spaces between the texels are less regular than are the areas of the texels; therefore the fit to the negative-contrast regions is not as accurate as the fit to the positive-contrast regions. A second reason the background regions produce inaccurate results is because the properties of the physical texels are more important than the properties of background regions. In images where the physical texels are separated by gaps, the intertexel spacing carries more information than does the shape or area

TABLE I

Description	Figures	Fit to positive-contrast regions			Fit to negative-contrast regions			Difference	
		A_c	slant	tilt	A_c	slant	tilt	slant	tilt
A rock pile	3, 4	40	62.5°	65°	40	60°	75°	2.5°	10°
Aerial view of houses	5, 6	35	62.5°	55°	60	67.5°	110°	5°	15°
Birds flying over water	7, 8	35	45°	80°	40	57.5°	100°	12.5°	20°
Prayer at a mosque	9, 10	160	27.5°	50°	120	42.5°	100°	15°	50°
Fleecy clouds	11, 12	100	55°	275°	160	55°	280°	0°	5°
3D movie audience	13, 14	280	45°	105°	320	7.5°	230°	large	
Sunflowers	15, 16	160	70°	95°	200	70°	90°	0°	5°
A tree trunk	17, 18	70	65°	345°	80	42.5°	0°	25.5°	15°
Bathers on the Ganges	19, 20	100	45°	80°	80	65°	85°	20°	5°
A plowed field	21, 22	80	42.5°	40°	100	65°	80°	22.5°	40°
A field of flowers	23, 24	50	70°	90°	140	52.5°	20°	large	
Water lilies	25, 26	120	75°	90°	160	52.5°	70°	22.5°	20°
Ripples	27, 28	50	52.5°	105°	120	62.5°	105°	10°	0°
Water Hyacinths	29, 30	100	37.5°	80°	100	40°	80°	2.5°	0°

of the background regions. Thus, the results for the negative-contrast regions of the movie image (Fig. 14) and the lily pad image (Fig. 26) are inaccurate because the area of the background regions poorly reflects the intertexel spacing. A third reason for discrepancies between the two slant and tilt estimates is a large variability in texel area (as occurs in Fig. 9, the image of prayer at a mosque). This causes a broad peak in the planar fit space; hence the exact peak location is not as accurate for these images as for others. A fourth reason for inaccurate results is that the current extraction of uniform regions fragments noncompact regions in an arbitrary way, increasing the variabilities of the measured areas. This effect can be seen in the background of the movie image (Fig. 14). For nearly all of the images, at least one of the two analyses produces results that are in good agreement with human perception.

VI. SUMMARY

We have presented a general discussion of the problem of recovering scene-layout information from the texture cues present in an image. We argue that extraction of texels is useful and perhaps even necessary for correct interpretation of texture gradients in the face of subtexture and supertexture. In order to separate texture elements from other regions (such as subtexture) it is necessary to perform texel identification and surface fitting simultaneously.

We have presented an implementation that is based on these ideas; the implementation is restricted to the detection of gradients of texel area. A multiscale region detector is developed from the response of an ideal disk to convolution with a Laplacian-of-Gaussian ($\nabla^2 G$) over a range of scales. The output of the region detector is used to form a list of candidate texels. These candidate texels then provide the evidence needed to choose a good planar fit to the image texture; at the same time, the best planar fit determines which of the candidate texels are true texels. Results are shown for a variety of natural textures.

One consequence of the integration approach presented in this paper is that all regions whose properties are not unified by the gradient of a given property are treated as noise. For any given property, the noise regions do not contribute significantly to the fit-rating quality by virtue

of the exponential function in (10). Such regions could be the result of noise in the original image or in the region detection process. However, these noise regions could be valid texels if the gradient of a different property is considered, and they could quite possibly support the same surface orientation as the nonnoise regions. As long as there is some property whose gradient is supported by regions occupying a sufficiently large image area, the corresponding regions must be treated as texels. This is why the use of multiple texture gradients is necessary. A goal of our ongoing research is to estimate surface orientation from an integrated analysis of several relevant texture gradients, including area gradients, aspect-ratio gradients, and density gradients.

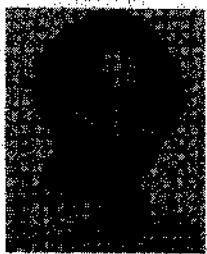
Because of the significant variability which is characteristic of natural textures, texture gradient as a cue of surface orientation appears to be more useful to obtain a coarse judgment of surface orientation and scene layout than as a source of obtaining accurate estimates. Stereo and other sources of scene information may be more appropriate for obtaining greater accuracy, e.g., for extracting shapes of curved, complex surfaces. In this sense, the analysis based on planar surfaces may suffice for most natural scenes containing textured surfaces, although mathematically (or for use with synthetic textures, where texture variability could be controlled), the approach presented in this paper could be extended to apply to curved surfaces. The extension required would be only in the surface fitting process. A much more important use of texture cues in real scenes is for segmentation of a scene into different textured surfaces [25]. With such segmentation available, it would be possible to identify image parts to which the approach of this paper could be applied meaningfully. As we stated earlier, we have not addressed the problem of texture segmentation in this paper.

REFERENCES

- [1] J. Aloimonos and M. Swain, "Shape from texture," in *Proc 9th Int. Joint Conf. AI*, 1985, pp. 926-931.
- [2] J. Aloimonos, "Detection of surface orientation from texture I: The case of planes," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1986, pp. 584-593.
- [3] R. Bajcsy and L. Lieberman, "Texture gradient as a depth cue," *Comput. Graphics Image Processing*, vol. 5, pp. 52-67, 1976.
- [4] D. Blostein, "Recovering the orientation of textured surfaces in natural scenes," Ph.D. dissertation, Univ. Illinois, Coordinated Science Lab. Rep. UILLU-ENG-87-2219, Apr. 1987.
- [5] D. Blostein and N. Ahuja, "Representation and three-dimensional interpretation of image texture: An integrated approach," in *Proc. IEEE First Int. Conf. Computer Vision*, June 1987, pp. 444-449.
- [6] —, "A multi-scale region detector," *Comput. Vision, Graphics, Image Processing*, vol. 45, no. 1, pp. 22-41, Jan. 1989.
- [7] M. L. Braunstein and J. W. Payne, "Perspective and form ratio as determinants of relative slant judgments," *J. Exp. Psychol.*, vol. 81, no. 3, pp. 584-590, 1969.
- [8] J. Crowley and A. Parker, "A representation for shape based on peaks and ridges in the difference of low pass transform," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 2, pp. 156-170, Mar. 1984.
- [9] J. E. Cutting and R. T. Millard, "Three gradients and the perception of flat and curved surfaces," *J. Exp. Psychol.: General*, vol. 113, no. 2, pp. 198-216, 1984.
- [10] L. Davis, L. Janos, and S. Dunn, "Efficient recovery of shape from texture," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 5, pp. 485-492, Sept. 1983.
- [11] J. Gibson, *The Perception of the Visual World*, Boston, MA: Houghton Mifflin, 1950.
- [12] K. Ikeuchi, "Shape from regular patterns (an example of constraint propagation in vision)," MIT A.I. Memo 567, Mar. 1980.
- [13] K. Kanatani, "Detection of surface orientation and motion from texture by a stereological technique," *Artificial Intell.*, vol. 23, pp. 213-237, 1984.
- [14] K. Kanatani and T. Chou, "Shape from texture: General principle," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition* 86, Miami, FL, June 1986, pp. 578-583.
- [15] J. Kender, "Shape from texture," Ph.D. dissertation, Carnegie-Mellon Univ., Rep. CMU-CS-81-102, Nov. 1980.
- [16] H. Nakatani, S. Kimura, O. Saito, and T. Kitahashi, "Extraction of vanishing point and its application to scene analysis based on image sequence," in *Proc. Int. Conf. Pattern Recognition*, 1980, pp. 370-372.
- [17] R. Nevatia and K. R. Babu, "Linear feature extraction and description," *Comput. Graphics Image Processing*, vol. 13, pp. 257-269, 1980.
- [18] Y. Ohta, K. Maenobu, and T. Sakai, "Obtaining surface orientation from texels under perspective projection," in *Proc. Int. Joint Conf. Artificial Intelligence*, 1981, pp. 746-751.
- [19] R. J. Phillips, "Stationary visual texture and the estimation of slant angle," *Quart. J. Psychol.*, vol. 22, pp. 389-397, 1970.
- [20] A. Rosenfeld, "A note on automatic detection of texture gradients," *IEEE Trans. Comput.*, vol. C-24, pp. 988-991, Oct. 1975.
- [21] R. R. Rosinski, "On the ambiguity of visual stimulation: A reply to Eriksson," *Perception Psychophys.*, vol. 16, no. 2, pp. 259-263, 1974.
- [22] R. Rosinski and N. Levine, "Texture gradient effectiveness in the perception of surface slant," *J. Exp. Child Psychol.*, vol. 22, pp. 261-271, 1976.
- [23] K. A. Stevens, "Slant-tilt: The visual encoding of surface orientation," *Biol. Cybern.*, vol. 46, pp. 183-195, 1983.
- [24] D. Vickers, "Perceptual economy and the impression of visual depth," *Perception and Psychophys.*, vol. 10, no. 1, pp. 23-27, 1971.
- [25] H. Voorhees and T. Poggio, "Detecting textons and texture boundaries in natural textures," in *Proc. IEEE First Int. Conf. Computer Vision*, June 1987, pp. 25-258.
- [26] A. P. Witkin, "Recovering surface shape and orientation from texture," *Artificial Intell.*, vol. 17, pp. 17-45, 1981.
- [27] —, "Scale space filtering," in *Proc. Eighth Int. Joint Conf. Artificial Intelligence*, Karlsruhe, West Germany, Aug. 1983, pp. 1019-1022.

Dorothea Blostein (S'87-M'88) received the B.S. degree in mathematics and computer science from the University of Illinois, Urbana-Champaign, in 1978, the M.S. degree in computer science from Carnegie-Mellon University, Pittsburgh, PA, in 1980, and the Ph.D. degree in computer science from the University of Illinois in 1987.

From 1980 to 1982, she worked at Intel Corporation. During 1987-1988, she worked at the University of Illinois Computer-Based Education Research Laboratory. She is currently an Assistant Professor in the Department of Computing and Information Science at Queen's University in Kingston, Ontario. Her research interests include computer vision, pattern recognition, computer music, and user-interface design.



Narendra Ahuja (S'79-M'79-SM'85) received the B.E. degree with honors in electronics engineering from the Birla Institute of Technology and Science, Pilani, India, in 1972, the M.E. degree with distinction in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1974, and the Ph.D. degree in computer science from the University of Maryland, College Park, in 1979.

From 1974 to 1975 he was Scientific Officer in the Department of Electronics, Government of India, New Delhi. From 1975 to 1979 he was at the Computer Vision Laboratory, University of Maryland, College Park. Since 1979 he has been with the University of Illinois at Urbana-Champaign where he is currently (1988-) a Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute. His interests are in computer vision, robotics, image processing, and par-

allel algorithms. He has been involved in teaching, research, consulting, and organizing conferences in these areas. His current research emphasizes integrated use of multiple image sources of scene information to construct three-dimensional descriptions of scenes, the use of the acquired three-dimensional information for object manipulation and navigation, and multiprocessor architectures for computer vision.

Dr. Ahuja received the University Scholar Award (1985), Presidential Young Investigator Award (1984), National Scholarship (1967-1972), and President's Merit Award (1966). He has coauthored the books *Pattern Models* (Wiley, 1983) with Bruce Schachter, and *Motion and Structure from Image Sequences* (Springer-Verlag, to appear) with Juyang Weng and Thomas Huang. He is Associate Editor of the journals *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, and *Computer Vision, Graphics, and Image Processing*. He is a member of the American Association for Artificial Intelligence, the Society of Photo-Optical Instrumentation Engineers, and the Association for Computing Machinery.

Chapter 5: Three-Dimensional Object Recognition

The real world that we see and touch is primarily composed of three-dimensional solid objects. When an object is viewed for the first time, people typically gather information about that object from many different viewpoints. The process of gathering detailed object information and storing that information is referred to as "model formation." Once a person is familiar with many objects, the objects are then identified from an arbitrary viewpoint without further investigation. People are also able to identify, locate, and qualitatively describe the orientation of objects in black-and-white photographs. This basic capability is significant to computer vision because it involves the spatial variation of a single parameter within a framed rectangular region that corresponds to a fixed, single view of the world. The ability to identify, locate, and describe objects motivates the following definition of the autonomous, single arbitrary-view, three-dimensional-object recognition problem:

- Given any collection of labeled solid objects, each object can be examined and labeled models can be created using information from this examination.
- Given digitized sensor data corresponding to one particular, but arbitrary, field of view of the real world and the list of distinguishable objects, the following issues must be addressed for each object: Does the object appear in the digitized sensor data? If so, how many times and in what locations in the sensor data? For each occurrence, what are the three-dimensional location and orientation with respect to a known coordinate system?

Recognition system components

Recognition implies awareness of something already known. In modeling real-world objects for recognition purposes, many different kinds of schemes have been used. These are briefly described in the next section. To determine how recognition will take place, a method for matching model data to sensor data must be considered. A straightforward blind-search approach would entail (1) transforming all possible combinations of all possible known object models in all possible distinguishable orientations and locations into a digitized sensor data format and (2) matching based on minimization of a matching-error criterion. Clearly, this approach is impractical. On the other hand, since object models contain more object information than the sensor data, we are prohibited from transforming sensor data into complete model data and matching in the model data format. However, this does not prevent us from matching with partial model data. As a result, working with an intermediate domain that is computable from both sensor and model data is advantageous. This domain is referred to as the "symbolic scene description domain." A matching procedure is carried out on the quantities in this domain, which are referred to as "features."

Interactions between the individual components of a recognition system are illustrated in Figure 5.1.¹⁴⁷ The image formation process (represented on the figure by *I*) creates intensity or range data based purely on physical principles. The description process (*D*) acts on the sensor data and extracts relevant application-independent features. This process is completely data-driven and includes only the knowledge of the image formation process. The modeling process (*M*) provides object models for real-world objects. Object reconstruction from sensor data is one method for building models automatically. The understanding, or recognition, process (*U*) involves an algorithm to perform matching between models and data descriptions. This process might include data- and model-driven subprocesses, where segmented sensor data regions seek explanations in terms of models and hypothesized models seek verification from the data. The rendering process (*R*) produces synthetic sensor data from object models. Rendering provides an important feedback link by allowing an autonomous system to check on its own understanding of the sensor data by comparing synthetic images to the sensed images.

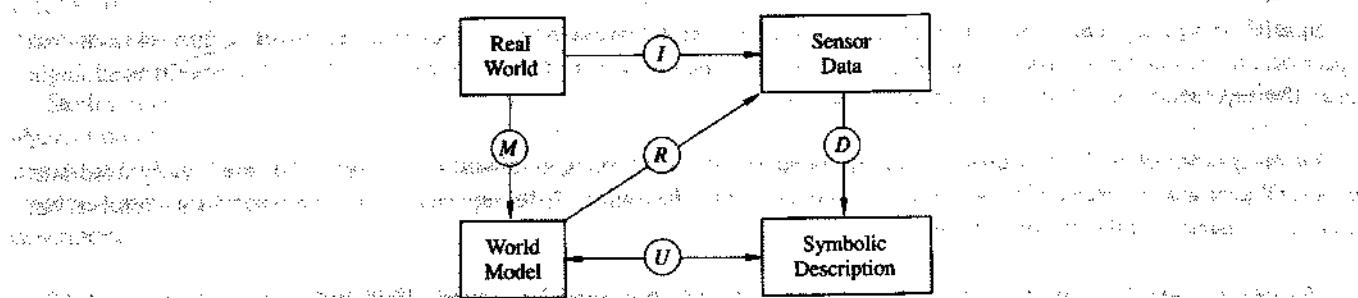


Figure 5.1: General object recognition system structure: *I*, image formation process; *M*, world-modeling process; *D*, description process; *U*, understanding process; *R*, model-rendering process (from P. Besl and R.C. Jain, "Three Dimensional Object Recognition," *ACM Computing Surveys*, Vol. 17, No. 1, Copyright 1985, Association for Computing Machinery, Inc., reprinted with permission).¹⁴⁷

Object representation schemes

Numerous schemes have been developed for representing three-dimensional objects. The choice of a particular scheme is governed by its intended application. In computer graphics, schemes such as wire-frame representation and constructive solid-geometry representation are popular since their data structures are suitable for image-rendering operations. In computer vision systems, other methods for representing three-dimensional objects, such as generalized cones and characteristic views, are used extensively. Using characteristic views for object recognition has been an important research topic in recent years and is discussed in a separate section. In this section we present an overview of other representation schemes.

Wire-frame representation. Wire-frame representation of a three-dimensional object consists of a three-dimensional vertex point list and an edge list of vertex pairs. Although this representation is very simple, it is an ambiguous representation for determining such quantities as surface area and volume of an object. Wire-frame models can sometimes be interpreted as several different solid objects or as different orientations of the same object.

Constructive solid-geometry representation. Constructive solid-geometry (CSG) representation of an object is specified in terms of a set of three-dimensional volumetric primitives (blocks, cylinders, cones, and spheres are typical examples of volumetric primitives) and a set of Boolean operators: union, intersection, and difference. Figure 5.2(a)¹⁴⁸ shows an example of computational solid-geometry representation. The corresponding solid object is shown in figure 5.2(b). The storage data-structure is a binary tree, where the terminal nodes are instances of primitives and the branching nodes represent Boolean set operations and positioning information. CSG trees define object surface area and volume unambiguously and can, with very little data, represent complex objects. However, the boundary evaluation algorithms required to obtain usable surface information are very computationally intensive. Also, general sculptured surfaces are not easily represented using CSG models.

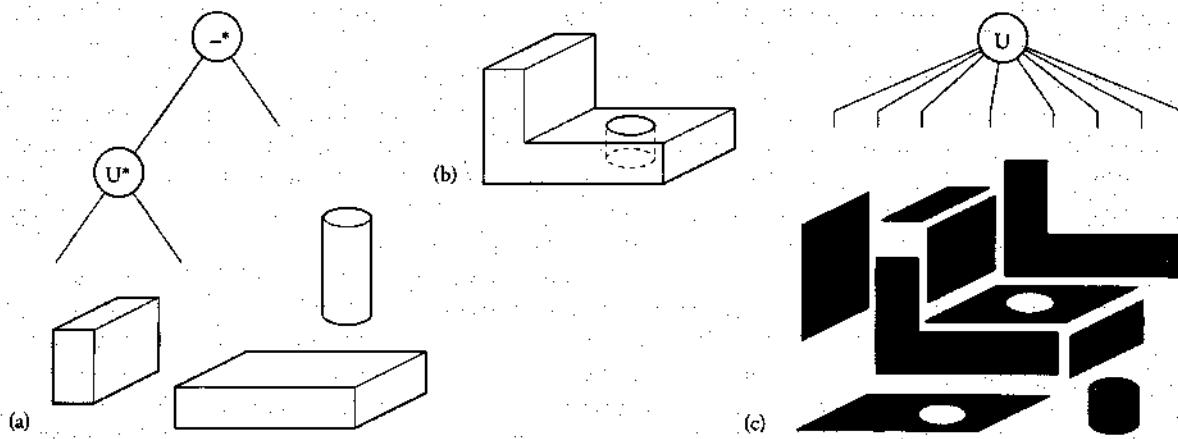


Figure 5.2: Computational solid geometry (CSG) and surface-boundary representations for a solid.
(a) CSG representation; (b) solid object; (c) surface-boundary representation (from Requicha and Voelcker).¹⁴⁸

Spatial-occupancy representation. Spatial-occupancy representation uses nonoverlapping subregions of the three-dimensional space occupied by an object to define that object. This method unambiguously defines an object's volume. Commonly used single primitive representations of this type are discussed below.

Voxel representation. Voxels are small-volume elements of discretized three-dimensional space. They are usually fixed-size cubes. Objects are represented by the list of voxels occupied by the object. Voxel representation is very memory intensive, but algorithms using it tend to be very simple.

Octree representation. An octree is a hierarchical representation of spatial occupancy.^{149,150} Volumes are decomposed into cubes of different sizes, where the cube size depends on the distance from the root node. Each branching node of the tree represents a cube and points to eight other nodes, each of which describes object volume occupancy in the corresponding octant subcubes.

of the branching node cube. Octree representation offers the advantages of the voxel description but is more compact. Because of this compactness, more complicated algorithms are required for many computations than those needed for voxel representation.

Other representations. Two other representations used are tetrahedral cell decomposition and hyperpatch representation. Tetrahedral decomposition of three-dimensional objects is similar to the lower dimensional analog of decomposing flat surfaces into triangles. Tetrahedral decomposition defines volume and surface area unambiguously and is useful for mathematical purposes. In hyperpatch representation, each volume element is a hyperpatch, or a generalization of bicubic surface patches.¹⁵¹ A hyperpatch defines volume, surface area, and internal density variations of a solid element. It is more general than most solid models — which allow only uniform density within a solid primitive — but a price is paid in memory and algorithm complexity. (For example, 192 scalars are required for each volume element.)

Surface boundary representation. Surface boundary representation defines a solid object by defining the three-dimensional surfaces that bound the object. Figure 5.2(c)¹⁴⁸ shows an example of surface boundary representation for the object shown in figure 5.2(b). The simplest boundary representation is the triangle-faced polyhedron, which can be stored as list of three-dimensional triangles. Arbitrary surfaces are approximated to any desired degree of accuracy by utilizing many triangles. A slightly more compact representation allows the replacement of adjacent, connected, coplanar triangles with arbitrary n-sided planar polygons. This type of representation is popular because model surface area and volume are well defined and all object operations are carried out using piecewise planar algorithms. Structural relationships between bounding surfaces may also be included as part of a model. Lu and Fu¹⁵² proposed a syntactic model that uses a context-free three-dimensional plex grammar for this purpose.

The object representation schemes relating to surface boundary representation discussed above have been used successfully to provide realistic renderings of real-world scenes. On the other hand, each real-world object shape requires, for matching, a unique description within the framework of a given representation. Most representations do not guarantee unique numerical descriptions of object shapes. Points, edges, faces, and/or primitives of a given representation can often be reordered or reorganized to obtain the same shape. If a modeling scheme suffers from a nonuniqueness problem, then model-based matching algorithms for computer vision systems must be made insensitive to this nonuniqueness. Badler and Bajcsy,¹⁵³ Bajcsy,¹⁵⁴ Brown,¹⁵⁵ Requicha,¹⁵⁶ and Requicha and Voelcker^{148,157} present more details on the relative merits of the above representation schemes. Three-dimensional representation schemes popular in computer vision literature are now briefly described.

Generalized-cone or sweep representation. In generalized-cone (or generalized-cylinder) representation, an object is represented by: a three-dimensional space curve that acts as a spine or axis of the cone, a two-dimensional cross-sectional figure, and a sweeping rule that defines how the cross section is to be swept and possibly modified along the space curve.^{158,159} Generalized cones are well suited for representing many real-world shapes; however, certain objects — such as a human face or an automobile body — are almost impossible to represent as generalized cones. Despite its limitations, generalized-cone representation is popular in computer vision.

Multiple two-dimensional projection representation. For some applications, a library of two-dimensional silhouette projections that represent three-dimensional objects can be conveniently stored. For recognition of three-dimensional objects with a small number of stable orientations on a flat light table, this representation is ideal, if object silhouettes are different enough. Silhouettes have also been used to recognize aircraft in any orientation against a well-lit sky background.¹⁶⁰ However, because many different three-dimensional-object shapes can possess the same set of silhouette projections, this type of representation is not a general-purpose technique. Recently, more powerful two-dimensional projection representation schemes, known as "aspect graphs" and "characteristic views," are becoming popular. Because of their current importance, these schemes are discussed in more detail in the section entitled "Aspect Graphs and Characteristic Views."

Skeleton representation. Skeleton representation uses space-curve skeleton models.^{161,162} A skeleton can be considered an abstraction of the generalized cone description that consists of only the spines. Nackman¹⁶³ generalized the symmetric-axis-transform concept for arbitrary three-dimensional objects that have surface skeletons. Skeleton geometry provides useful abstract information. If a radius function is specified at each point on the skeleton, this representation is capable of general-purpose object description.

Surface representation schemes. The object recognition problem requires a representation that can model arbitrary solid objects to any desired level of detail and that can provide abstract shape properties for matching purposes. Because both range and intensity images are strongly influenced by object surface properties, surfaces must be evaluated explicitly in at least one

module of a vision system — no matter what representations are used. Thus, object recognition is dependent on surface perception. To avoid unnecessary computation when explicit surface information is required, a surface boundary representation is the natural choice. For polyhedral objects, surface representation can be easily accomplished using planar polygons. In this section, we now discuss briefly methods for representing surfaces of smooth-curved objects.

A general surface in three dimensions is written as

$$S = \{(x,y,z) : F(x,y,z) = 0\}. \quad (5.1)$$

This equation is referred to as an "implicit" representation of a surface. If the gradient vector ∇F exists, is continuous, and is nonzero for every point (x,y,z) , then S is a smooth surface. The implicit surface representation is useful for low-order polynomials of the spatial variables. Planar surfaces are precisely represented with only four coefficients (which describe three degrees of freedom), as follows:

$$F_{\text{plane}}(x,y,z) = Ax + By + Cz + D \quad (5.2)$$

A, B , and C specify the direction of the single normal to the surface and D specifies the distance of the plane from the origin of the coordinate system if A, B , and C are properly normalized. Quadric surfaces require 10 coefficients (which describe nine degrees of freedom), as follows:

$$F_{\text{quadric}}(x,y,z) = Ax^2 + By^2 + Cz^2 + Gxy + Hyz + Izx + Ux + Vy + Wz + D \quad (5.3)$$

Only three coefficients are needed to describe the shape of a quadric surface of a given type, whereas six parameters are needed to locate and orient the surface in space. If a quadric surface is properly translated and rotated, at least six of the 10 coefficients will be zero. All quadric surfaces can then be classified as one of the following types, using three or four nonzero coefficients in that particular coordinate system:

- (1) Ellipsoid ($A > 0, B > 0, C > 0, D = -1$);
- (2) Elliptic paraboloid ($A > 0, B > 0, W = -1$);
- (3) Hyperbolic paraboloid ($A > 0, B < 0, W = -1$);
- (4) Hyperboloid of one sheet ($A > 0, B > 0, C < 0, D = -1$);
- (5) Hyperboloid of two sheets ($A > 0, B < 0, C < 0, D = -1$); and
- (6) Quadric cone ($A > 0, B > 0, C < 0$).

Suppressing undesired undulations in polynomial surface functions becomes more difficult as the order of the polynomial increases. Therefore, implicit surface representation unfortunately is not generally useful for arbitrary surface descriptions — unless surfaces are decomposed into a collection of locally homogeneous surface patches.

The standard alternative approach is to use "explicit" parametric surface representation, as follows:

$$S = \{(x,y,z) : x = h(u,v), y = g(u,v), z = f(u,v), (u,v) \in D \subseteq \mathbb{R}^2\}, \quad (5.4)$$

where f, g , and h are smooth scalar functions of two variables. A less general, but still useful, parametric description of surfaces is given via the graph surface (or Monge patch) representation, as follows:

$$S = \{(x,y,z) : x = u, y = v, z = f(u,v), (u,v) \in D \subseteq \mathbb{R}^2\}, \quad (5.5)$$

Gray-level surfaces in intensity images and depth surfaces in range images are typically analyzed using this common representation.

A wide variety of techniques has been developed for representing objects and surfaces for digital-computing purposes. To make informed decisions, designers of object recognition systems should be aware of these techniques.

Aspect graphs and characteristic views

Koenderink and Van Doorn^{164,165} introduced the concept of aspect graphs for representing shapes of three-dimensional objects. The space of viewpoints is partitioned into maximal regions, where every viewpoint in each region gives the same qualitative view of the object, called the “aspect” of the object. Within each region, projections of the object will have the same number and types of features, with identical spatial relationships among them. However, the quantitative properties of these features, such as lengths of edges, vary with change in viewpoint. Changes in the aspect, called “visual events,” take place at the boundaries between regions. Two aspects are said to be connected by a visual event if their corresponding regions are adjacent in the viewpoint space. An aspect graph is a graph structure whose nodes represent aspects and whose arcs denote visual events and boundaries between adjacent regions. Figure 5.3¹⁴⁷ shows an aspect graph for a cube; the nodes represent various aspects, and the arcs that connect these nodes represent visual events that transform one aspect to another.

Figure 5.3: Nodes illustrating aspects and arcs representing visual events for a cube (from P. Beal and R.C. Jain, "Three Dimensional Object Recognition," ACM Computing Surveys, Vol. 17, No. 1, 1985, pp. 75-145. Copyright 1985, Association for Computing Machinery, Inc., reprinted with permission.)¹⁴⁷

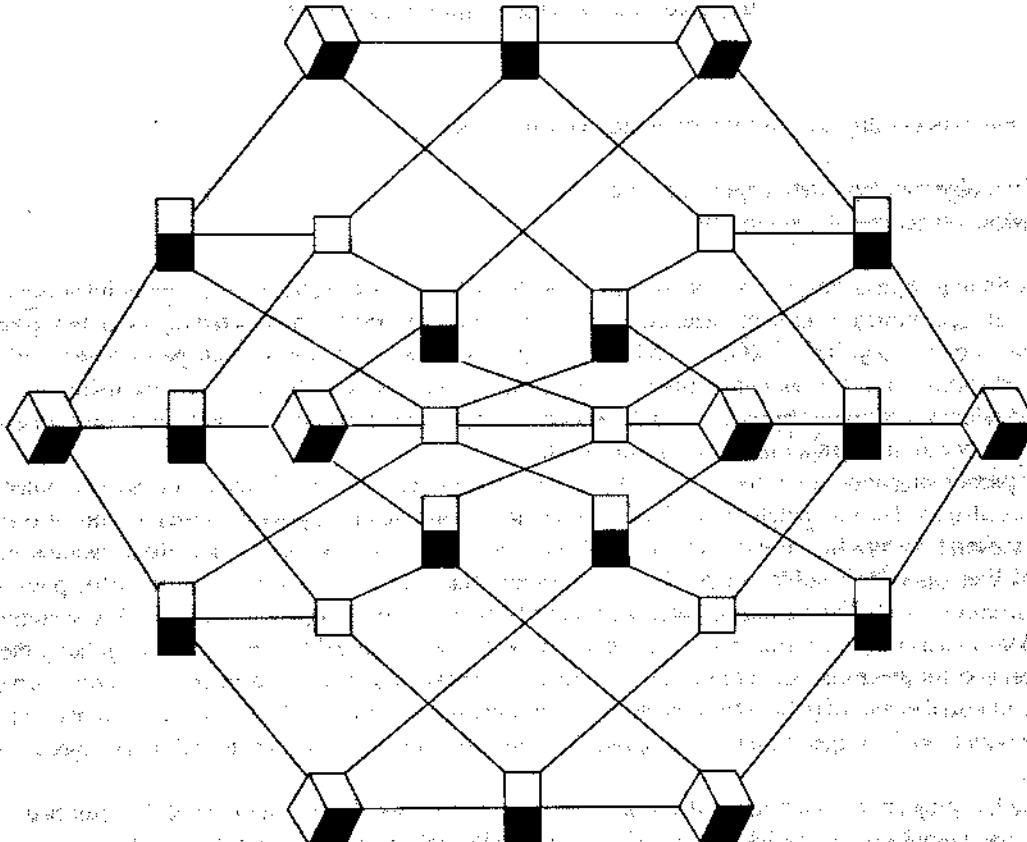


Figure 5.3: Nodes illustrating aspects and arcs representing visual events for a cube (from P. Beal and R.C. Jain, "Three Dimensional Object Recognition," ACM Computing Surveys, Vol. 17, No. 1, 1985, pp. 75-145. Copyright 1985, Association for Computing Machinery, Inc., reprinted with permission.)¹⁴⁷

Using a concept similar to that of the aspect graph, Chakravarty and Freeman¹⁶⁶ proposed representing an object by a set of characteristic views. In Chakravarty and Freeman’s system, all of the infinite two-dimensional perspective projection views of an object are grouped into a finite number of topologically equivalent classes. Different views within an equivalence class are related via linear transformations. A representative member of an equivalence class is called a “characteristic view.” In this system, objects rest on a supporting plane; hence, they are restricted to appear in a number of stable positions. Characteristic views of objects are derived with certain constraints on camera configuration. Figure 5.4¹⁴⁷ shows representative characteristic views for a polyhedron. Because characteristic views specify the three-dimensional structure of an object, they provide a general-purpose representation of that object.

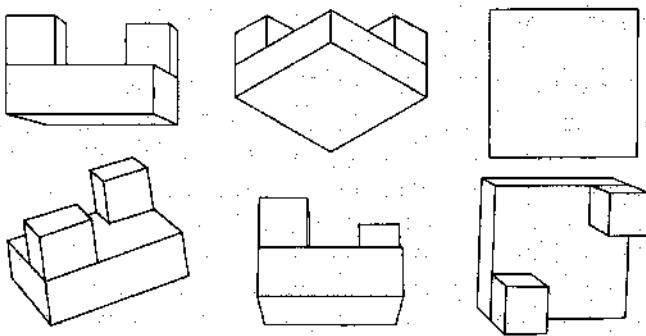


Figure 5.4: Representative characteristic views for a polyhedron (from P. Besl and R.C. Jain, "Three Dimensional Object Recognition," *ACM Computing Surveys*, Vol. 17, No. 1, 1985, pages 75-145. Copyright 1985, Association for Computing Machinery, Inc., reprinted with permission.)¹⁴⁷

The two general types of approaches for partitioning viewing space are

- (1) Uniform, object-independent approaches and
- (2) Analytical, object-specific approaches.

In uniform partitioning approaches, regions of constant aspects are found by tessellating the viewpoint space into uniformly distributed cells of approximately the same size; then, adjacent cells are grouped into the same region if they give qualitatively similar projections of the object.¹⁶⁷⁻¹⁷² (Gigus and Malik¹⁷³ presented an example of such partitioning with a tessellated dodecahedron projected onto the viewing sphere.) The grouping process can be achieved by using a region-growing method.¹⁷¹ The uniform partitioning approach is independent of object shape structures, but the exact aspect graph cannot be created because only a finite number of views is examined.

In analytical partitioning approaches for constructing exact aspect graphs, accidental viewpoints are computed directly from the object surface shapes. The computations are based on the understanding of (1) possible visual events of certain classes of objects and (2) the relationships between object surface shapes and accidental viewpoints where these visual events take place. Koenderink and Van Doorn^{164,165} published the first visual-event catalog for smooth objects; it describes possible changes in topological structures of occluding contours. Later, Arnold,¹⁷⁴ Gaffney,¹⁷⁵ and Kergosien¹⁷⁶ independently compiled all of the individual catalogs into a complete visual-event catalog for smooth objects using different tools in singularity theory. Rieger¹⁷⁷ extended this catalog for piecewise smooth objects bounded by nonplanar surfaces and their intersection curves. Gigus and Malik¹⁷² described visual events of polyhedral objects. Sripradisvarakul and Jain¹⁷⁸ studied new visual events for curved objects, which, together with those in Rieger's catalog, comprise the complete list of visual events for arbitrary objects with planar and curved surfaces.

For orthographic projection, Plantinga and Dyer,^{179,180} Gigus and Malik,¹⁷³ and Gigus et al.¹⁸¹ described algorithms for computing the aspect graphs of polyhedral objects. Kriegman and Ponce¹⁸² and Eggert and Bowyer¹⁸³ presented algorithms for creating aspect graphs of solids of revolution. Sripradisvarakul and Jain¹⁷⁸ described an algorithm for aspect graph construction for a general curved object. A similar algorithm for curved objects was published recently by Ponce and Kriegman.¹⁸⁴

For perspective projection, Castore,¹⁸⁵ Stewman and Bowyer,^{186,187} and Watts¹⁸⁸ gave algorithms for creating aspect graphs of convex polyhedral objects.

Object recognition systems

ACRONYM is a model-based system for three-dimensional interpretation of two-dimensional images.¹⁸⁹⁻¹⁹⁰ It is a complex, large-scale, domain-independent modular system that uses view-independent volumetric object models. Figure 5.5¹⁸⁹ shows for the ACRONYM system both the block diagram and the hierarchical geometric-reasoning process diagram. The system is based on the prediction-hypothesis-verification paradigm. The three main data structures of the system are the following:

- Object graph: Nodes of the object graph are generalized-cone object models. Arcs of the object graph correspond to the spatial relationships between the nodes (e.g., relative translations and rotations) and the subpart relations (e.g., is-a-part-of).

- **Restriction graph:** Nodes of the restriction graph are constraints on the object models of a given object class. The directed arcs of the restriction graph represent subclass inclusions.
- **Prediction graph:** Nodes of the prediction graph are “invariant” and “quasi-invariant” observable-image object features. Arcs of the prediction graph specify the image relationships between the invariant features. These arcs are of the following types: must-be, should-be, and exclusive.

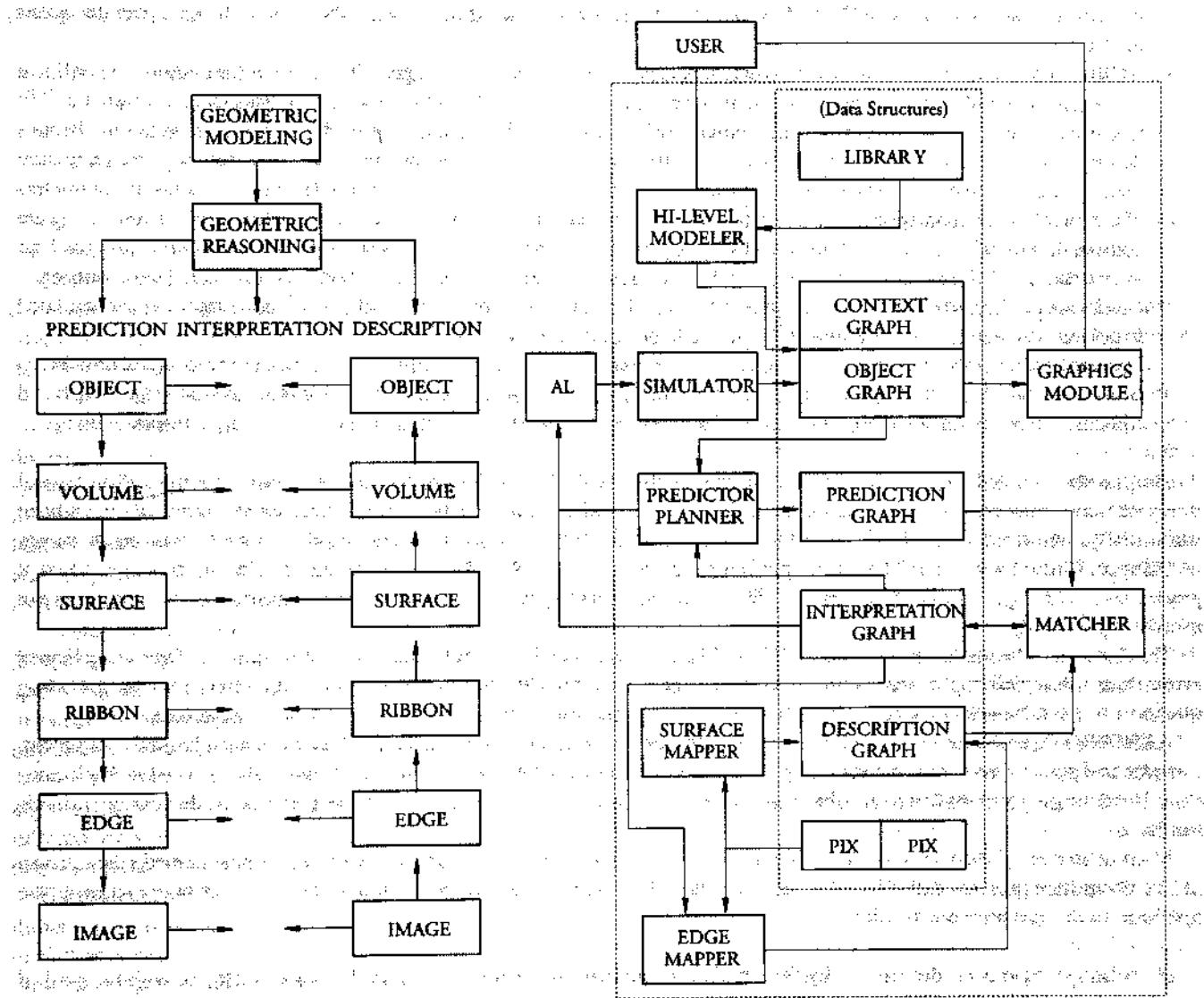


Figure 5.5: The block diagram and hierarchical geometric reasoning process diagram of the ACRONYM system (from Brooks et al., "The ACRONYM Model-Based Vision System," Proc. 6th Int'l Conf. on Artificial Intelligence, Copyright Int'l Joint Conf. on Artificial Intelligence, Inc.)

Each data object of the system is referred to as a “unit.” Every unit has associated slots to hold descriptive information. For example, a cylinder has a length slot and a radius slot. Slots accept numeric fillers or quantifier expressions. The ACRONYM system operates approximately as follows:

- An a priori world model is given to the system as a set of object sets and object classes. Simple objects are represented as generalized cones with specific dimensions. Each object and object class can be a hierarchy of subparts, each with its own local coordinate system. Object classes are represented as objects with constraints on subpart dimensions and configurations.
- An object graph, a restriction graph, and a prediction graph are formed based on the world model and a set of production rules.

- The system is given a digitized intensity image, a camera model, and the three graph data structures formed in the preceding step.
- The image is processed in two steps. First, an edge operator is applied to the image. Second, an edge linker is applied to the output of the edge operator and is directed to look for ribbons and ellipses. (Ribbons are the two-dimensional image projections of elongated bodies and ellipses are the ends of generalized-cone models.) The high level of three-dimensional geometric reasoning in ACRONYM is based entirely on the two-dimensional ribbon-and-ellipse symbolic scene description.
- ACRONYM searches for instances of object models in terms of ribbons and ellipses. The heart of the system is a nonlinear constraint manipulation system (CMS) that generalizes the linear SUP-INF methods of Presburger arithmetic.^{191,192} Constraint implications are propagated "downward" during prediction and "upward" during interpretation. Brooks describes the interpretation-matching process as follows: "Matching does not proceed by comparing image feature measurements with predictions for those measurements. Rather the measurements are used to put constraints on parameters of the three-dimensional models, of which the objects in the world are hypothesized to be instances. Only if constraints are consistent with what is already known of the model in three dimensions, then these local matches are retained for later interpretation."¹⁹⁰ Interpretation proceeds by the combination of local matches of ribbons into clusters. Two consistency checks are performed on the ribbon clusters: Each match must satisfy constraints of the prediction graph and accumulated matching constraints must be consistent with the hypothesized object model.
- The final output of the system is the labeled ribbons of the consistent image interpretation. Since orientation and translation constraints have been propagated during matching, three-dimensional positioning parameters are available for the labeled ribbons. Three-dimensional object identities, locations, and orientations are thus found using a single intensity image.

Despite the detailed three-dimensional concerns in the ACRONYM design, to our knowledge no three-dimensional interpretation results have ever been published. Aerial images of jets on runways and jets near airport terminals have been successfully interpreted using ACRONYM. Other less complicated schemes exist that could yield similar results on aerial images of this type. Binford wrote that "there is no profound reason why ACRONYM could not recognize aircraft in images taken at ground level, although it will probably break when tested on such images because of bugs or missing capabilities that were not exercised previously."¹⁹³

The theory and the implementation of ACRONYM are two separate issues, but readers are left wondering if this complicated system is as robust as it might seem. Many system difficulties have been blamed on the quality of output from the ribbon-finding mechanism. No feedback connections exist between the final decision-making mechanism and the original data.

ACRONYM's problems provide a reminder for us that any open-loop system is only as robust as its most limited component. Even the best possible geometric-reasoning system cannot be successful if its input is consistently unreliable and no feedback paths exist. Rendering algorithms that relate object models to sensor data at intermediate levels of interpretation could possibly provide feedback.

Many other three-dimensional object recognition schemes based on intensity images exist. Fisher¹⁹⁴ implemented a data-driven object recognition program called IMAGINE, in which surfaces are used as geometric primitives. The three major stages in the operation of this program are as follows:

- (1) Image regions — determined by their region boundaries — are matched to model object surfaces, with the goal of estimating surface orientation parameters. Specific object surfaces are hypothesized.
- (2) Hypothesized object surfaces are related to object models constrained by the structural relationships implied by the objects. Specific objects are hypothesized.
- (3) Hypothesized objects are verified using consistency checks against constraints due to adjacency and ordering.

The four specific goals of the IMAGINE program are

- To locate instances of three-dimensional objects in two-dimensional images;
- To locate image features corresponding to all features of the model or to explain why the image features are not present;
- To verify that all features are consistent with the geometric and topological predictions of the model; and
- To extract translation and rotation parameters associated with all objects in the scene.

The input to the IMAGINE program is presegmented surface regions that have the property that all boundaries between regions correspond to surface or shape discontinuities. The only information used by IMAGINE is the two-dimensional boundary shape of the segmented surface regions. The object models of the program are surface boundary models wherein all surfaces are planar

or have only a single axis of curvature. Subcomponent hierarchies for objects determine the joint connections of subparts. Model surface-to-image region matching is performed using a set of heuristics that generate hypotheses about rotation, slant and tilt, distance, and x-y translation in a plane. These heuristics gave reasonable results in 94 of 100 test cases¹⁹⁴. Given the hypothesized surfaces and their positions and orientations in space, a set of 10 rules is applied to generate object model hypotheses. Another set of occlusion-handling rules is applied for object verification. Fisher¹⁹⁴ provided his own list of program criticisms, which include the following:

- The heuristic parameter estimation techniques require mostly planar surfaces.
- The program's surface modeling does not account for surface shape internal to the region boundary.
- Surface segmentation is currently done by hand, with the assumption that adequate techniques will soon be available.
- The program's object models are nongeneric.

Despite these criticisms, the program did achieve its goals of recognizing and locating, in a test image, a PUMA robot and "understanding" its three-dimensional structure. Valuable ideas concerning occlusion are presented in Fisher's¹⁹⁴ paper.

Chakravarty and Freeman¹⁹⁵ developed a technique that uses characteristic views — described earlier — as a basis for three-dimensional object recognition in intensity images. Matching is performed using line junction labeling constraints on detected edges, which requires silhouette determination to guide the matching process. (This requirement is a disadvantage for occlusion handling.) In addition to identifying objects, this technique produces — as output — position and orientation information.

Some three-dimensional object recognition techniques are based purely on object silhouettes; as a result, these techniques cannot distinguish between objects having the same set of silhouettes. McKee and Aggarwal¹⁹⁶ worked on recognizing three-dimensional curved objects from a partial-silhouette description. However, three-dimensional object models are not used in this system. During the training process, the system learns the global silhouette boundary description for each view of an object and then stores the description in an object view library. The recognition algorithm accepts a partial-boundary description and then lists all of the compatible objects in the library. The work of McKee and Aggarwal¹⁹⁶ did not include view-independent processing and had problems with noisy edges. Dudani et al.¹⁹⁶ and Reeves et al.¹⁹⁷ used global moment-based silhouette shape description techniques for aircraft shape description. Sadjadi and Hall¹⁹⁸ discussed three-dimensional moment invariants.

Casasent et al.¹⁹⁹ developed a pattern recognition approach to object recognition based on synthetic-discriminant functions (SDFs), maximum-common-information filters, and decorrelation transformations. An SDF is a linear combination of matched spatial filters. It processes an entire input image without segmentation or preprocessing. Image correlations are performed instantaneously using optical means. SDFs are synthesized from training data chosen to represent various views of different objects. A type of nongeometric-model formation occurs during this training phase. Casasent et al.¹⁹⁹ discuss experimental results for two different objects. Thirty-six images of each object were obtained (10-degree rotation increments); for each of the two objects, six of these images were used for training. With two SDFs created from 12 images, 60 additional images taken from different views were correctly classified. A two-class mutual-orthogonal-function filter recognized the object and gave the correct orientation for 90 percent of the 72 images used.

Silberberg et al.²⁰⁰ used a generalized Hough transform to match detected two-dimensional line segments with three-dimensional-model line segments and observed two-dimensional edge junctions to three-dimensional model vertices. Silberberg et al.²⁰⁰ assumed that (1) all objects are polyhedra with single, stable positions, (2) the ground plane is known, and (3) camera position and parameters are known. They discussed experimental results for a synthetic image of a nonconvex polyhedron.

Goad²⁰¹ presented a technique for object recognition based on special-purpose automatic programming. In his technique, individual object descriptions are compiled into programs in which the only task is recognizing one object from any view. Time-consuming shape analysis is performed off-line prior to the recognition phase, so that actual recognition execution time is minimal (about one second). Goad²⁰¹ used a multiple-view object feature model that incorporates 218 different three-dimensional views of each object. Features are line segments stored as a pair of endpoints and a 218-bit bit-string, which describes the visibility of the feature in each of the 218 discrete views. Edges for objects are ordered by their expected utility for matching purposes. Goad²⁰² showed experimental results for a jumbled pile of key-caps for keyboards.

Shneier²⁰² proposed a combined multiple-object representation — a "graph of models" — where each graph node represents a three-dimensional surface primitive. The nodes contain a set of properties describing surface shape and a set of pointers to the object names to which the surface belongs. The arcs between the nodes describe relationships between surfaces and also contain pointers to the model names where those relationships occur. The integration of multiple objects into a single, shared-data structure provides a compact representation that is indexed quickly for fast recognition processing.

We conclude this section on a historical note. The pioneering work of Roberts,²⁰³ published in 1965, involved an intensity-image-based three-dimensional object recognition system. In this system, objects were constrained to be blocks, wedges, prisms, or combinations thereof. A cross operator was used to detect edges, and collinear segments were merged into lines to produce a

line drawing of the scene. Regions were classified as triangles, quadrilaterals, and hexagons; these regions were matched to faces of prototype objects. Possible object part model matches were rendered using a hidden-line algorithm to verify correct object matches. Recognized object parts were cut away from the image, and the same process was repeated until all detected edges and vertices were explained. After identifying an object, the system could draw the object from any view to demonstrate its understanding of the object shape. This research by Roberts was followed by the more advanced work of Guzman,²⁰⁴ published in 1968, Waltz,²⁰⁵ published in 1972, and others, which concentrated on line edge and edge junction labeling for detecting polygonal regions. These early systems addressed many of the fundamental problems encountered in computer vision, but were limited to processing high-quality images of block-world scenes. The algorithms were not robust enough to handle problems of real-world scenes, such as noise and curved objects.

Research trends

Automatic generation of three-dimensional-object representations from images has been the goal of many computer vision systems. Eleven papers were selected that are representative of papers dealing with the topics covered in this chapter. Six are included in this book and the other five in the companion book *Computer Vision: Advances and Applications*. In *Principles*, the paper entitled "Volumetric Descriptions of Objects From Multiple Views," by Martin and Aggarwal describe a system in which occluding contours from multiple images acquired from different — but known — viewpoints are used to construct a bounding volume. This volume approximates the three-dimensional structure of the object.

Many object recognition systems are built upon low-level vision modules that operate upon images to derive depth measurements. These measurements are often incomplete and unreliable, thereby adversely affecting the performance of higher level recognition modules. In contrast Lowe²⁰⁶ describes in detail a system in which bottom-up image description is designed to generate viewpoint-invariant groupings of image features. These features are used to reduce the search space for model matching. The viewpoint consistency constraint is applied, and object level data are mapped directly onto the image to determine model parameters. The ACRONYM system,¹⁸⁹ as already described in an earlier section, is a domain-independent model-based interpretation system that uses generalized cylinders to describe model and scene objects. ACRONYM's interpretation of aerial images is described by Brooks in "Model-Based Three-Dimensional Interpretations of Two-Dimensional Images" found in *Principles*.

Most object recognition research has considered a small set of objects. If a very large number of objects are to be recognized, the recognition task will be dominated by hypothesize-and-test approaches. The hypothesis formation phase requires organization of models indexed by features so that, based on observed features, a small set of likely objects can be selected. Later, these selected models can be used in object recognition to verify which object from this set is present in the given image. Such hypothesize-and-test approaches are already being designed.^{100,96,207,208}

In many industrial applications, detailed geometric models of objects are available. These models can be used for generating recognition strategies, including feature selection, for three-dimensional objects. CAD-based object recognition is now being studied at several places.^{101,167,172,209,210} An important step in recognition of three-dimensional objects is to consider their possible two-dimensional projections to determine effective features and recognition strategy. Classification of infinite two-dimensional projection views of objects into topologically equivalent classes and the application thereof to object recognition is described in *Principles* by Chakravarty and Freeman in "Characteristic Views as a Basis for Three-Dimensional Object Recognition." In *Advances*, "Computing the Aspect Graph for Line Drawings of Polyhedral Objects," authors Gigus and Malik describe an algorithm that partitions the set of viewpoints on the Gaussian sphere around an object into regions such that the qualitative structure of the line drawing remains unchanged. Their algorithm, which assumes orthographic projection, also computes a representative view of the object for each region. Recently, algorithms have been designed for computing aspect graphs for curved objects also.^{178,182,183} In *Principles*, "Fleshing Out Projections," by Wesley and Markowsky describes an algorithm that finds all solid polyhedral objects with a given set of two-dimensional orthographic projections. This problem reduces to that of a wire-frame interpretation problem if the projections are labeled. This algorithm has practical applications in the automatic conversion of engineering drawings into their corresponding volumetric representations.

In "Automatic Generation of Object Recognition Programs," Ikeuchi and Kanade in *Advances* describe a novel system in which object and sensor models are automatically compiled into a visual-recognition strategy. The system extracts from the models those features that are useful for recognition and determines the control sequence that must be applied to handle different object appearances. An alternative to this kind of approach is the neural-network approach to object recognition. Object recognition is one of the most researched areas in neural networks; however, most neural-network research has addressed only two-dimensional objects limited in the number of objects, simple shapes, binary images, and success.

In "Recognizing Three-Dimensional Objects Using Surface Descriptions," Fan et al. in *Principles* describe the system they designed that, using range image data, generates a symbolic object description in terms of visible surface patches. This segmented

image, which is treated as a graph describing the patches and their interrelationships, is decomposed into subgraphs corresponding to different objects. In *Advances*, "CAD-Based Computer Vision," by Flynn and Jain presents a method for deriving features for recognition of objects from CAD models and then use them in object recognition. Also in *Advances*, "The Evolution and Testing of a Model-Based Object Recognition System," by Mundy and Heller, discusses several important aspects of an object recognition system, from an approach to selecting features using models to a performance evaluation of the system. The last paper in *Advances*, "Geometric Reasoning for Recognition of Three-Dimensional Object Features" by Marefat and Kashyap, describes a method of extracting shape features from boundary representation of polyhedral objects. The method is based on "cavity graphs," which provide a topological and geometric description of the depressions in the object boundary. Finally, in *Principles*, we have included a survey paper, "Model-Based Recognition in Robot Vision," by Chin and Dyer. Other extensive surveys on object recognition are found in a paper by Besl and Jain⁴⁷ and in the paper entitled "Survey of Model-Based Image Analysis Systems," by Binford, which is reprinted in Chapter 7 of this book.

References Cited Chapter 5

147. P. Besl and R.C. Jain, "Three Dimensional Object Recognition," *ACM Computing Surveys*, Vol. 17, No. 1, 1985, pp. 75-145.
148. A.A. Requicha and H.B. Voelcker, "Solid Modeling: Current Status and Research Directions," *IEEE Computer Graphics and Applications*, Vol. 3, No. 7, 1983, pp. 25-37.
149. D.J. Meagher, "Geometric Modeling Using Octree Encoding," *Computer Graphics and Image Processing*, Vol. 19, 1981, pp. 129-147.
150. D.J. Meagher, "Efficient Synthetic Image Generation of Arbitrary 3-D Objects," *Proc. Pattern Recognition and Image Processing*, IEEE CS Press, Los Alamitos, Calif., 1982, pp. 473-478.
151. M.S. Casale and E.L. Stanton, "An Overview of Analytic Solid Modeling," *IEEE Computer Graphics and Applications*, Vol. 5, No. 2, 1985, pp. 45-56.
152. H.R. Lu and K.S. Fu, "A General Approach to Inference of Context-Free Programmed Grammars," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 14, 1984, pp. 191-202.
153. N. Badler and R. Bajcsy, "Three Dimensional Representation for Computer Graphics and Computer Vision," *ACM Computer Graphics*, Vol. 12, 1978, pp. 153-160.
154. R. Bajcsy, *Proc. Workshop on Representation of Three-Dimensional Objects*, University of Pennsylvania, Philadelphia, 1979.
155. C.M. Brown, "Some Mathematical and Representational Aspects of Solid Modeling," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 3, No. 4, 1981, pp. 444-453.
156. A.A. Requicha, "Representations for Rigid Solids: Theory, Methods, and Systems," *ACM Computing Surveys*, Vol. 12, 1980, pp. 437-464.
157. A.A. Requicha and H.B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, 1982, pp. 9-24.
158. S.A. Shafer and T. Kanade, "The Theory of Straight Homogeneous Generalized Cylinders and a Taxonomy of Generalized Cylinders," *Tech. Report CMU-CS-83-105*, Carnegie-Mellon University, Dept. of Computer Science, 1983.
159. B.I. Soroka and R.K. Bajcsy, "A Program for Describing Complex Three-Dimensional Objects Using Generalized Cylinders as Primitives," *Proc. Pattern Recognition and Image Processing*, IEEE CS Press, Los Alamitos, Calif., 1978, pp. 331-339.
160. T.P. Wallace and P.A. Wintz, "An Efficient Three-Dimensional Aircraft Recognition Algorithm Using Normalized Fourier Descriptors," *Computer Graphics and Image Processing*, Vol. 13, 1980, pp. 96-126.
161. G. Garibotto and R. Tosini, "Description and Classification of 3-D Objects," *Proc. Sixth Int'l Conf. Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1982, pp. 833-835.
162. K.J. Udupa and I.S.N. Murthy, "New Concepts for Three-Dimensional Shape Analysis," *IEEE Trans. Computer*, Vol. 26, 1977, pp. 1043-1049.
163. L.R. Nackman, "Three-Dimensional Shape Description Using the Symmetric Axis Transform," PhD thesis, Computer Science Dept., University of North Carolina, Chapel Hill, 1982.
164. J.J. Koenderink and A.J. van Doorn, "The Singularities of the Visual Mapping," *Biological Cybernetics*, Vol. 24, 1976, pp. 51-59.
165. J.J. Koenderink and A.J. van Doorn, "The Internal Representation of Solid Shape with Respect to Vision," *Biological Cybernetics*, Vol. 32, 1979, pp. 211-216.
166. I. Chakravarty and H. Freeman, "Characteristic Views as a Basis for Three-Dimensional Object Recognition," *Proc. SPIE Conf. Robot Vision*, Vol. 336, 1982, pp. 37-45.
167. C. Hansen and T. Henderson, "CAGD-Based Computer Vision," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 11, No. 11, 1989, pp. 1181-1193.
168. K. Ikeuchi, "Generating an Interpretation Tree from a CAD Model for 3-D Object Recognition in Bin-Picking Tasks," *Int'l J. Computer Vision*, 1987, pp. 145-165.
169. K. Ikeuchi and T. Kanade, "Automatic Generation of Object Recognition Programs," *Proc. IEEE*, Vol. 76, No. 8, IEEE Press, New York, N.Y., 1988, pp. 1016-1035.
170. A.C. Kak et al., "Knowledge-Based Robotics," *Proc. IEEE Int'l Conf. Robotics and Automation*, IEEE CS Press, Los Alamitos, Calif., 1987, pp. 637-646.
171. M.R. Korn and C.R. Dyer, "3-D Multiview Object Representations for Model-Based Object Recognition," *Pattern Recognition*, Vol. 20, No. 1, 1987, pp. 91-103.
172. L.G. Shapiro, "A CAD-Model-Based System for Object Localization," *Proc. SPIE Digital and Optical Shape Representation and Pattern Recognition*, Vol. 938, 1988, pp. 408-418.
173. Z. Gigus and J. Malik, "Computing the Aspect Graph for Line Drawings of Polyhedral Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 12, No. 2, 1990, pp. 113-122.
174. V.I. Arnold, "Singularities of Systems of Rays," *Russian Mathematical Surveys*, Vol. 38, No. 2, 1983, pp. 87-176.
175. T. Gaffney, "The Structure of TA(f), Classification and an Application to Differential Geometry," *Proc. Symposia in Pure Mathematics*, Vol. 40, American Math. Society, 1983, pp. 409-427.
176. Y.L. Kergosien, "La Famille des Projections Orthogonales d'une Surface et ses Singularités," *C.R. Acad. Sc. Paris*, Vol. 292, 1981, pp. 929-932.
177. J.H. Rieger, "On the Classification of Views of Piecewise Smooth Objects," *Image and Vision Computing*, Vol. 5, No. 2, 1987, pp. 91-97.

178. T. Sripradisvarakul and R.C. Jain, "Generating Aspect Graphs for Curved Objects," *Proc. IEEE Workshop on Interpretation of 3-D Scenes*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 109-115.
179. H.W. Plantinga and C.R. Dyer, "An Algorithm for Constructing the Aspect Graph," *Proc. 27th Symp. on Foundation of Computer Science*, IEEE CS Press, Los Alamitos, Calif., 1986, pp. 123-131.
180. H.W. Plantinga and C.R. Dyer, "The Asp: A Continuous Viewer-Centered Representation for 3-D Object Recognition," *Proc. First Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1987, pp. 626-630.
181. Z. Gigus, J.F. Canny, and R. Seidel, "Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects," *Proc. Second Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1988, pp. 30-39.
182. D.J. Kriegman and J. Ponce, "Computing Exact Aspect Graphs of Curved Objects: Solids of Revolution," *Proc. IEEE Workshop on Interpretation of 3-D Scenes*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 109-115.
183. D. Eggert and K. Bowyer, "Computing the Orthographic Projection Aspect Graph of Solids of Revolution," *Proc. IEEE Workshop on Interpretation of 3-D Scenes*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 102-108.
184. J. Ponce and D.J. Kriegman, "Computing Exact Aspect Graphs of Curved Objects: Parametric Surfaces," *Tech. Report UIUCDCS-R-90-1579*, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1990.
185. G. Castore, "Solid Modeling, Aspect Graphs, and Robot Vision," in *Solid Modeling by Computers: From Theory to Applications*, M.S. Pickett and J.W. Boyce, eds., Plenum, New York, N.Y., 1984.
186. J. Stewman and K. Bowyer, "Aspect Graphs for Planar-Face Convex Objects," *Proc. IEEE Workshop on Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1987, pp. 123-130.
187. J. Stewman and K. Bowyer, "Creating the Perspective Projection Aspect Graph of Polyhedral Objects," *Proc. Second Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1988, pp. 494-500.
188. N.A. Watts, "Calculating the Principal Views of a Polyhedron," *Proc. Ninth Int'l Conf. Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1988, pp. 316-322.
189. R.A. Brooks, R. Greiner, and T.O. Binford, "The ACRONYM Model-Based Vision System," *Proc. Sixth Int'l Joint Conf. on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., San Mateo, Calif., 1979, pp. 105-113.
190. R.A. Brooks, "Symbolic Reasoning among 3-D Models and 2-D Images," *Artificial Intelligence*, Vol. 17, 1981, pp. 285-348.
191. W.W. Bledsoe, "The Sup-Inf Method in Presburger Arithmetic," *Tech. Report ATP-18*, Dept. of Mathematics and Computer Science, University of Texas, Austin, 1974.
192. R.E. Shostak, "On the Sup-Inf Method for Proving Presburger Formula," *J. ACM*, Vol. 24, 1977, pp. 529-543.
193. T.O. Binford, "Survey of Model-Based Image Analysis," *Int'l J. Robotics Research*, Vol. 1, No. 1, 1982, pp. 18-64.
194. R.B. Fischer, "Using Surfaces and Object Models to Recognize Partially Obscured Objects," *Proc. Int'l Joint Conf. Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., San Mateo, Calif., 1983, pp. 989-995.
195. J.W. McKee and J.K. Aggarwal, "Computer Recognition of Partial Views of Three-Dimensional Curved Objects," *Tech. Report 171*, Dept. of Computer Science, University of Texas, Austin, 1975.
196. S. Dudani, K.J. Breeding, and R.B. McGhee, "Aircraft Identification by Moment Invariants," *IEEE Trans. Computer*, Vol. 26, 1977, pp. 39-46.
197. A.P. Reeves, R.J. Prokop, and F.P. Kuhl, "Three-Dimensional Shape Analysis Using Fourier Descriptors," *Proc. Seventh Int'l Conf. Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1984, pp. 447-450.
198. F.A. Sadjadi and E.L. Hall, "Three-Dimensional Moment Invariants," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 2, No. 2, 1980, pp. 127-136.
199. D. Casasent, B.V.K. Vijaya-Kumar, and V. Sharma, "Synthetic Discriminant Functions for Three-Dimensional Object Recognition," *SPIE Proc. of Conf. on Robotics and Industrial Inspection*, Vol. 360, 1982, pp. 136-142.
200. T.M. Silberberg, D.A. Harwood, and L.S. Davis, "Object Recognition Using Oriented Model Points," *Computer Vision, Graphics and Image Processing*, Vol. 35, 1986, pp. 47-71.
201. C. Goad, "Special Purpose Automatic Programming for 3-D Model-Based Vision," *Proc. DARPA Image Understanding Workshop*, 1983, pp. 94-104.
202. M.O. Shneier, "Models and Strategies for Matching in Industrial Vision," *Computer Science Tech. Report TR-1073*, University of Maryland, College Park, Md., July, 1981.
203. L.G. Roberts, "Machine Perception of Three-Dimensional Solids," in *Symposium On Optical and Electro-Optical Information Processing Technology*, J.T. Tippett et al., eds., MIT Press, Cambridge, Mass., 1965.
204. A. Guzman, *Computer Recognition of Three-Dimensional Objects in a Visual Scene*, PhD thesis, MIT, Cambridge, Mass., 1968.
205. D.L. Waltz, "Generating Semantic Descriptions from Drawing of Scenes with Shadows," *Tech. Report AI-TR-271*, Artificial Intelligence Laboratory, MIT, Cambridge, Mass., 1972.
206. D.G. Lowe, "Three-Dimensional Object Recognition from Single Two-Dimensional Images," *Artificial Intelligence*, Vol. 31, No. 3, 1987, pp. 355-395.
207. W.E.L. Grimson, "Recognition of Object Families Using Parameterized Models," *Proc. First Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1987, pp. 93-101.
208. Y. Lamdan and H.J. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Proc. Second Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1988, pp. 238-249.
209. T.O. Binford, "Spatial Understanding: The SUCCESSOR System," *Proc. DARPA Image Understanding Workshop*, 1989, pp. 12-20.
210. N. Narasimhamurthy and R.C. Jain, "Computer-Aided, Design-Based Object Recognition: Incorporating Metric and Topological Information," *Proc. SPIE Conf. Digital and Optical Shape Representation and Pattern Recognition*, Vol. 938, 1988, pp. 436-433.

Reprinted from *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume PAMI-5, Number 2, March 1983, pages 150-158. Copyright © 1983 by The Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

Volumetric Descriptions of Objects from Multiple Views

WORTHY N. MARTIN, MEMBER, IEEE, AND J. K. AGGARWAL, FELLOW, IEEE

Abstract—Occluding contours from an image sequence with viewpoint specifications determine a bounding volume approximating the object generating the contours. The initial creation and continual refinement of the approximation requires a volumetric representation that facilitates modification yet is descriptive of surface detail. The "volume segment" representation presented in this paper is one such representation.

Index Terms—Dynamic scene analysis, occluding boundary, surface description, volume description.

I. INTRODUCTION

VOLUMETRIC models have been the basis of numerous three-dimensional object modeling systems. In order to represent the desired volume, various types of primitives have been specified. Initially, polyhedrons were used as combinations of a small set of simple volumes [1] and later as general

Manuscript received May 17, 1982; revised August 17, 1982. This work was supported in part by the U.S. Air Force Office of Scientific Research under Grant AFOSR 82-0064.

W. N. Martin was with the Laboratory for Image and Signal Analysis, University of Texas, Austin, TX 78712. He is now with the Department of Applied Mathematics and Computer Sciences, University of Virginia, Charlottesville, VA 22901.

J. K. Aggarwal is with the Laboratory for Image and Signal Analysis, University of Texas, Austin, TX 78712.

volumes [2]. Each face of a polyhedron may be considered to be part of the bounding plane of a half-space. The interior of the polyhedron can then be described procedurally by the "addition" and "subtraction" of these half-spaces. Requicha [3] defines "general regularized set operators" and a syntax for specifying the procedural description in a constructive solid geometry scheme. The operators will always yield "valid" representations if the primitives have finite volumes. Through additional operators general sweep representations can also be specified. Generalized cylinders are a class of sweep representations that have been studied extensively [4]-[8]. Elsewhere, spheres [9] have been used in structures approximating symmetric surfaces, i.e., three-dimensional generalizations of symmetric axes [10].

Most of the systems referenced to this point are geometric modeling systems [11] primarily developed for computer aided design applications, such as BUILD [2] and PADL [12], or for image interpretation, such as MSYS [13], VISIONS [14], and ACRONYM [15]. These systems usually have interactive procedures for deriving the three-dimensional models and often have a major interest in the suitability of the model for graphical display [16]-[18]. The primary concern of this paper is the development of volumetric descriptions suitable for deriving three-dimensional object representations from

two-dimensional images. Determining such representations has been the goal of many computer vision systems since Roberts' original paper [1]. Roberts demonstrated that three-dimensional information about the actual objects in a single image can be derived under two classes of constraints. The first class included overall scene domain constraints, e.g., the objects were planar faced, while the second class involved specific object constraints, e.g., the objects were combinations of a limited number of polyhedrons. These polyhedrons were the primitives instanced in various sizes and arrangements to form the actual objects in the scene.

Roberts' paper is extremely important for having established a paradigm that many researchers in scene analysis have followed. Attempts have been made to lessen the restrictions imposed by either or both classes of constraints. For example, the specific object constraints were replaced by more extensive domain constraints in the thorough work on line drawings of polyhedral scenes; see [19]-[22]. Different sorts of scene constraints have also been applied. For instance, known properties of special illumination conditions can indicate surface orientations; see [23]-[27].

The constraints can be reduced further by using multiple views of the scene, often with a time ordering resulting in an image sequence [28], [29]. For each view in the sequence feature points can be detected and a correspondence [30] formed between the features in successive views. The image positions of the corresponding feature points provide additional constraints that can be written in the form of a set of nonlinear equations [31]-[33]. Solving the set of equations yields a wire-frame model of the three-dimensional positions of the scene components underlying the image features. The multiple views presented in image sequences can result from either camera or object movement (or both) and provide an additional source of constraints through the control [34], [35] or restriction [36] of the exhibited motions. In the system we have developed the motion creating the multiple views will not be controlled but will be precisely known through viewpoint specifications.

Our work has been in the pursuit of two major goals. The first goal is the development of a system that is capable of deriving three-dimensional object description images, yet does not depend completely on feature point measurements. The second goal is the development of a scheme for representing three-dimensional objects that is descriptive of surface detail, while remaining functional in the context of a structure from multiple view system. The result of this work is a system which uses the occluding contours from multiple images with viewpoint specifications to construct a bounding volume approximating the actual three-dimensional structure of the rigid object generating the contours. The representation constructed to facilitate the creation and continual refinement of the bounding volume is referred to as a "volume segment" representation and will be discussed in Section III.

II. VOLUMES FROM CONTOURS

By an "occluding contour" we mean the boundary in the image plane of the silhouette of an object generated by an orthogonal projection. In an intensity image the silhouette

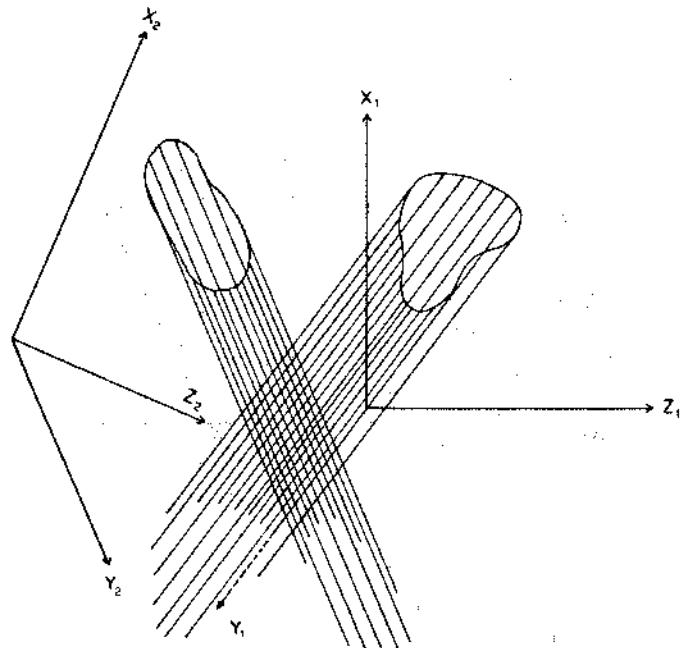


Fig. 1. Silhouettes and contour generating lines for two views.

can most often be formed by a simple thresholding of the intensity values. A connected component analysis [37] of the resulting binary valued image yields the boundary of the object silhouette.

Throughout this paper, the orientation of a view will be specified by a point and three mutually perpendicular directions that form a right-handed, three-axis coordinate system with the given point as origin. The y -axis of each coordinate system is considered to be the direction of the line of sight with the (x, z) -plane being the image plane, i.e., the silhouette is projected onto the image plane by lines parallel to the line of sight. Thus, an occluding contour is generated by the lines that are parallel to the y -axis and that intersect the object, but do so only at points which are on the object surface. For objects with smooth surfaces this means that the lines are in the tangent planes of their intersection points.

Clearly, then, the object surface must touch each contour generating line. The problem is to determine which points on the line are also object surface points. This problem is solved by using the constraints imposed by the contour generating lines from a second (and subsequent) view. The situation of two views is shown in Fig. 1. The contours are displayed in their respective image planes and some representative contour generating lines are indicated by equal length sections of lines that are parallel to the appropriate y -axis. The set of contour generating lines for a given view defines a volume which bounds the actual object, however, this volume is by itself infinite. A second view will also define an infinite volume that contains the object, and if the second view is distinct, i.e., the two lines of sight are not parallel, then the intersection of the two volumes will still encompass the object and will be of finite extent. Intersecting the volumes to define the approximation is in the spirit of the geometric modeling systems [2], [3], [12], yet is a process that can be applied to image data.

It is clear from Fig. 1 that each contour generating line for one of the views serves to constrain the extent of the possible

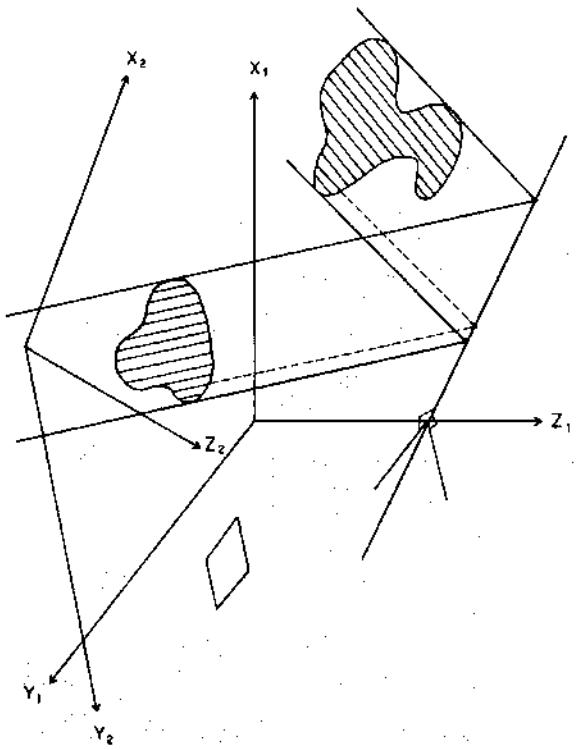


Fig. 2. Scan line derivation of parallelogram constraints.

object surface points on some of the contour generating lines for the other view. In order to establish the initial estimate of the object, the system must determine how each contour generating line constrains the volume from the other view and create a structure that satisfies the various constraints simultaneously.

To see how these constraints are specified consider Fig. 2. Again, two contours are shown in their respective image planes with the lines of sight corresponding to the y -axes. Since these directions are distinct, the image planes are not parallel and thereby must share a common line. The common line for the two image planes is shown on the right-hand side of Fig. 2. This line is in the direction of the cross-product of the two y -axes, and so is perpendicular to both axes, as indicated in the figure. Now, the upper right contour is in the (x_1, z_1) -plane and can be orthogonally projected in that plane onto the common line. The two solid lines, i.e., projection rays, that are tangent to the contour and intersect the common line delimit the projection of the contour onto the common line. In a similar manner, the second contour has two limiting projection rays, as indicated.

The fundamental property of the contours making the construction of the initial estimate possible is that each of the limiting rays for one contour projects onto the common line precisely at the point onto which a limiting ray from the other contour projects. In particular, the projection is orthogonal so that each ray is perpendicular to the common line. Thus, two rays that meet at a point on the common line are coplanar and this plane contains the contour generating lines corresponding to the points on the contour intersected by the limiting rays. It is these coplanar contour generating lines which provide the mutual constraints that determine the bounding volume.

In order to find the coplanar generating lines, each boundary

is rasterized to scan lines in the direction perpendicular to the common line and in the appropriate image plane. That is, the scan lines are chosen parallel to the limiting projection rays and at equal intervals along the common line. The rasterized area is a set of parallel line segments, the endpoints of which specify boundary points of the area (see Newman and Sproull [38] or Martin and Aggarwal [39]). For a given scan line, e.g., the dotted line in Fig. 2, the endpoints of the associated segments determine the boundary points that specify the contour generating lines for the plane containing the scan line. The second contour has a corresponding scan line and thus a set of contour generating lines that are also in the given plane.

The mutual constraints provided by these two sets of contour generating lines are as follows. A given scan line segment in one view has two contour generating lines. These two lines form limits for each of the contour generating lines from the second view that are in the plane defined by the given scan line. The limits specify the extent over which points of the contour generating lines can also be object surface points, i.e., the portion of the contour generating line that is between the limiting lines. Of course, the limited contour generating line is also associated with a scan line segment for the second view and so is paired to another contour generating line for that view. This pair of lines, then, constrains the lines from the first view in the given plane. In particular, the latter pair combines with the original pair from the first view to form mutual constraints on the object volume. The combined constraints are in the form of a parallelogram in the plane containing the scan lines. Each pair of lines specifies a pair of opposing sides on the parallelogram. For example, the dotted scan lines in Fig. 2 form the shown parallelogram.

Each such parallelogram is a bounding figure for the cross section of the object generated by the given plane. The parallelograms circumscribe the cross sections in that a cross section must have at least one point in common with each side of the parallelogram. Of course, there may be more than one parallelogram in a given plane, but that is indicative of the cross section being disconnected with each connected subset bounded by a separate parallelogram. The locus of all the parallelograms in the planes of the various scan lines is the bounding volume which establishes the initial estimate of the object structure.

With the object structure described in this manner, the system is ready to form the volume segment representation which will be used in the subsequent refining process. The next section will define the volume segment representation and then discuss how it is derived from this initial parallelogram structure.

The process described in this section should be contrasted with computer aided tomography [40] and other cross-sectional methods [9], [41], [42] on two main points. First, the source of data is a sequence of simple binary images possibly from a television camera as opposed to the elaborate data acquisition mechanisms required for tomography and range measurement [43]. Second, neither the initial volume specification described above nor the refinement process explained in Section IV depends on predefined cross-sectional decomposition of the data.

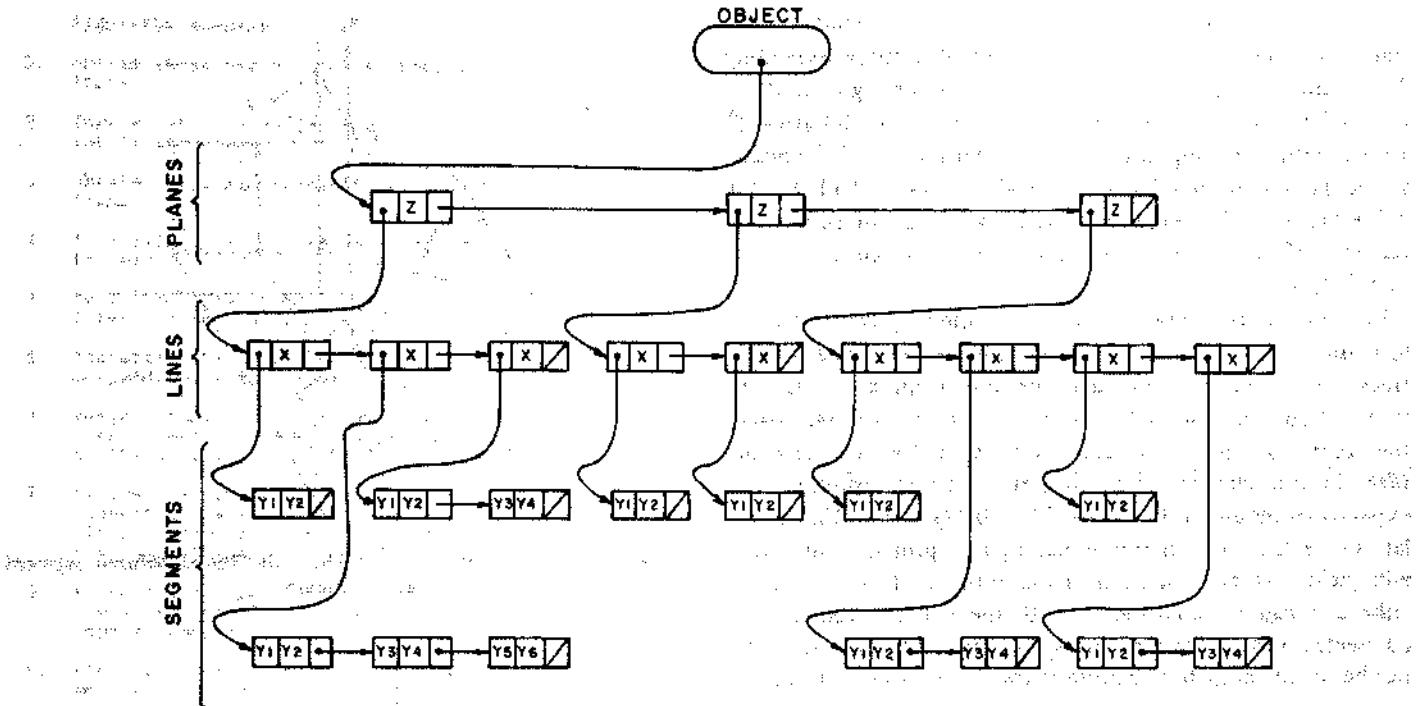


Fig. 3. Data structure schema for the volume segment representation.

III. VOLUME SEGMENT REPRESENTATION

The three-dimensional structure to be derived from the sequence of occluding contours is a bounding volume approximating the actual object. For this reason the representation incorporated in this system is based on volume specification through a "volume segment" data structure. The volume segment representation is a generalization to three dimensions of the rasterized area description. For the rasterized area, the segments each denote a rectangular area. The generalization to three dimensions is to have each segment represent a volume, i.e., a rectilinear parallelepiped with edges parallel to the coordinate axes. In addition to grouping collinear segments into lists, the set of segment lists is partitioned so that the subsets contain lists having coplanar segments. The primary dimension of the rectilinear parallelepiped specified by a segment is the length of the segment, i.e., a rectilinear parallelepiped with edges parallel to the coordinate axes. The second dimension is given by the interline spacing within the plane of the segment, while the third dimension is the interplane distance. The latter two dimensions are specified to be uniform throughout the volume segment representation.

The structure then maintains an ordered pair of values for each volume segment; the values are the y -coordinates of the segment endpoints and are ordered into lists of segments having the same x -coordinate, i.e., collinear. The x -level lists are then coalesced into z -level ordered lists by common z -coordinate, i.e., coplanar. From the top down this structure is a set of "planes" parallel to the $z = 0$ plane, that are ordered by z -value. Each "plane" contains a set of "lines" parallel to the y -axis, that are ordered by x -value. Each "line" comprises a set of disjoint segments that are ordered by endpoint y -value. Fig. 3 shows a schematic of a volume segment structure using linked lists to order the various components.

In a general situation the primary advantage of this structure

is that the process of determining whether an arbitrary point is within the surface boundary consists of a simple search of three ordered lists: select a "plane" by z -coordinate; select a "line" by x -coordinate; and, finally, check for inclusion of the y -coordinate in a segment. The simplicity of this process is in contrast to that required by a representation scheme such as constructive solid geometry [3], wherein the object is represented using regularized set operations on primitive solids. For a particular object represented by given solids, point inclusion is specified by determining if the point is in each primitive solid (possibly, a difficult problem itself) and then performing the logical operations corresponding to the set operations from which the object representation is constructed.

The volume segment structure can also provide a fairly succinct representation, particularly for objects that are elongated in the direction of the y -axis. It becomes more verbose as the elongation extends in the z -axis direction, however, it always remains more compact than surface-ordered enumerative representations, e.g., connected sets of "voxel" faces [44]. In comparing a volume segment structure to a connected voxel face structure one should observe that each leaf in the tree of the volume segment structure represents two voxel faces and remains only two levels from the root of the structure. In addition each voxel face must maintain its three-dimensional position and connections to four edge-adjacent faces. In balance, surface connectivity is not specified directly in the volume segment structure and may require a small amount of searching to compute. As will be shown by the example in Section V, forming a surface description from the volume segments is possible and is computed as a matter of course by this system.

Before proceeding with the details of that system let us indicate a comparison with another volume oriented representation, that of oct-trees [45]. Consider a simple cube

measuring 2^n units on each edge that is embedded with standard orientation in a single octant of a space extending 2^{n+1} units in each direction. For the volume segment structure to represent the cube requires 2^n z-planes each having 2^n x-lines with 2 y-endpoints per line. This yields 2^{2n} volume segments represented by a tree having a total of $(1 + 2^n + 2^{2n} + 1)$ nodes. Note the connected voxel face structure would require 2^{2n} voxel faces for each side of the cube, yielding $6(2^{2n})$ faces.

In contrast, the cube can be represented by an oct-tree of just nine nodes. Of course, this example is a best case for oct-trees. Now, translate the cube one unit in the x direction so that the cube intersects two octants of the embedding space: the volume segment structure and connected voxel face structure do not change. However, the oct-tree representation expands to require $1 + 6 + ((4^{n+2} - 10)/3)$ nodes. Unit translations of the cube in the remaining two principle directions will yield further expansion of the oct-tree. Rotation of the cube can cause expansion in both the volume segment and connected voxel face structures, but these expansions would not be of the magnitude described for the oct-tree structure.

IV. CONSTRUCTION AND REFINEMENT FOR THE VOLUME REPRESENTATION

This section describes both the process by which the volume segment representation is initially constructed from the parallelogram structure detailed in Section II and the process that refines the representation using subsequent views. The first step in the initial construction process is to define a new coordinate system, relative to which the volume segment representation will be specified. Consider, now Fig. 4, in which the contours from two views are displayed in their image planes. Also shown are the lines common to both image planes and a few appropriate scan line segments for the contours. For the bottom scan line of each contour the limiting contour generating lines, i.e., the dotted lines parallel to the respective y-axes, are shown forming the parallelogram for that plane. Also shown are the parallelograms for the planes specified by the remaining scan line segments. Note that the top scan line of the triangular contour is a single point, resulting in a degenerate parallelogram. Again, each parallelogram is in a plane that is perpendicular to the common line. This observation suggests that the common line could be used as the z-axis of the volume segment representation's coordinate system. Making that choice, the y-axis of the new system is defined to be in the direction of the line of sight for the first view, while the x-axis is selected to complete the right-handed coordinate system, e.g., the (x', y', z') -axes in Fig. 4. Of course, either line of sight could be used for the y-axis as it is known that both are perpendicular to the common line. The choice of one of these two directions is made to simplify the next step: parallelogram rasterization.

Each parallelogram is taken as an area in the appropriate (x', y') -plane and rasterized along scan lines parallel to the y' -axis. For the plane of a parallelogram, the rasterization results in a set of line segments parallel to the y' -axis for each x' -value. Thus there is a set of planes each having various lines that are broken into segments. With this information all

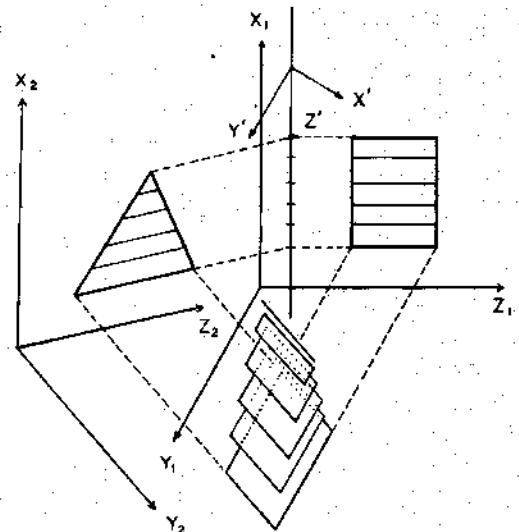


Fig. 4. Parallelogram structure yielding the initial volume segment representation.

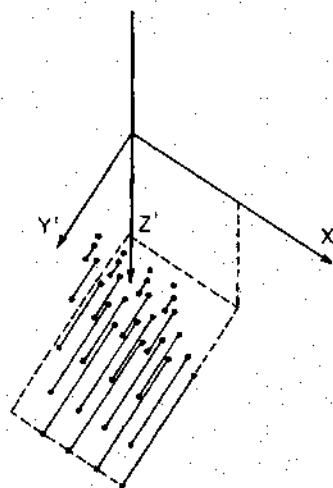


Fig. 5. Volume segment representation in its own coordinate axis system.

that is needed is to add the linking structure and the volume segment representation will be complete. Fig. 5 displays the representation formed for the parallelogram structure of Fig. 4. In Fig. 5 the segments are drawn relative to the representation's coordinate axis system. The dotted line from the z-axis indicates the plane containing the segments derived from what was the bottom parallelogram in Fig. 4. The other segments are in planes nearer the origin. Fig. 6 displays a summary of the algorithm to create the initial volume segment representation that was just explained, while Fig. 7 exhibits a summary of the volume segment representation refinement process that is described in the remainder of this section.

The refinement process is applied as each new frame (after the first two) of the dynamic image is obtained. As was stated earlier, this process makes extensive use of the clipping procedure that is based on rasterized area descriptions. To see how this is done consider Fig. 8. The volume segment representation of Fig. 5 is again shown, however, the contour, image plane and line of sight for a new frame are included in the figure. The overall process is to clip the volume segments

Algorithm summary:

1. obtain image and viewpoint specification for first frame.
2. form silhouette and extract boundary coordinate list for first frame.
3. obtain image and viewpoint specification for second frame.
4. form silhouette and extract boundary coordinate list for second frame.
5. determine line common to the image planes from the first and second frames.
6. rasterize the boundaries along scan line direction perpendicular to the common line.
7. merge mutual constraints from corresponding scan line segments to form the parallelogram description of the object.
8. define the coordinate axis system for the volume segment representation having the z-axis in direction of the common line and the y-axis in the direction of the line of sight for the first frame.
9. rasterize the parallelograms in each z-plane of the new axis system to define the segments of the volume representation.
10. add the linkage structure to complete the volume segment representation.

Fig. 6. Initial representation creation.

Algorithm summary:

1. obtain image and viewpoint specification for new frame.
2. form silhouette and extract boundary coordinate list for new frame.
3. project the y-axis of the volume segment representation's coordinate system along the line of sight and onto the image plane of the new frame.
4. rasterize the new boundary along scan lines that are parallel to the projection of the y-axis in the image plane.
5. for each segment in the volume segment representation do the following:
 - 5a. project the segment along the line of sight and onto the image plane of the new frame.
 - 5b. clip the projected segment to the rasterized boundary.
 - 5c. update the actual segment with respect to the clipped projection.
6. create surface description if desired.
7. continue at step 1 if there are more frames, else stop.

Fig. 7. Continuing representation refinement.

by the new contour. For this to be done, the contour must be rasterized properly. The required direction for the scan lines is the direction in the image plane of the projection, along the new line of sight, of the y-axis for the volume segment representation. The dotted lines in Fig. 8 from the y-axis into the (x', z') -plane indicate this projection. Note that the resulting direction will not normally be parallel to the original y-axis. The contour in Fig. 8 is shown to be rasterized along scan lines parallel to this projected direction.

Given the properly rasterized area description of the new contour the refinement procedure is as follows. Each segment of the volume representation is projected onto the new image plane, again according to the direction of the new line of sight. The projected segment, then, is in the image plane and

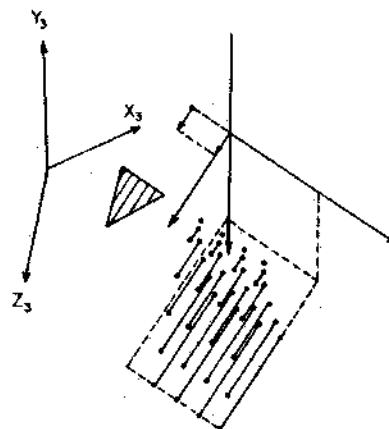


Fig. 8. Refining the volume segment representation by subsequent views.

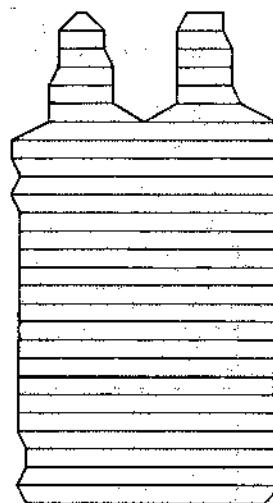


Fig. 9. Contour from first view with "raster" lines inserted.

can be clipped as described previously for the two-dimensional case. The original volume segment can then be updated by modifying its length in a proportion equivalent to that by which the projected segment was clipped. In geometric terms, the clipped segment could be inversely projected onto the line of the original volume segment, with the resulting segment replacing the original segment. A special case occurs when the line of sight happens to be parallel to the y-axis of the coordinate system of the volume segment representation. However, in this case each volume segment projects onto a single point in the image plane and a simple point inclusion test on the area description (rasterized to an arbitrary direction) determines whether the entire volume segment is to be retained in or deleted from the representation.

V. AN EXAMPLE DYNAMIC SCENE

To illustrate the volume segment representation and the process which constructs it, an example is presented in this section. The various stages of the example are shown in Figs. 9-17. The four frames, with raster lines inserted, of the input dynamic scene are presented in Figs. 9-12. In these frames the object rotates about a vertical axis so that it traverses 90° between the first and last views. The combination of the first

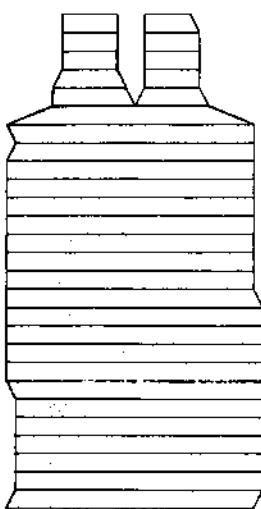


Fig. 10. Contour from second view.

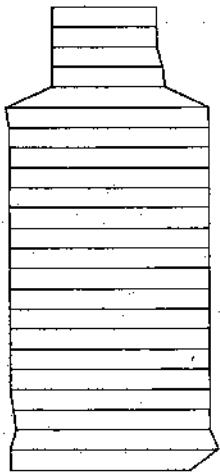


Fig. 11. Contour from third view.

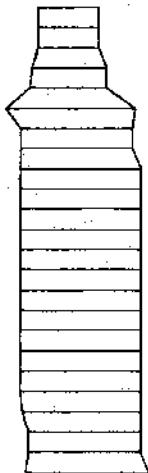


Fig. 12. Contour from the fourth and last view.

two views, i.e., Figs. 9 and 10 results in the surface shown in Fig. 13. Each pair of "raster" lines (one from each view) generates a parallelogram which lies in a plane parallel to the global (x, y) -plane. The parallelograms have been connected along corresponding corners with the top and bottom faces

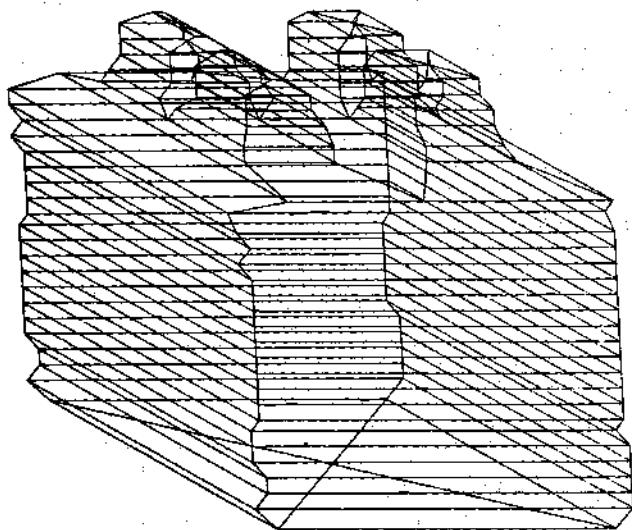


Fig. 13. "Parallelogram cross section" surface description derived from the views displayed in Figs. 9 and 10.

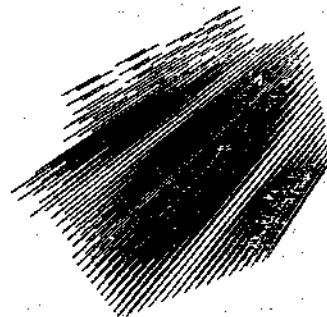


Fig. 14. The volume segment representation of the surface in Fig. 13.



Fig. 15. The volume segment representation derived by constraining the representation shown in Fig. 14 by the contour displayed in Fig. 11.

marked by crossing lines. The volume segment representation resulting from this surface is exhibited in Fig. 14. As stated earlier, each segment is parallel to the global y -axis. In Fig. 14 perspective cues have been added to provide the proper sense of depth. The third and fourth frames are then processed to constrain the volume segment representation to be as it is displayed in Figs. 15 and 16, respectively. Finally, the volume segment representation of Fig. 16 is transformed into a surface description, as illustrated in Fig. 17. It should be noted that the surface representation of Fig. 17 is derived directly from the volume segment representation and is much more general than the surface description shown in Fig. 13.



Fig. 16. The final volume segment representation for the dynamic scene.

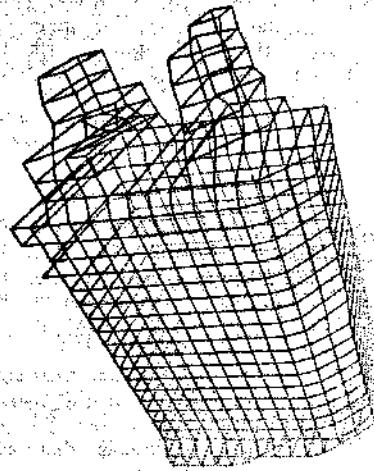


Fig. 17. The general surface description derived from the volume segment representation illustrated in Fig. 16.

VI. CONCLUSION

The two major goals pursued in this work are, first, to lessen the dependence on feature point measurements in a structure from multiple views system, and second, to develop a descriptive three-dimensional object representation that was suitable for a dynamic process of volume refinement. The results are twofold: 1) a system has been developed which constructs a volumetric structure for an object from a sequence of occluding contours and 2) an algorithm has been formulated for the representation and refinement of this structure.

The occluding contours with viewpoint specifications from a dynamic image are analyzed to initially form and continually update a description of the three-dimensional object generating the contours. The description is a bounding volume for the object and is successively refined to yield finer approximations to the actual object. Of course, from the silhouettes that form the occluding contours it is not possible to resolve certain kinds of concavities. In particular, object surface points for which every tangent line (in the tangent plane, any line that contains the given surface point) also intersects the object at some nonsurface point cannot be resolved using silhouettes. However, the class of objects that can be described exactly is large, and in fact, the object surface may have saddle points and holes.

Clearly, to analyze the structure of objects a system must provide a representation scheme. For three-dimensional objects many different schemes have been proposed and used.

The details of the representations usually are determined either by the data acquisition techniques or by the ultimate application of the system, with an important problem being the development of methods for transforming between structures of the first type and structures of the second type; see Aggarwal *et al.* [46]. The volume segment representation described in this paper has been developed to facilitate the acquisition of three-dimensional information from dynamic images. The main attributes of the volume segment representation are that it is easy to update (as required by the continual refinement), maintains fine surface detail, simplifies the point inclusion test, and can be readily transformed into a surface representation.

For these reasons the work presented in this paper provides an excellent basis for further research. In particular, the work is appropriate to industrial automation applications. For example, selecting one of several parts on a conveyor using the views taken from several cameras fixed along the line of travel, or in conjunction with a manipulator arm that could successively reposition a part until an adequate approximation was derived. In such applications there are usually fixed sets of possible objects from which an unknown object must be recognized, implying the need for a representation scheme suitable for creating a library of possible objects, describing the unknown sample and matching it to the library entries. Future research should be directed toward exploiting the methods developed here in those applications.

REFERENCES

- [1] L. G. Roberts, "Machine perception of three-dimensional solids," in *Computer Methods in Image Analysis*, J. K. Aggarwal, R. O. Duda, and A. Rosenfeld, Eds. New York: IEEE Press, 1977, pp. 285-323.
- [2] I. C. Braid, "The synthesis of solids bounded by many faces," *Commun. Ass. Comput. Mach.*, vol. 18, pp. 209-216, Apr. 1975.
- [3] A. A. G. Requicha, "Representations for rigid solids: Theory, method, and systems," *Comput. Surveys*, vol. 12, no. 4, pp. 437-464, 1980.
- [4] G. J. Agin and T. O. Binford, "Computer description of curved objects," *IEEE Trans. Comput.*, vol. C-25, pp. 439-449, Apr. 1976.
- [5] J. Hollerbach, "Hierarchical shape description of objects by selection and modification of prototypes," MIT AI-TR-346, Nov. 1975.
- [6] B. I. Soroka and R. K. Bajcsy, "A program for describing complex three-dimensional objects using generalized cylinders as primitives," presented at the IEEE Conf. Pattern Recognition and Image Processing, Chicago, IL, 1978.
- [7] R. Nevatia and T. O. Binford, "Description and recognition of curved objects," *Artificial Intell.*, vol. 8, pp. 77-98, 1977.
- [8] D. Marr, "Visual information processing: The structure and creation of visual representations," in *Proc. IJCAI-6*, Tokyo, Aug. 1979, pp. 1108-1126.
- [9] J. O'Rourke and N. Badler, "Decomposition of three-dimensional objects into spheres," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 295-305, July 1979.
- [10] H. Blum, "A transformation for extracting new descriptors of shape," in *Models for the Perception of Speech and Visual Form*, W. Dunn, Ed. Cambridge, MA: MIT Press, 1964, pp. 362-380.
- [11] B. G. Baumgart, "Geometric modeling for computer vision," Stanford AI Lab. Memo. AIM-249, Oct. 1974.
- [12] H. B. Voelcker, A. A. G. Requicha, E. E. Hartquist, W. B. Fisher, J. E. Shapiro, and N. K. Birrell, "An introduction to PADL:

- Characteristics, status, and rationale," Univ. Rochester, Prod. Automat. Project Tech. Memo. TM-22, Dec. 1974.
- [13] H. G. Barrow and J. M. Tenenbaum, "MSYS: A system for reasoning about scenes," SRI AI Center, Tech. Note 121, Mar. 1976.
- [14] A. Hansen and E. Riseman, "VISIONS: A computer system for interpreting scenes," in *Computer Vision Systems*, A. Hansen and E. Riseman, Eds. New York: Academic, 1978, pp. 303-333.
- [15] R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," Dep. Comput. Sci., Stanford Univ., Rep. STAN-CS-81-861, June 1981.
- [16] E. Miyamoto and T. O. Binford, "Display generated by a generalized cone representation," presented at the IEEE Conf. Comput. Graphics and Image Processing, May 1975.
- [17] J. H. Clark, "Hierarchical geometric models for visible surface algorithms," *Commun. Ass. Comput. Mach.*, vol. 19, pp. 547-554, Oct. 1976.
- [18] N. Badler and R. K. Bajcsy, "Three-dimensional representations for computer graphics and computer vision," *Comput. Graphics*, vol. 12, pp. 153-160, Aug. 1978.
- [19] D. Huffman, "Impossible objects as nonsense sentences," in *Machine Intelligence 6*, B. Meltzer and D. Michie, Eds. Edinburgh, Scotland: Edinburgh Univ. Press, 1971.
- [20] M. Clowes, "On seeing things," *Artificial Intell. J.*, vol. 2, no. 1, pp. 79-116, 1971.
- [21] D. Waltz, "Understanding line drawings of scenes with shadows," in *Psychology of Computer Vision*, P. H. Winston, Ed. New York: McGraw-Hill, 1975, pp. 19-92.
- [22] A. Mackworth, "Interpreting pictures of polyhedral scenes," *Artificial Intell. J.*, vol. 4, pp. 121-137, 1973.
- [23] P. M. Will and K. S. Pennington, "Grid coding: A preprocessing technique for robot and machine vision," in *Proc. 2nd IJCAI*, London, 1971, pp. 66-70.
- [24] B. K. P. Horn, "Understanding image intensities," *Artificial Intell. J.*, vol. 8, pp. 201, 231, 1977.
- [25] R. J. Woodham, "A cooperative algorithm for determining surface orientation from a single view," in *Proc. 5th Int. Joint Conf. Artificial Intell.*, Cambridge, MA, 1977, pp. 635-641.
- [26] Y. Shirai and M. Suiva, "Recognition of polyhedrons with a range finder," in *Proc. 2nd Int. Joint Conf. Artificial Intell.*, London, 1971, pp. 71-79.
- [27] H. Freeman and M. Potmesil, "Curved surface representation utilizing data extracted from multiple photographic images," in *Proc. Workshop on the Representation of Three-Dimensional Objects*, Philadelphia, PA, 1979.
- [28] W. N. Martin and J. K. Aggarwal, "Survey: Dynamic scene analysis," *Comput. Graphics Image Processing*, vol. 7, pp. 356-374, June 1978.
- [29] H.-H. Nagel, "Image sequence analysis: What can we learn from applications?," in *Image Sequence Analysis*, T. S. Huang, Ed. Heidelberg, Germany: Springer-Verlag, 1981, pp. 19-228.
- [30] J. K. Aggarwal, L. S. Davis, and W. N. Martin, "Correspondence processes in dynamic scene analysis," *Proc. IEEE*, vol. 69, no. 5, pp. 562-572, May 1981.
- [31] S. Ullman, *The Interpretation of Visual Motion*. Cambridge, MA: MIT Press, 1979.
- [32] J. W. Roach and J. K. Aggarwal, "Determining the movement of objects from a sequence of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 554-562, Nov. 1980.
- [33] H.-H. Nagel, "Representation of moving rigid objects based on visual observations," *IEEE Computer*, vol. 14, no. 8, pp. 29-39, Aug. 1981.
- [34] R. Nevatia, "Depth measurement by motion stereo," *Comput. Graphics Image Processing*, vol. 15, pp. 203-214, 1976.
- [35] H. H. Baker, "Three-dimensional modelling," in *Proc. 2nd IJCAI*, London, 1977, pp. 649-655.
- [36] J. A. Webb and J. K. Aggarwal, "Visually interpreting the motion of objects in space," *IEEE Computer*, vol. 14, pp. 40-46, Aug. 1981.
- [37] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic, 1976, pp. 336-347.
- [38] W. E. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, 2nd ed. New York: McGraw-Hill, 1979, pp. 229-245.
- [39] W. N. Martin and J. K. Aggarwal, "Analyzing dynamic scenes," Lab. for Image and Signal Anal., Univ. Texas, Austin, Rep. TR-81-5, pp. 51-56, Dec. 1981.
- [40] G. T. Herman, *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*. New York: Academic, 1980.
- [41] C. Levinthal and R. Ware, "Three-dimensional reconstruction from serial sections," *Nature*, vol. 236, pp. 207-210, Mar. 1972.
- [42] H. Fuchs, Z. M. Kedem, and S. P. Uselton, "Optimal surface reconstruction from planar contours," *Commun. Ass. Comput. Mach.*, vol. 20, pp. 693-702, Oct. 1977.
- [43] D. Nitzan, A. E. Brain, and R. O. Duda, "The measurement and use of registered reflectance and range data in scene analysis," *Proc. IEEE*, vol. 65, pp. 206-220, Feb. 1977.
- [44] G. T. Herman and H. K. Liu, "Dynamic boundary surface detection," *Comput. Graphics Image Processing*, vol. 7, pp. 130-138, 1978.
- [45] C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects," *Comput. Graphics Image Processing*, vol. 14, pp. 249-270, 1980.
- [46] J. K. Aggarwal, L. S. Davis, W. N. Martin, and J. W. Roach, "Survey: Representation methods for three-dimensional objects," in *Progress in Pattern Recognition*, vol. 1, L. Kanal and A. Rosenfeld, Eds. Amsterdam, The Netherlands: North-Holland, 1981, pp. 337-391.



Worthy N. Martin (S'78-M'79-S'80-M'81) was born on November 24, 1951. He received the B.A. degree in mathematics, the M.A. degree in computer science, and the Ph.D. degree in computer science in 1981, all from the University of Texas, Austin.

He is currently an Assistant Professor of Computer Science at the Department of Applied Mathematics and Computer Sciences, University of Virginia, Charlottesville. From 1980 to 1982, he was associated with the Laboratory for Image and Signal Analysis and during the 1981-1982 academic year was on the faculty of the Department of Computer Sciences at the University of Texas, Austin.

Laboratory for Image and Signal Analysis and during the 1981-1982 academic year was on the faculty of the Department of Computer Sciences at the University of Texas, Austin.

Dr. Martin is a member of the IEEE Computer Society and the Association for Computing Machinery.



J. K. Aggarwal (S'62-M'65-SM'74-F'76) received the B.S. degree in mathematics and physics from the University of Bombay in 1956, the B.Eng. degree from the University of Liverpool, Liverpool, England, in 1960, and the M.S. and Ph.D. degrees from the University of Illinois, Urbana, in 1961 and 1964, respectively.

He joined the University of Texas in 1964 as an Assistant Professor and has since held positions as Associate Professor (1968) and Professor (1972). Currently, he is the John J. McKetta Energy Professor of Electrical Engineering and of Computer Sciences at the University of Texas, Austin. Further, he was a visiting Assistant Professor to Brown University, Providence, RI (1968), and a visiting Associate Professor to the University of California, Berkeley, during 1969-1970. He has published numerous technical papers and several books, *Notes on Nonlinear Systems* (1972), *Nonlinear Systems: Stability Analysis* (1977), *Computer Methods in Image Analysis* (1977), *Digital Signal Processing* (1979), and *Deconvolution of Seismic Data* (1982). His current research interests are image processing and digital filters.

Dr. Aggarwal is an active member of the IEEE, Pattern Recognition Society, and Eta Kappa Nu. He was an ADCOM member of the Circuits and Systems Society (1972-1975), Chairman of the Technical Committee on Signal Processing (1974-1975), and Co-Editor of the Special Issues on Digital Filtering and Image Processing (CAS TRANSACTIONS, March 1975) and on Motion and Time Varying Imagery (PAMI TRANSACTIONS, November 1980). Currently he is an Associate Editor of the journal *Pattern Recognition*, and Guest Editor for a Special Issue of *Computer Graphics and Image Processing*.

Model-Based Three-Dimensional Interpretations of Two-Dimensional Images

RODNEY A. BROOKS

Abstract—ACRONYM is a comprehensive domain independent model-based system for vision and manipulation related tasks. Many of its submodules and representations have been described elsewhere. Here the derivation and use of invariants for image feature prediction is described. Predictions of image features and their relations are made from three-dimensional geometric models. Instructions are generated which tell the interpretation algorithms how to make use of image feature measurements to derive three-dimensional size, structural, and spatial constraints on the original three-dimensional models. Some preliminary examples of ACRONYM's interpretations of aerial images are shown.

Index Terms—Algebraic vision, computer vision, constraint systems, geometric models, model-based vision, spatial reasoning.

I. INTRODUCTION

THE ACRONYM system has always relied on detailed three-dimensional geometric models to direct image understanding. Originally, it used models of specific objects to direct a qualitative geometric reasoning system in labeling images [8].

The scope of ACRONYM has now been increased to include object class recognition and extraction of three-dimensional information from images (including monocular images), reasoning about how to grasp objects [3], and real-time simulation of multiple manipulator work stations for purposes of off-line programming and the design and analysis of new manipulators [11]. ACRONYM has moved from using a purely geometric representation and qualitative geometric reasoning system to a system with a combined algebraic and geometric representation and a geometric reasoning system which can make precise deductions about partially specified situations. The geometric and algebraic aspects of the representation complement each other during image interpretation.

To support these extended capabilities a large number of new techniques had to be developed. They include the addition of a class and subclass relation representation scheme to the geometric modeling system. This is based on the use of symbolic algebraic constraints. In support of this a con-

Manuscript received December 12, 1981; revised September 8, 1982. This work was supported in part by the Defense Advanced Research Projects Agency under Contract MDA903-80-C-0102, in part by the National Science Foundation under Contract DAR78-15914, and in part by a grant from the ALCOA Corporation. An earlier version of this paper was presented at the International Joint Conference on Artificial Intelligence, Vancouver, Canada, August 1981.

The author was with the Stanford Artificial Intelligence Laboratory, Stanford University, Stanford, CA 94305. He is now with the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139.

straint manipulation system which includes a partial decision procedure on consistency of sets of nonlinear inequalities was formulated and implemented. A geometric reasoning system which can deal with underconstrained spatial relations was developed. These are all detailed in [7].

This paper deals with techniques developed for image feature and feature-relation prediction, and a matching process which is directed by both geometric relations and algebraic constraints during image interpretation. Some first examples of the performance of the re-built ACRONYM system on some real images are presented. The low-level bottom-up processes used in these experiments (for reasons of expedience) provide either little or noisy data. Nevertheless, ACRONYM makes strong and accurate deductions about the objects appearing in the images. Even better performance is expected when more accurate low-level descriptive processes become available.

A. Model Domain

The reader is referred to Brooks [7] for a complete description of the ACRONYM modeling system. A short overview is given here to make the paper reasonably self-contained.

An *a priori* model of the world is given to ACRONYM. The world is modeled as a coordinate system. Objects are modeled as subpart hierarchies of generalized cones, each with a local coordinate system related to an object coordinate system. Cameras are modeled as coordinate systems (with viewing direction along the -z axis) and a focal ratio. Objects and cameras are placed in the world by constraining the transforms between their local coordinate systems and the world coordinate system.

The particular domain considered in this paper is aerial views of airfields. However, ACRONYM has no particular knowledge of airfields and is applicable to other domains, and camera geometries other than aerial cameras—an example of such is given in Section II.

Generalized cones, introduced by Binford [2], describe a three-dimensional volume. A generalized cone is represented by a curve through space, called the spine, along which a two-dimensional shape, called the cross section, is swept. The cross section is kept at a constant angle to the tangent of the spine, and is deformed according to a deformation function called the sweeping rule.

In ACRONYM generalized cones are restricted to having straight line segments or circular arcs as spines, cross sections bounded by straight lines and circular arcs, and sweeping rules which are linear magnification functions about the spine, or

linear in two orthogonal directions about the spine. By convention in ACRONYM the coordinate systems of generalized cones have their origins at one end of the spine, with the cross section laying in the y - z plane at that point, and the spine extending into the positive x half-space. For cross sections which are to be kept normal to the spine this implies that the tangent of the spine at the origin lies along the x -axis.

In ACRONYM, generalized cones, the subpart hierarchy of objects, and the affixment tree representing spatial relationships of objects are all represented as *units* with *slots* and *fillers* (e.g., Bobrow and Winograd [5]). Thus, a generalized cone is represented by a unit with slots called *SPINE*, *CROSS-SECTION*, and *SWEEPING-RULE*, each of which is filled with a unit of the appropriate type. Eventually, the units hierarchy bottoms out in units with slots filled with numbers. For instance, a spine unit for a straight spine has a slot called *LENGTH* which can be filled with a number representing the length of the spine.

The above describes a representation scheme sufficiently rich to model a large class of specific objects where all parameters are completely specified. However, for image interpretation tasks it is often the case that not all details of objects in the world are known *a priori*. Thus, it is desirable to be able to represent classes of objects. More particularly, it is often desirable to be able to talk about classes of objects classified according to their function. For instance, an image interpretation task might involve identification of automobiles, buildings, or oil refineries.

Fortunately, the functionality of many man-made objects is reflected in the geometric structure. For instance, automobiles have four cylindrical sections (wheels) in contact with the ground and with their axes parallel to the ground plane. The cylindrical sections are paired with colinear axes, and the centroids of the cylinders form a rectangle (or perhaps an isosceles trapezium). An essentially box-like structure is above the ground with base parallel to the ground and below the rectangle formed by the wheelbase. In addition there may be a section of the box-like structure which extends higher than the rest which is as wide as the main body but not as long. Such general geometric descriptions of functional classes will be referred to as *generic* descriptions throughout the rest of the paper. ACRONYM includes mechanisms for representing such generic object classes.

Representation of geometric classes is achieved by representing commonality of class members by shared structure, and variations across class members by generalizing the fillers of slots which ordinarily might contain a number. Generalizing the slots of the generalized cones allows representation of variations in size and shape, the slots in the subpart hierarchy allow representation of variations in structure, and those in the affixment tree allow variations in spatial relationships. The generalization is to allow numbers to be replaced by formal variables (referred to below as *quantifiers*) and algebraic expressions over formal variables. Additionally, constraints are placed on the formal variables (generally as inequalities, not necessarily linear). The constraints and the geometric unit slot representations together represent the class of all objects which have the given geometric structure and whose parameters

satisfy the constraints. The reader is referred to Brooks [7] for details of this modeling scheme.

As an example of a use of quantifiers consider the problem of representing the fact that airplanes on the ground are usually upright with their wings and fuselage parallel to the ground, but be headed in arbitrary directions. Also they can be found at many places at ground level within the confines of an airfield. Suppose the world coordinate system has $z = 0$ as the ground plane, with positive z above the ground. Then the coordinate transform relating the coordinate system of the airplane to the ground would consist of a translation and a rotation. The translation would have its x and y component slots filled by quantifiers bounded only to lie within the confines of the airfield (for a polygonal shaped airfield this would simply be a conjunction of linear inequalities over the two quantifiers), and its z slot filled by some constant dependent on the airplane (the height of the airplane origin above the ground). The rotation would be about the z axis with its magnitude slot filled by a quantifier bound only to lie in the range 0 to 2π .

B. Image Domain

ACRONYM is given images which have been preprocessed by a line finder due to Nevatia and Babu [9]. Figs. 3(b), 4(b), and 5(b) show examples of this processing.

From the edges, a rather poor performance module extracts descriptions of the images as ribbons and ellipses. A ribbon is a two-dimensional analog of a generalized cone. In the ACRONYM implementation they are restricted to having straight spines and linear sweeping rules. Ellipses are described by the lengths of their major and minor axes.

In addition to the ribbons and ellipses ACRONYM is told the resolution in pixels of the original digitization. It uses this to estimate pixel error magnitudes.

In the examples of this paper ACRONYM is given preprocessed images of airfields with a number of airplanes in view on the ground.

C. The Image Interpretation Task

This paper is concerned with the image interpretation tasks performed by ACRONYM. It is given some classes of generic geometric models. The world is described in terms of the typical coordinate transforms of those objects to the world coordinate system (as in the example of the airplane on the ground, above). A camera is also modeled by the relationship of its coordinate system to the world coordinate system.

ACRONYM is given a preprocessed image as above. Its task is to identify instances of the object model classes in the image, along with their location and orientation in world coordinates, to make any subclass identification which is possible, to determine constraints implied by the image on the quantifiers in the models (i.e., determine three-dimensional parameters of the objects from the image) and finally to determine the location and orientation of the camera if that was not completely determined *a priori*.

In the examples of Section III the task which ACRONYM must carry out is to locate airplanes and identify their type. In addition it determines parameters of the camera.

II. PREDICTION

In the ACRONYM system generic object classes and specific objects are represented by volumetric models based on generalized cones along with a partial order on sets of nonlinear algebraic inequalities relating model parameters. Image features and relations between them which are invariant over variations in the models and camera parameters are identified by a geometric reasoning system. Such predictions are combined first to give guidance to low-level image description processes, then to provide coarse filters on image features which are to be matched to local predictions. Predictions also contain instructions on how to use noisy measurements from identified image features to construct algebraic constraints on the original three-dimensional models. Local matches are combined subject both to consistently meeting predicted image feature relations, and the formation of consistent sets of algebraic constraints derived from the image. The result is a three-dimensional interpretation of the image.

This section describes some of the invariants that are identified by the reasoning system, and gives examples of how the back constraints are set up giving three-dimensional information about the instances of the models which appear in images.

A. Constraints

To illuminate the discussion in succeeding subsections the uses and capabilities of ACRONYM's constraint mechanism are briefly described along with the allowed structure of constraints themselves.

ACRONYM's three-dimensional models are represented by *units* and *slots* (e.g., Bobrow and Winograd [5]). Any slot which admits numeric *fillers* also admits *quantifiers* (pre-declared variable names) and expressions over quantifiers using the operators +, -, X, /, and $\sqrt{}$.

Constraints can be put on quantifiers. They take the form of inequalities between expressions as defined above, along with the possibility of including max and min (on the left and right of \leq , respectively). Equality can be encoded as two inequalities. For instance suppose a cylinder is represented as a generalized cone whose straight spine has its length defined by the quantifier CYL-LENGTH and whose cross section is a circle with radius CYL-RADIUS. Then the class of all cylinders of volume 5 (in some units) can be represented by the two constraints:

$$5 \geq \text{CYL-LENGTH} \times \text{CYL-RADIUS} \times \text{CYL-RADIUS} \times \pi$$

$$5 \leq \text{CYL-LENGTH} \times \text{CYL-RADIUS} \times \text{CYL-RADIUS} \times \pi$$

The ACRONYM constraint manipulation system (CMS), described in detail in [7], operates on sets of constraints. A set of constraints (implicitly conjunctive) defines a subset of n -dimensional space for which all constraints are true (where n is the number of quantifiers mentioned in the constraint set). This is called the satisfying set, and is empty if the constraints are inconsistent. The CMS is used for three tasks related to this constraint set.

- 1) Given a set of constraints partially decide whether their satisfying set is empty. The outcomes are "empty" or "I don't know."
- 2) Find numeric (or $\pm\infty$) upper and lower bounds on an ex-

pression in quantifiers over the satisfying set of a constraint set. This uses procedures called SUP and INF.

3) (A generalization of task 2.) For an expression E and a set of quantifiers V find expressions L and H in V such that $L \leq E \leq H$ identically over the satisfying set of the constraint set.

In tasks 2 and 3 the expressions being bounded can include trigonometric functions such as sin, cos, and arcsin.

The CMS implemented in ACRONYM is a nonlinear generalization [7] of the linear SUP-INF method described by Bledsoe [4] and Shostak [10]. In [7] it was shown that ACRONYM's CMS behaves identically to that described by Shostak for purely linear sets of constraints and linear expressions, and determines least upper and greatest lower bounds.

For nonlinear expressions and constraints there is not such a well delineated characterization of the behavior of the CMS. The proofs of [7] can be extended to cases of sums of independent terms, each of which is a product or quotient of terms whose sign can be determined purely from the subset of constraints which are linear, i.e., in those cases the CMS produces the best bounds possible.

Informal empirical evidence suggests that there are other cases where the CMS produces good bounds. No good characterization of these cases exists. It is also possible to demonstrate cases where the bounds found by the CMS are not good. If all terms have signs determinable from the subset of constraints which are linear then the bounds are no worse than considering all terms to be independent. If some terms have indeterminate signs then the bounds may be quite poor.

Much of the expertise in the ACRONYM system lies in being able to reduce sets of constraints by projection into subspaces where better bounds can be found on expressions. Sometimes, of course, it is not possible to find such a projection, and then poor bounds must suffice. The rules used for prediction and interpretation were written with these limitations in mind.

B. The Prediction Process

The prediction module of ACRONYM is implemented as a set of approximately 280 product-type rules.

The major control paradigm is backward chaining, i.e., rules set up subgoals and recursively invoke the rule mechanism to satisfy those subgoals. Both subgoals and rule capabilities are represented as Lisp s-expressions. Rules are invoked if they unify with a stated subgoal. The unification process allows passing of multiple parameters to rules, and receiving multiple results from them. In addition, the pattern matching aspect of unification provides rule selection criteria.

A global assertional database, and additional local databases provide the means for recording the state of the prediction computation. The data structures embodying the predictions are built as side effects of the rules firing.

During a typical prediction phase, e.g., in the example of Section III of this paper, there are on the order of 6000 rule firings.

The order of rule firings, and flow of control can not be characterized at a local level as it is completely dependent on the models given to the system. At a global level the order of computations can be roughly described as follows.

A breadth first walk down the subpart hierarchy of the

models is made. At each level prediction is carried out, followed by partial interpretations of the images. Then refined predictions are made at the next level of the hierarchy. (In the examples of Section III only the first level of interpretations are carried out).

At each level the coordinate transforms of the generalized cones, relative to the camera are computed and examined. Visibility conditions and implications of visibility are computed. Individual shape predictions are made for the generalized cones. Relationships between the generalized cones are examined for invariant characterizations. These four steps are summarized in more detail in the following paragraphs.

The affixment tree is transversed to get a symbolic expression for the coordinate transform relating a generalized cone's local coordinate system to that of the camera. The transform is represented as a product of rotations and translations. Typically, these products are long (i.e., over ten terms) and contain many quantifiers. The reader is referred to [7] for the details of some rules which make such expressions manageable. During later phases of prediction these rules are applied in a goal directed manner to find a simplification of the products in the form most suitable for the task at hand. (Again examples are given in [7].)

One can associate with a camera a volume of space (an infinite pyramid with rectangular cross section and apex at the focal point of the camera) in which objects can be visible. Outside of that volume an object is definitely invisible to a camera. Thus, for the purposes of predicting the appearance of a generalized cone it can be assumed that it lies within the sight of the camera, since otherwise it will not be seen anyway. Therefore, constraints are added to the object model which confine its coordinates so that it lies within the visible volume. If these constraints are inconsistent with those that already exist then the object is definitely invisible and no further prediction need be made. Otherwise it provides possibly tighter constraints on the possible range of positions and orientations possible for the object. Once the visibility conditions for individual generalized cones are established, pairwise comparisons are made to see if it can definitely be established that one cone always occludes another. This process is carried out by tentatively adding constraints that imply the converse (i.e., one cone never occludes the other). If these constraints are inconsistent then the obscuration must always occur. If they cannot be shown to be inconsistent (recall that the decision procedure is only partial) then it can be concluded that perhaps the obscuration will not always occur, so both cones may be visible.

Actual predictions of shapes proceed in five phases. These are described in detail in Section II-C.

An exhaustive pairwise examination of shapes produced by different generalized cones is carried out. The geometric reasoning system tries to find invariant characterizations of the relationships between the shapes in terms of the relationships detailed in Section II-D.

C. Shape Prediction

Shapes are predicted as *ribbons* (the two-dimensional analog of three-dimensional generalized cones) and *ellipses*. These are also the features which are found by the low-level descriptive process which are temporarily being used in ACRONYM.

Ribbons are a good way of describing the images generated by generalized cones. Consider a ribbon which corresponds to the image of the swept surface of a generalized cone. For straight spines, the projection of the cone spine into the image would closely correspond to the spine of the ribbon. Thus, a good approximation to the observed angle between the spines of two generalized cones is the angle between the spines of the two ribbons in the image corresponding to their swept surfaces. A quantitative theory of these correspondences is not used. Ellipses are a good way of describing the shapes generated by the ends of generalized cones. The perspective projections of ends of cones with circular cross-sections are exactly ellipses.

Shape prediction involves deciding what shapes will be visible, predicting ranges for shape parameters (to be used as a coarse filter during interpretation and also to guide the low-level descriptive processes) and deriving instructions about how to locally invert the perspective transform and hence use image measurements to generate constraints on the original three-dimensional models.

To predict the shapes generated by a single generalized cone, ACRONYM does not explicitly predict all possible quantitatively different viewpoints. Rather, it predicts what shapes may appear in the image, and associates with them methods to compute constraints on the model that are implied by their individual appearance in an image. For example, identification of the image of the swept surface of a right circular cone constrains the relative orientation of the cylinder to the camera (these are called back constraints). Identification of an end face of the cylinder provides a different set of constraints. If both the swept surface and an end face are identified then both sets of constraints apply. Also predicted are specific relations between shapes that will be true if they are both observed correctly. For more complex cones, the payoff is even greater for predicting individual shapes rather than exhaustive analysis of which shapes can appear together.

At other times during prediction invariant cases of obscuration are noticed. For instance, it may be noticed that one cone abuts another so that its end face will never be visible. The consequences of such realizations are propagated through the predictions.

Prediction of shapes proceeds in five phases. First, all the contours of a generalized cone which could give rise to image shapes are identified by a set of special purpose rules. These include occluding contours and contours due purely to internal cone faces. Thus, for instance, a right square cylinder will generate contours for the end faces, the swept faces, and contours generated by the swept edges at diagonally vertices of the square cross section. The contours are generated independently of camera orientation, and in terms of object dimensions rather than image quantities.

Second, the orientation of the generalized cone relative to the camera (this is done by the geometric reasoning system; see [7]) is then examined to decide which contours will be visible and how their image shapes will be distorted over the range of variations in the model parameters which appear in the orientation expressions.

The third phase predicts relations between contours of a single generalized cone (see Section II-D).

Fourth, the actual shapes are then predicted. The expected

values for shape parameters in the image are estimated as closed intervals (see below).

Finally, the back constraints which will be instantiated during interpretation are constructed.

1) Back Constraints: Consider the following simple camera geometry. Suppose that it is desired to predict the length of an observable feature which is generated by something of length l lying in a plane parallel to the camera image plane, at distance d from the camera. Furthermore, suppose the camera has a focal ratio of f . Then the measured length of the observed feature is given by $p = (l \times f)/d$. Any or all of l , f , and d may be expressions in quantifiers, rather than numbers. Using the CMS bounds can be obtained on the above expression for image feature length, giving that it will lie in some range $P = [p_l, p_h]$ where p_l and p_h are either numbers or $\pm\infty$. For more complex geometries the expression for p will be more complex, but the method is the same (trigonometric functions are usually involved).

Now, given an image feature, which is hypothesized to correspond to the prediction it must be decided whether it is acceptable on the basis of its parameters. The low-level descriptive processes are noisy and provide an error interval, rather than an exact measurement for image parameters. Suppose the interval is $M = [m_l, m_h]$ for a feature parameter predicted with expression p . Then the parameter is acceptable if $P \cap M$ is nonempty. This is the coarse filtering used during initial hypothesis of image feature to feature prediction matches.

But note also that it must be true that the true value of p for the particular instance of the model which is being imaged must lie in the range M . Thus the constraints

$$m_l \leq (l \times f)/d$$

$$m_h \geq (l \times f)/d$$

can be added to the instance of the model being hypothesized, where l , f , and d are numbers or expressions in quantifiers.

The above is the analysis for the simplest possible camera geometry. In general the predicted geometry is much more complex and requires stronger symbolic analysis methods. ACRONYM has individual rules which are capable of handling all cases of up to three orthogonal degrees of freedom in orientation of objects relative to the camera.

2) Trigonometric Back Constraints: When the expression p involves trigonometric functions the above method of generating back constraints will not work. It would generate constraints involving trigonometric functions, which ACRONYM's CMS cannot handle.

One approach to this problem is to bound expression p above and below by expressions involving no quantifiers contained in arguments to trigonometric functions, and then use these expressions in setting up the back constraints. This has the unfortunate side effect of losing all information implied by the image feature about the quantifiers eliminated from the bounds.

A second approach is sometimes applicable. If a trigonometric function has as its argument e an expression, and if the CMS determines that e is bounded to lie within a region of the function's domain where it is strictly monotonic and

hence invertible, then specific back constraints on e can be computed at interpretation time (as distinct from during prediction). An example illustrates this. A cylinder with length CYL-LENGTH is sitting upright on a table. A camera with unknown but constrained pan and tilt (the latter is constrained to lie in the interval $[\pi/12, \pi/6]$) is looking across from the side of the table, and it is elevated above table top height. The geometric details and numeric constants are not important here. Suffice it to say that the geometric reasoning system deduces that the pan of the camera is irrelevant to the prediction of the length of the ribbon corresponding to the swept surface of the cylinder. It predicts that the length of the ribbon in the image will in fact be

$$-2.42 \times \text{CYL-LENGTH} \times \cos(-\text{TILT})$$

CYLINDER.CAMZ

where 2.42 is the focal ratio of the camera and CYLINDER.CAMZ is an internal quantifier generated by the prediction module. Since cosine is invertible over the range $[\pi/12, \pi/6]$ the expression can be solved for TILT.

Both of the above approaches are used to generate back constraints to ensure coverage of all the relevant quantifiers. They are

$$m_h \geq -2.096 \times \text{CYL-LENGTH} \times (1/\text{CYLINDER.CAMZ})$$

$$m_l \leq -2.338 \times \text{CYL-LENGTH} \times (1/\text{CYLINDER.CAMZ})$$

$$-\text{TILT} \leq \arccos(\sup(-0.413 \times m_h$$

$$\times \text{CYLINDER.CAMZ} \times (1/\text{CYL-LENGTH}))$$

$$-\text{TILT} \geq \arccos(\inf(-0.413 \times m_l$$

$$\times \text{CYLINDER.CAMZ} \times (1/\text{CYL-LENGTH}))$$

The first two are nontrigonometric back constraints and at interpretation time a simple substitution of the measured numeric quantities for m_l and m_h is done. The latter two require further computation at interpretation time. After the substitution, expressions must be bounded over the satisfying set of all the known constraints, and the function arccos applied to give numeric upper and lower bounds on the quantifier TILT.

The technique described here work for a more general class of functions than trigonometric functions (in the current implementation of ACRONYM it is used for functions sin, cos, and arcsin). The requirement is that the domain of the function (e.g., the interval $[-\pi, \pi]$ for sin and cos), can be subdivided into a finite number of intervals over which the function is strictly monotonic, and hence locally invertible.

D. Feature Relation Prediction

Image feature (shape) predictions are organized as the nodes of the *prediction graph*. The arcs of the graph predict image-domain relations between the features. During interpretation correspondences are constructed which match image features and prediction nodes. More global interpretations are derived by taking pairs of such correspondences and trying to instantiate any prediction arcs linking the two prediction nodes. The semantics of the arc types used are now described in detail.

As with shape predictions (Sections II-C1 and 2) many relation predictions involve measurable parameters. For each parameter associated with a relation prediction, both a range of acceptable values and a set of back constraints are computed. When instantiating the prediction of measured parameters can be quickly checked against the value range prediction as a coarse filter to eliminate grossly inconsistent instantiations. The back constraints are then used both to check for more global consistency and to compute what the particular instantiation of the prediction arc implies about the model.

Prediction arcs are generated between pairs of shapes arising in two ways.

1) Prediction arcs are generated to relate multiple shapes predicted for a single cone. For instance a right circular cylinder prediction includes shapes for the swept surface and perhaps each of the end faces (depending on whether the camera geometry is known well enough to determine *a priori* exactly which faces will be visible). It can be predicted that a visible end face will be coincident at least one point in the image with a visible swept surface. (In fact, a stronger prediction can be made: the straight spine of the swept surface image ribbon can be extended through the center of mass of the elliptical image of the end face.)

2) Prediction arcs are also generated between shapes associated with predictions for different generalized cones. These are actually of more importance in arriving at a consistent global interpretation of collections of image features as complex objects.

The semantics of the arc types currently used are as follows.

1) *Exclusive*: If a generalized cone has a straight spine, and during sweeping, the cross section is kept at a constant angle to the spine, then at most one of the cone's end faces can be visible in a single image. *Exclusive* arcs relate image features which are mutually exclusive for this or other reasons. (Note that in this case, instantiations of the two end faces would probably result in inconsistent back constraints being applied to the spatial orientation of the original model, so that eventually the CMS would detect an inconsistency. However, checking for the existence of a simple arc at an early stage is computationally much cheaper than waiting to invoke the decision procedure.)

2) *Collinear*: If two straight line segments in three-space are *collinear* then any two-space image of them will either be a single degenerate point or two collinear line segments. As was pointed out earlier, the spine of the image shape corresponding to the swept surface of a cone is usually a good approximation to the projection of the spine of the cone into the image. Thus, if two cones are known to have collinear spines in three dimensions, a *collinear* spine arc between the prediction of their swept surfaces can be included.

3) *Coincident*: If two cones are physically *coincident* at some point(s) in three-space, then for any camera geometry, if they are both visible then their projections will be coincident at some point(s) (except for some cases of obscuration). Failure to match predicted coincident arcs turns out to be the strongest pruning process during image interpretation.

4) *Angle*: If the *angle* between the spines of two generalized cones as viewed from the modeled camera is invariant over all

the rotational variations in the model, or if an expression for the observed angle can be symbolically computed and is sufficiently simple, then a prediction of the observed angle can be made. For example, wing-wing and wing-fuselage angles are invariant when an aircraft is viewed from above—that is because the only rotational freedom of an aircraft on the ground is about an axis parallel to the direction of view of an overhead camera. Again the fact that the projections of model spines correspond to image spines is used here. This arc type includes (trigonometric) back constraints which make use of the observed angle. Some such constraints constrain relative spatial orientations of generalized cones. Others provide constraints on the orientation of the plane of rotation, which generated the angle, relative to the camera, and hence constraints on an object's orientation relative to the camera.

5) *Approach-Ratio*: Suppose a cone *B* is affixed at one end of its spine to another cone *A*, with a straight spine, somewhere along its length. The spines need not be coincident, but the cones must be. Suppose the spine of cone *A* has endpoints a_1 and a_2 , and let a_3 be the point on the spine of *A* closest to the end of the spine of *B*. Then the *approach-ratio* is the ratio of the length of the spine segment from a_1 to a_3 and the length of the complete spine from a_1 to a_2 . If the spines of *A* and *B* are both observable, then the approach-ratio is invariant under a normal projection for all camera geometries. Thus it is a quasi-invariant for a perspective projection for a camera sufficiently far from the object. For example, the ratio of the distance from the rear of the fuselage to the point of wing attachment, to the length of the fuselage, is almost invariant over all viewing angles for objects sufficiently far from the camera. Again this relies on the correspondences between the projection of a cone spine and the spine of the ribbon generated by the image of its swept surface. Approach-ratios arcs are only generated for pairs of image features which have a coincident arc. They provide back constraints on the model via the symbolic expression which describes the modeled spine approach ratio.

6) *Distance*: Sometimes symbolic expressions for the image distance between two image features can be computed. *Distance* arcs are only generated for pairs of image features which also have an *angle* arc, but no coincident arc. Distance arcs generate back constraints on the original model.

7) *Ribbon-Contains*: This is a directed arc type which two dimensionally relates two predicted ribbons, one of which will contain the other in the image. For instance, *ribbon-contains* arcs are built between the ribbon predicted from the occluding contour of a generalized cone with rectangular cross section, and each of the ribbons generated by the two visible swept faces.

III. SOME IMAGE INTERPRETATIONS

The image interpretations reported here are of a rather preliminary nature. They are based on a low-level descriptive module [6] chosen for its availability rather than its performance and an environment where experimentation has been hampered by address space limitations—the current system occupies two 256K address spaces on a DEC-10. The system

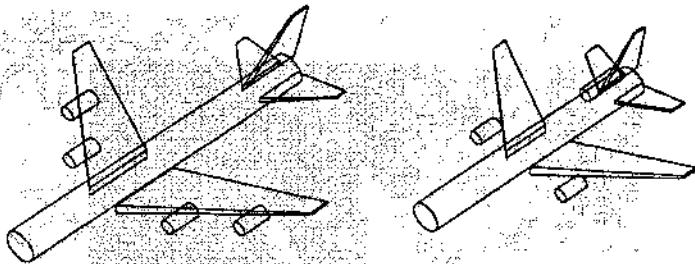


Fig. 1. Instances drawn by ACRONYM of 747's and L-1011's, which are themselves both subclasses of the generic class of wide-bodied passenger jet aircraft.

has been transported to a VAX. Further experimentation is planned on that version by a number of people.

In the examples to be described here ACRONYM was given a generic model of wide-bodied passenger jet aircraft, along with class specializations to L-1011's and Boeing-747's. The Boeing-747 class had further subclass specializations to Boeing-747B and Boeing-747SP. The subclasses do not completely partition their parent classes. The classes are described by sets of constraints on some 30 quantifiers. Fig. 1 shows instances of the two major modeled classes of jet aircraft. These diagrams were drawn by ACRONYM from the models given it to carry out the image interpretations. The constraints for the generic class of wide bodied jets are given in Fig. 2. Units are in meters. The diagrams of Fig. 1 demonstrate the range of variations represented in the generic model. The geometric structure consists of a cylindrical fuselage, two symmetrically placed wings perhaps with rudder, and perhaps a centrally mounted cylindrical rear engine.

The camera was modeled as being between 1000 and 12 000 m above the ground. Thus there is little *a priori* knowledge of the scale of the images. A specific focal ratio was given: 20. (Similar interpretations have been carried out with a variable focal ratio, but then the final constraints on camera height and focal ratio are coupled, and not as clear for illustrative purposes—no accuracy is lost due to the nonlinearities that are introduced into the constraints, although both computation time and garbage collection time are increased.)

The aircraft models, the camera model and the number of pixels in each dimension of the image (512 X 512 in these examples) were the only pieces of world knowledge input to ACRONYM. It has no special knowledge of aerial scenes: all its rules are about geometry and algebraic manipulation. These were applied to the particular generic models it was given to make predictions and then to carry out interpretations.

Figs. 3-5 show three examples of interpretations carried out by ACRONYM. In each case part (a) is a half-tone of the original gray level image. The (b) version is the result of applying the line finder of Nevatia and Babu [9]. That line finder was designed to find linear features such as roads and rivers in aerial photos. Close examination of results on these images indicate many errors, and undue enlargement in width of narrow linear features. It also produces many noise edges in smooth brightness gradients (not visible at the resolution of the reproductions of these figures). These edges are the lowest level input to ACRONYM.

An edge linker [6] is directed by the predictions to look

```

ENG-DISP-GAP ∈ [6, 10]
ENG-DISP ∈ [0, 4]
ENG-GAP ∈ [7, 10]
STAB-ATTACH ∈ [3, 5]
R-ENG-ATTACHMENT ∈ [3, 5]
ENG-OUT ∈ [5, 12]
WING-ATTACHMENT ∈ [20, 40]
    WING-ATTACHMENT ≥ 0.1*FUSELAGE-LENGTH
    WING-ATTACHMENT ≤ 0.6*FUSELAGE-LENGTH
STAB-RATIO ∈ [0.2, 0.55]
STAB-SWEEP-BACK ∈ [3, 7]
STAB-LENGTH ∈ [7.6, 13]
STAB-THICK ∈ [0.7, 1.1]
STAB-WIDTH ∈ [5, 11]
RUDDER-RATIO ∈ [0.3, 0.4]
RUDDER-SWEEP-BACK ∈ [3, 9]
RUDDER-LENGTH ∈ [8.5, 14.2]
RUDDER-X-HEIGHT ∈ [7, 13]
RUDDER-X-WIDTH ∈ [0.7, 1.1]
WING-RATIO ∈ [0.35, 0.45]
WING-THICK ∈ [1.5, 2.5]
WING-WIDTH ∈ [7, 12]
    WING-WIDTH ≤ 0.5*WING-LENGTH
WING-LIFT ∈ [1, 2]
WING-SWEEP-BACK ∈ [13, 18]
WING-LENGTH ∈ [22, 33.5]
    WING-LENGTH ≥ 2*WING-WIDTH
    WING-LENGTH ≥ 0.43*FUSELAGE-LENGTH
    WING-LENGTH ≤ 0.65*FUSELAGE-LENGTH
REAR-ENGINE-LENGTH ∈ [6, 10]
ENGINE-LENGTH ∈ [4, 7]
ENGINE-RADIUS ∈ [1, 1.8]
FUSELAGE-RADIUS ∈ [2.5, 4]
FUSELAGE-LENGTH ∈ [10, 70]
    FUSELAGE-LENGTH ≥ 1.06660000*WING-ATTACHMENT
    FUSELAGE-LENGTH ≥ 1.53816154*WING-LENGTH
    FUSELAGE-LENGTH ≤ 2.5*WING-ATTACHMENT
    FUSELAGE-LENGTH ≤ 2.3255814*WING-LENGTH
R-ENG-QUANT ∈ [0, 1]
    R-ENG-QUANT ≤ 2 + -1*F-ENG-QUANT
F-ENG-QUANT ∈ [1, 2]
    F-ENG-QUANT ≤ 2 + -1*R-ENG-QUANT

```

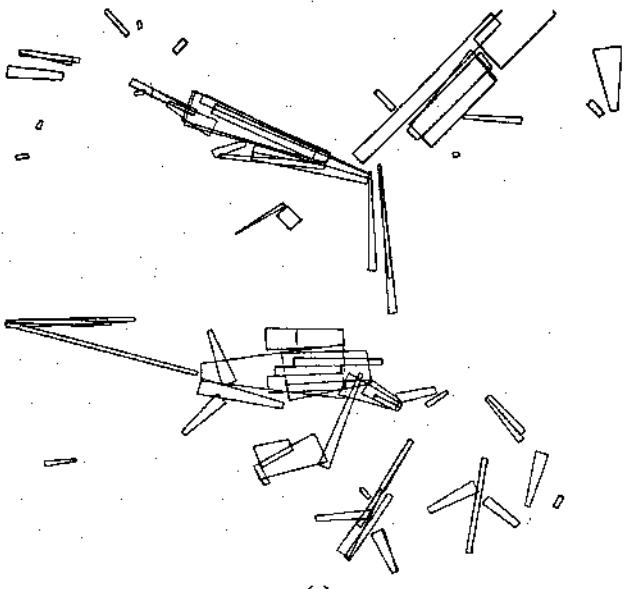
Fig. 2. The constraints implied by the model given to ACRONYM for the generic class of wide-bodied passenger aircraft.

for ribbons and ellipses. In this case there is very little *a priori* information about the scale of the images. The (c) versions of each figure show the ribbons fitted to the linked edges when it is searching for candidate matches for the fuselage and wings of aircraft. There is even further degradation of image information at this stage. These are the only data which the ACRONYM reasoning system is given to interpret. Notice that in the Fig. 5 almost all the shapes corresponding to aircraft are lost. Quite a few aircraft in Fig. 4 are lost also. Besides losing many shapes, the combination of the edge finder and edge linker conspire to give very inaccurate image measurements. It is assumed that all image measurements have a ± 30 percent error, except that for very small measurements, it is assumed that pixel noise swamps even those error estimates. Then the error is estimated to be inversely proportional to the measurement with a 2 pixel measurement admitting a 100 percent error. Thus the data which ACRONYM really gets to work with are considerably more fuzzy than indicated by the (c) series of Figs. 3-5.

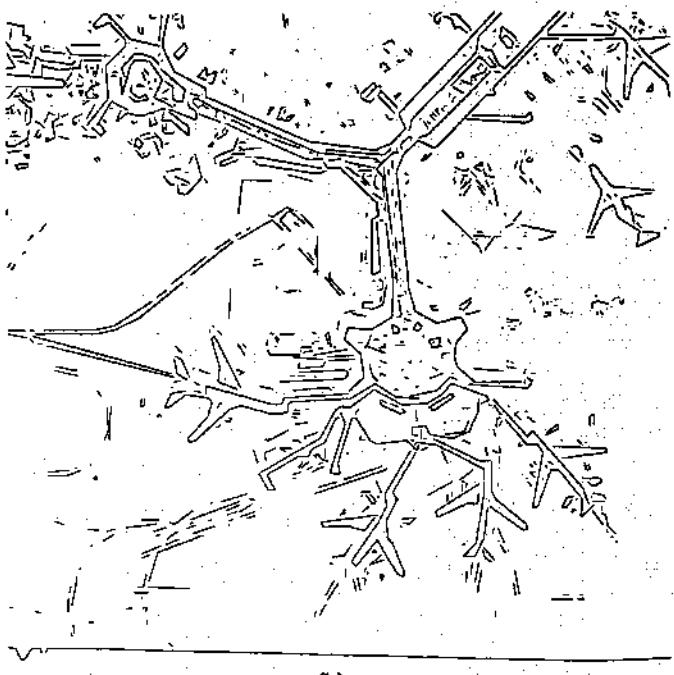
It is intended to make use of new and better low-level de-



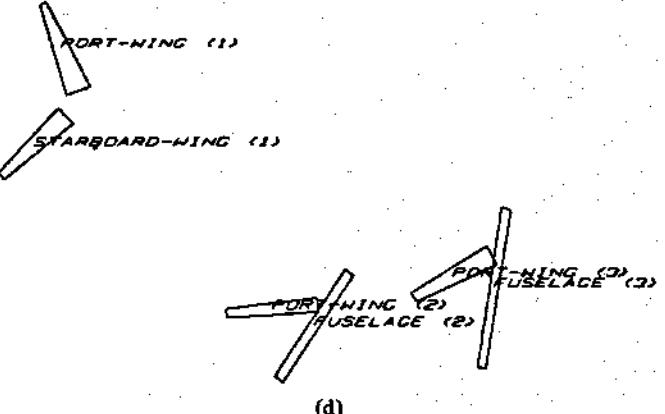
(a)



(b)



(c)



(d)

Fig. 3. Illustrations of some of the computations performed by ACRONYM in interpreting an image. The text contains the details.

scriptive processes being developed by other researchers as soon as they become robust enough for every day use (e.g., Baker [1] whose descriptions from stereo will also include surface and depth information).

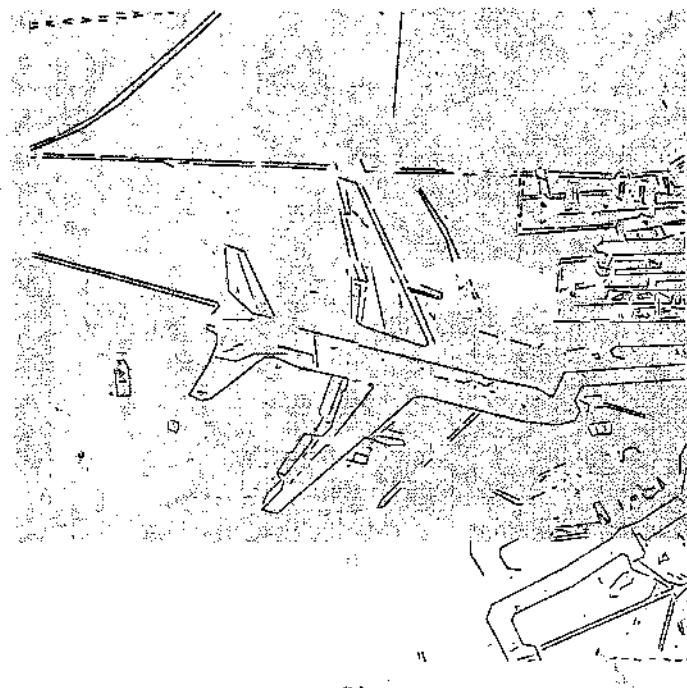
Despite this very noisy descriptive data ACRONYM makes good interpretations of the images. Figs. 3(d), 4(d), and 5(d) show their interpretations with the ribbons labeled by what part of the model they were matched to. (The numbers which may be unreadable in Fig. 3(d) show the groupings into individual aircraft.)

ACRONYM first uses the most general set of constraints, those associated with the generic class of wide-bodied jets, when carrying out initial prediction and interpretation. Interpretation adds additional constraints for each hypothesized aircraft instance. For example, in finding the correspondences in Fig. 4(d) constraints were added which eventually constrained the WING-WIDTH (the width of the wings where they attach to fuselage) to lie in the range [7, 10.5677531] compared to the modeled bounds of [7, 12]. The height of the camera, modeled to lie in the range [1000, 12 000] is constrained by the interpretation to the range [2199, 3322].

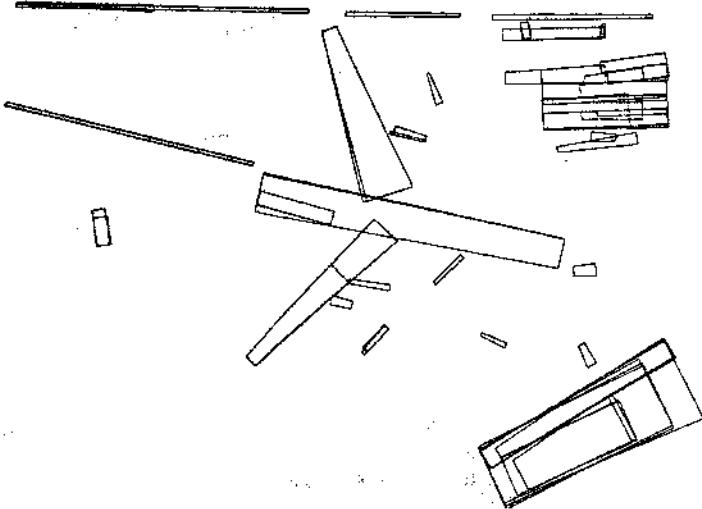
Once a consistent match or partial match to a geometric model has been found in the context of some set of constraints (model class), it is easy to check whether it might also be an instance of a subclass. Only the extra constraints associated with the subclass need be added and checked for consistency with those already implied by the interpretation using the CMS as described in Section II-A. The aircraft located in Fig. 4(d) is consistent with the constraints for an L-1011, but not for a Boeing-747. Examination of the images by the author had previously indicated that the aircraft was an L-1011. The additional symbolic constraints implied by accepting that the aircraft is in fact an L-1011 propagate through the entire constraint set. Although the constraints describing an L-1011 do not include constraints on camera height, the back constraints deduced during interpretation relate quantifiers representing such quantities as length of the wings to the height (and focal ratio in the more general case). Thus the height of the camera is further constrained in Fig. 4(d) to lie in the range [2356, 2489]. Recall that all image measurements were subject to ± 30 percent errors, and that this estimate has taken all such errors into account.



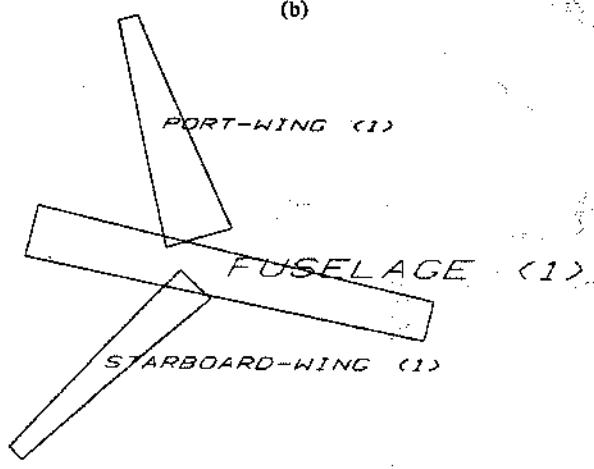
(a)



(b)



(c)



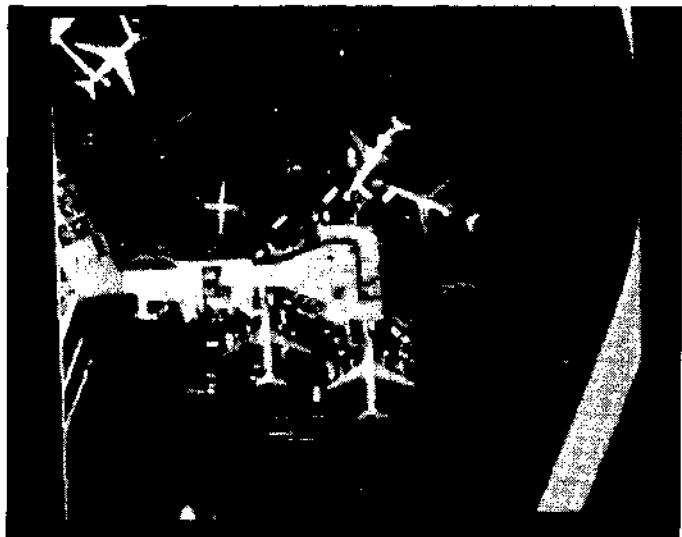
(d)

Fig. 4. Illustrations of some of the computations performed by ACRONYM in interpreting an image. The text contains the details.

Fig. 3(d) indicates matches were found for three airplanes. Examination of the data in Fig. 3(c) indicates that this is the best that could be expected. Note, however, that only partial matches were found in all three cases. For such small ribbons errors were apparently larger than the generous estimate used. The fuselage ribbon in the leftmost aircraft (number 1), for instance, fails to pass the coarse filtering stage. Despite the partial match, this particular aircraft is found to be consistent with the constraints for an L-1011, but not consistent with those of a Boeing-747. Again this is correct.

The other two aircraft identified are even more interesting. The author had thought from casual inspection of the gray level image that they were instances of Boeing-747's. They both gave matches consistent with the class of wide-bodied jets. As expected neither was consistent with the extra constraints of an L-1011. However, although each indi-

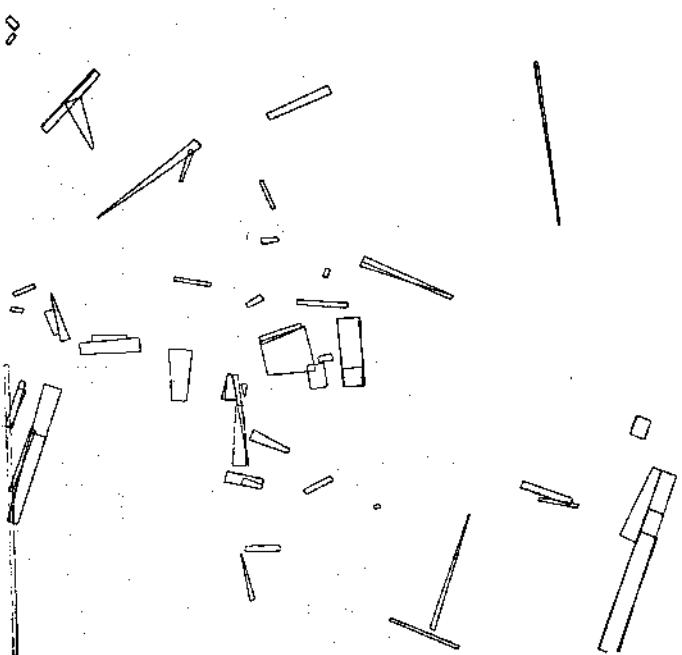
vidual parameter range from the interpretation constraint sets was consistent with the individual parameter value or range for the class of Boeing-747's, neither set of constraints was consistent with that subclass (the constraints contain much finer information than just the parameter ranges—in the same manner as in the example above where constraints on wing length propagate to constrain the camera height). On close examination of the gray level image it was determined that the aircraft were not in fact Boeing-747's. The author used the fact that they were much smaller than the L-1011 to make that deduction, but ACRONYM made the deduction at the local level before considering comparisons between aircraft. It also found the inconsistency at a more global level. The aircraft (probably Boeing-707's) are in fact too small to be wide-bodied jets of any type. Since the scale of the image is unknown *a priori* this cannot be deduced locally. However, it



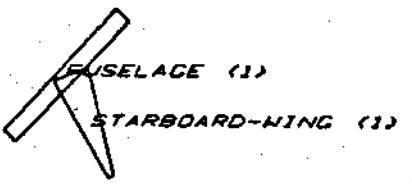
(a)



(b)



(c)



(d)

Fig. 5. Illustrations of some of the computations performed by ACRONYM in interpreting an image. The text contains the details.

is reflected in the height estimates derived at the local level—[5400, 8226] interpreting the L-1011 just as a generic wide-body, ([5786, 6170] as an L-1011), and [9007, 11846] for the rightmost aircraft. Thus ACRONYM deduces that either the left aircraft is a wide-body and the others are not, or the right two are wide-bodies and the left one is not (it is too big).

Finally, note that geometrically there were other candidates for aircraft in the ribbons of Fig. 3(c). For instance, the wing of the aircraft just to the right of those identified and a ribbon found for its passenger ramp could be the two wings of an aircraft with a fuselage missing between them. In fact, these two ribbons were instantiated as an aircraft on the basis of the coarse filters on the nodes and arcs. However, the set of back constraints they generated were mutually inconsistent.

Thus, it can be seen from the examples that even with very poor and noisy data the combined use of geometry and symbolic algebraic constraints can lead to accurate image interpretations. They system should be tested on more accurate low-level data to fully evaluate the power of this approach.

REFERENCES

- [1] H. H. Baker and T. O. Binford, "Edge based stereo correlation," in *Proc. 7th Joint Int. Conf. Artificial Intell.*, Vancouver, Canada, Aug. 1981, pp. 631-636.
- [2] T. O. Binford, "Visual perception by computer," presented at the IEEE Syst., Sci., Cybern. Conf., Miami, FL, invited paper, Dec. 1971.
- [3] —, "Computer integrated assembly systems," in *Proc. NSF Grantees Conf. Industrial Automation*, Cornell Univ., Sept. 1979.

- [4] W. W. Bledsoe, "The sup-inf method in Presburger arithmetic," Dep. Math. and Comput. Sci., Univ. Texas, Austin, Memo. ATP-18, Dec. 1974.
- [5] D. G. Bobrow and T. Winograd, "An overview of KRL: A knowledge representation language," *Cognitive Sci.*, vol. 1, pp. 3-46, 1977.
- [6] R. A. Brooks, "Goal-directed edge linking and ribbon finding," in *Proc. ARPA Image Understanding Workshop*, Menlo Park, Apr. 1979, pp. 72-76.
- [7] —, "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intell. J.*, vol. 17, pp. 285-348, 1981; a longer version is available as Stanford AIM-343, Dep. Comput. Sci., Stanford, CA.
- [8] R. A. Brooks, R. Greiner, and T. O. Binford, "The ACRONYM model-based vision system," in *Proc. IJCAI-6*, Tokyo, Japan, Aug. 1979, pp. 105-113.
- [9] R. Nevatia and K. R. Babu, "Linear feature extraction and description," *Comput. Graphics and Image Processing*, vol. 13, pp. 257-269, 1980.
- [10] R. E. Shostak, "On the sup-inf method for proving Presburger formulas," *J. Ass. Comput. Mach.*, vol. 24, pp. 529-543, 1977.
- [11] B. I. Soroka, "Debugging manipulator programs with a simulator," presented at the Autofact West Conf., Soc. Manuf. Eng., Anaheim, CA, Nov. 1980.



Rodney A. Brooks was born in Adelaide, South Australia, on December 30, 1954. He received the B.Sc. and M.Sc. degrees in mathematics from Flinders University, Adelaide, Australia, in 1974 and 1977, respectively, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1981.

He was a Visiting Scientist in the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, from June through September 1981. Since then he has been a Research Associate in the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge. His research interests span computer vision, robotics, spatial reasoning and planning, and issues in Lisp language and system development.

RODNEY A. BROOKS was born in Adelaide, South Australia, on December 30, 1954. He received the B.Sc. and M.Sc. degrees in mathematics from Flinders University, Adelaide, Australia, in 1974 and 1977, respectively, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1981. He was a Visiting Scientist in the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, from June through September 1981. Since then he has been a Research Associate in the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge. His research interests span computer vision, robotics, spatial reasoning and planning, and issues in Lisp language and system development.

RODNEY A. BROOKS was born in Adelaide, South Australia, on December 30, 1954. He received the B.Sc. and M.Sc. degrees in mathematics from Flinders University, Adelaide, Australia, in 1974 and 1977, respectively, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1981. He was a Visiting Scientist in the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, from June through September 1981. Since then he has been a Research Associate in the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge. His research interests span computer vision, robotics, spatial reasoning and planning, and issues in Lisp language and system development.

RODNEY A. BROOKS was born in Adelaide, South Australia, on December 30, 1954. He received the B.Sc. and M.Sc. degrees in mathematics from Flinders University, Adelaide, Australia, in 1974 and 1977, respectively, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1981. He was a Visiting Scientist in the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, from June through September 1981. Since then he has been a Research Associate in the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge. His research interests span computer vision, robotics, spatial reasoning and planning, and issues in Lisp language and system development.

RODNEY A. BROOKS was born in Adelaide, South Australia, on December 30, 1954. He received the B.Sc. and M.Sc. degrees in mathematics from Flinders University, Adelaide, Australia, in 1974 and 1977, respectively, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1981. He was a Visiting Scientist in the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, from June through September 1981. Since then he has been a Research Associate in the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge. His research interests span computer vision, robotics, spatial reasoning and planning, and issues in Lisp language and system development.

Characteristic views as a basis for three-dimensional object recognition

Indranil Chakravarty,* Herbert Freeman
Rensselaer Polytechnic Institute, Troy, New York 12181

Abstract

This paper describes a new technique for modeling 3D objects that is applicable to recognition tasks in advanced automation. Objects are represented in terms of canonic 2D models which can be used to determine the identity, location and orientation of an unknown object. The reduction in dimensionality is achieved by factoring the space of all possible perspective projections of an object into a set of characteristic views, where each such view defines a characteristic-view domain within which all projections are topologically identical and related by a linear transformation. The characteristic views of an object can then be hierarchically structured for efficient classification. The line-junction labelling constraints are used to match a characteristic view to a given unknown-object projection, and determination of the unknown-object projection-to-characteristic view transformation then provides information about the identity as well as the location and orientation of the object.

Introduction

A problem of current interest is the development of efficient procedures for the computer recognition of three-dimensional objects. This paper describes a modeling scheme which facilitates the recognition of arbitrarily positioned objects on a platform so that a mechanical arm can pick up and hold the object in a known manner. The environment consists of a fixed set of different objects with which the vision system must cope during a task period. For all objects in the set, a computer model is presumed to exist. There are no restrictions on the shape of the objects.

There are three facets to the recognition problem: to establish the identity of the object(s) currently in the field of view, to determine their location, and to determine their orientation. It is instructive to divide the recognition problem into two subproblems—that of object modeling and that of object recognition. The purpose of the research described here was to develop an efficient scheme for solving both these subproblems for objects of general shape. We shall assume that we are dealing with objects for which a pure silhouette-based recognition scheme would be inadequate. Instead moderate-resolution, gray-level images will be required from which the object-describing line structure can be derived. The objects will be assumed to be in a stable position; however, the camera-to-object attitude will be allowed to vary. Furthermore, we will assume that the scene illumination is arranged to be as favorable as possible for the recognition task.

A variety of different schemes have been used in the past for representing the computer model of an object. Although any recognition procedure necessarily requires the existence of a modeling scheme, the converse is not true. As a result many innovative modeling schemes have not been suitable for recognition. Computer models of an object can be classified as belonging to one of the following three classes: (1) volumetric representations based either on the use of generalized cones and cylinders [1,2] or an ordered subdivision of 3D space into homogenous cubes [3], (2) surface descriptions utilizing bicubic patches [4], quadric-surface patches [5,6] or polygonal approximations [7,8], and (3) multiple-view representations [9,10].

In a multiple-view representation a set of two-dimensional projections is used to describe a three-dimensional object. The object's line structure is then characterized in terms of a variety of two-dimensional features. Shape and feature matching techniques, invariant to rotation, translation and scaling are used for identification. This approach is characterized by the work of Chen et al. [11], Holland et al [10], and Perkins [9] and requires stringent positioning of the object with respect to the camera. Other efforts have been directed towards extracting spatial relationships and 3D features from an object's line-drawing projections. In this case the scene interpretation is attempted at a more global level using constraints based on object-level knowledge. This approach has been characterized by Brooks [13], Marr [2], Shapiro et al. [14] and Barrow and Tennenbaum [12].

*Current address: Schlumberger-Doll Research, Old Quarry Road, Ridgefield, CT 06877.

The approach developed in this paper utilizes low-level image operations that are guided by object-level knowledge. This knowledge is embodied in the form of function and line semantics and an hierarchical representation of a 3D object by a class of visible-line projections.

The Characteristic-View Model of an Object

The modeling scheme described here consists of representing a 3D object by means of a relatively large set of visible-line projections. Each projection, called a characteristic-view (CV), is a representation member of one of a finite set of equivalence classes called characteristic-view partitions (CVPs) into which the possible perspective projections of the object can be partitioned. The set of characteristic views is used as a basis for hierarchical object representation. The primary advantage this has over a 3D model is that the line-structure bears a more direct relationship to what is observed by a TV camera. By this we mean that the visibility of the lines and junctions is already predetermined for a given vantage-point domain.

In the following sections we develop (1) the necessary criteria for partitioning the projection set of a 3D object, (2) the geometric relationships that must exist among members of a CVP, and (3) the transformations with which one can determine the identity, orientation and location of a given projection with respect to the stored CV.

Characteristic view partitions

When we examine a projection of a body, such as a picture of a 3D object obtained with a camera, we find that only a portion of the body's edges and vertices are visible. The edges not seen are the so-called "hidden lines" of the object relative to the observer (vantage point). Now if we slowly change the vantage point, we observe that the projected-edge structure will not exhibit any change in topology until we come to a position at which some edges and vertices are no longer visible in the projection, or some new edges and vertices come into view. This leads to the conclusion that for every 3D object the universe of all vantage point locations can be divided into a finite set of vantage-point domains such that for each domain the projected edge structure will exhibit a junction-line identity. We shall use the term junction to refer to the projection of a vertex, and the term line to refer to the projection of an edge. By a junction-line identity we thus mean that for every vantage point in a domain, precisely the same junctions and lines are displayed with the same connectivity relationships, though, the lengths of the lines may differ. An example is given in Figure 1, where a cube with a hole is shown, observed from six different vantage points. Views (a) through (d) belong to the same vantage-point domain and, therefore,

exhibit junction-line identity; views (e) and (f) each belong to different vantage-point domains and do not share junction-line identity with any of the other views. Alternately stated, if C_0 and C_1 are two distinct vantage points in the same domain, then there must exist a continuous path from C_0 to C_1 such that all points along this path project the object into the same characteristic-view partition. It may be possible to generate identical projections from two different vantage-point domains. These, however, according to our definitions would belong to distinct CVPs.

Let us define Q to be the set of all perspective projections of a body on a plane. The set of projections Q can be partitioned into subsets P_i , $i=1, 2, \dots, k$, where each subset is associated with a particular vantage-point domain. We shall call these subsets the characteristic-view partitions (CVPs) of Q . Note that junction-line identity is a necessary but not a sufficient condition for two projections to belong to the same CVP.

Figure 1. Six views of a cube with a hole.

The characterization of bodies in terms of their CVs is useful, the resulting

The approach taken in solving the recognition problem will be to determine whether a given arbitrary projection belongs to one of the characteristic-view partitions of the known set of bodies. Thus we need to show that there exists a characteristic view, among the characteristic-view sets for all bodies, into which the given projection can be transformed.

The stable positions of an object

Although the characterization of bodies in terms of their CVs is useful, the resulting

set of visible-line projections may be large even for a simple object. We will, therefore, develop a more restrictive partitioning scheme for the projection set Q . This will permit not only a reduction of CVs required for representation but will also allow us to compute a linear transformation for determining the orientation and location of the body in 3-space. For two projections to belong to the same partition, they must be junction-line identical and it must be possible to compute this linear transformation.

To reduce the number of vantage-point domains we impose the following restrictions:

1. Objects will be assumed to appear only in stable positions. These are the positions in which the object would come to rest if thrown on a flat, horizontal surface. The surface is called the supporting plane.
2. The camera-to-supporting-plane geometry remains fixed during the recognition process.
3. The camera-to-object distance is limited to a fixed range.

Let us examine the relationship among the camera-to-object distance, the resolution of the sensor, and the number of CVs that need to be represented. Consider the object shown in Figure 2(a) and its plan view shown in Figure 2(b). C_0 and C_1 are two vantage points.

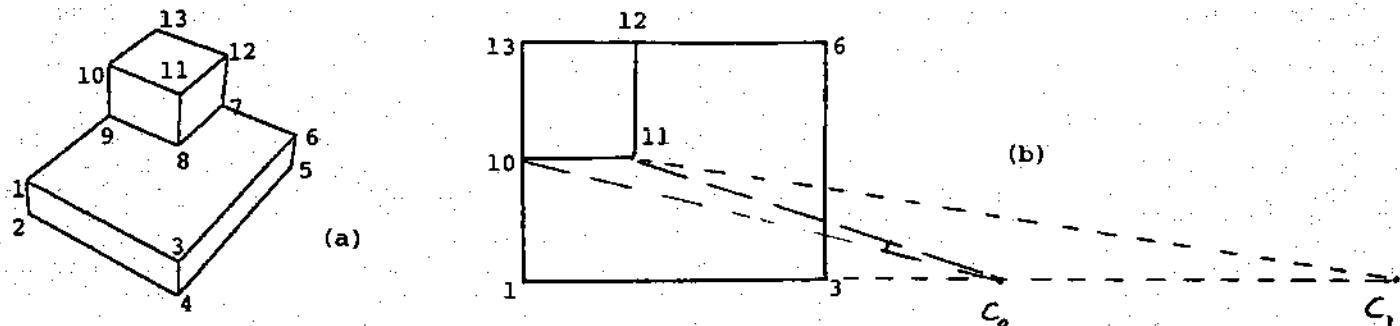


Figure 2. Illustration of maximum camera-to-object distance limitation.

lying in the plane defined by vertices 1, 3, and 4. Clearly from C_0 , the edge defined by vertices 10 and 11 will be visible. As C_0 recedes further away from the object the angular separation between the projection of vertices 10 and 11 will decrease until at infinity it vanishes completely. At some distance between infinity and C_0 the sensor resolution will prevent the edge from being discerned. The maximum camera-to-object distance thus is governed by two factors - the level of detail that one wishes to represent in distinguishing one CV from another and the limitation imposed by the sensor resolution.

One must also consider the case illustrated in Figure 3. Because of the closeness of the vantage point to the face of the polyhedron, a single-face CV is obtained. Clearly such CVs are not desirable in representing a 3D object.

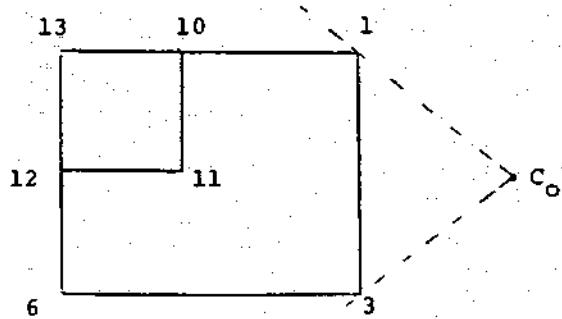


Figure 3. Minimum camera-to-object distance limitation.

Note also that the stable positions of a body provide a non-disjoint partitioning of Q ; that is, different stable positions of an object may yield perspective projections that are members of the same CV. This is illustrated using the polyhedron with 14 vertices and 9 faces shown in Figure 2(a). The first CV is generated where the polyhedron is in a stable position with the face specified by vertices 2, 4, 5, 14 on the supporting plane as shown in Figure 4(a). The same CV, however can be generated with the polyhedron in another stable position with the face specified by vertices 1, 2, 14, 13, 10, 9 on the supporting plane as illustrated in Figure 4(b).

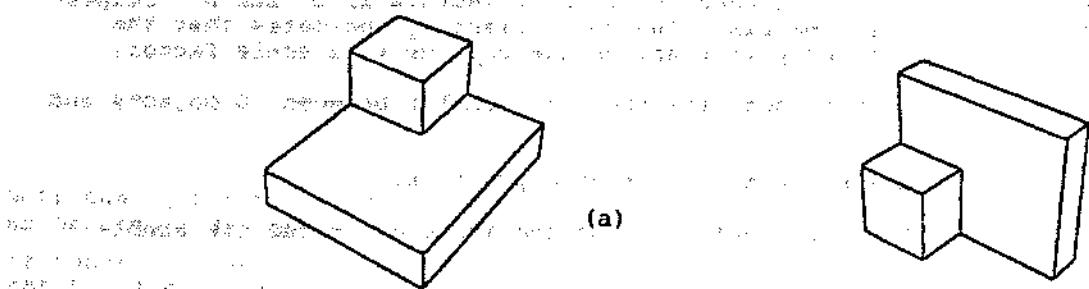


Figure 4. Identical CVs generated from two different stable positions.

Projection-to-CV transformation

Four coordinate systems are used to compute the projection-to-CV transformation, as shown in Figure 5:

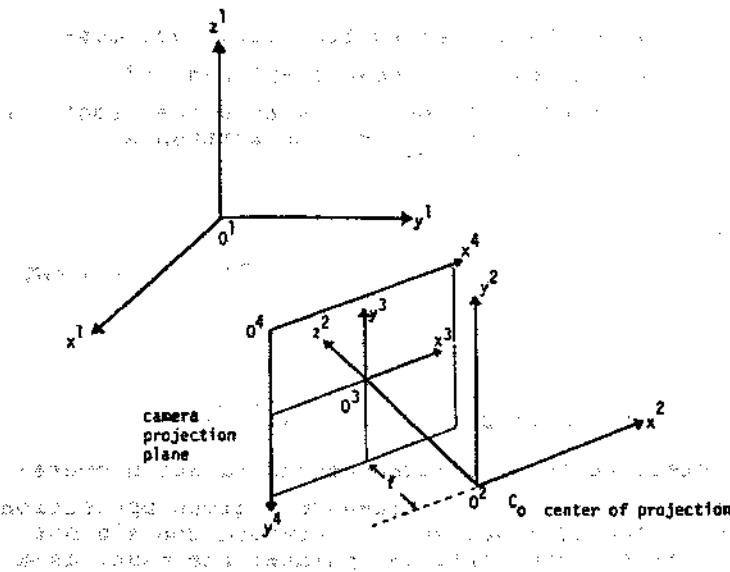


Figure 5. Camera geometry

The projection and CV transformation can be expressed as follows. A junction is a point where two edges meet. A face is a portion of a surface bounded by edges or closed on itself. A region is the projection of a connected visible part of a face. We define the picture-taking process as a projective mapping of coordinate vectors from 0^1 to 0^4 . The use of homogeneous coordinates makes this a linear transformation. The projection transformation is a function of the camera parameters, that is, the pan, tilt and swing angles, the location of the camera lens in 0^1 , and focal length of the lens. The transformation can be expressed as a single 4×3 matrix. Our objective is to determine a procedure for calculating the coefficients of this matrix. Once this transformation is known, the camera is calibrated with respect to the global coordinate system. This transformation is then used to generate the CVs from the 3D model. Notice that projection of the 3D model using the camera transformation matrix is the same as imaging the object except that the hidden lines/surfaces are not removed.

The projection transformation can be expressed as follows:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = [T_C] \begin{bmatrix} x^1_i \\ y^1_i \\ z^1_i \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = [T_C] \begin{bmatrix} x^1_i \\ y^1_i \\ z^1_i \\ 1 \end{bmatrix} \quad i=1,2,3,\dots,n$$

where $(x^4, y^4, 1)$ and $(x^1, y^1, z^1, 1)$ are homogeneous coordinate vectors in O^4 and O^1 , respectively, and $[T]$ is the transformation matrix. The coefficient k_i indicates that the homogeneous coordinates of the projected points are unique only up to a scale factor.

We will use the following notation to describe the relationship between 3D objects and corresponding CVs:

1. B_{ij} is the i th body in the scene, in the j th stable position.
2. CV_{ijk} is the k th characteristic-view partition of the i th body in the j th stable position.
3. Any perspective projection (view) of a body B_{ij} is denoted by V_{ij} . Since V_{ij} must be a member of some characteristic-view partition, we write the equivalence as $V_{ij} \in CV_{ijk}$.

Let us assume that we obtain an image of an object and, based upon the describing line-structure, conclude that $V_{ij} \in CV_{ijk}$. Since the camera transformation matrix is known, we also know that

$$CV_{ijk}(x_p^4, y_p^4, 1) = [T_C] B_{ij}(x_p^1, y_p^1, z_p^1, 1) \quad p=1, 2, \dots, n$$

This transformation maps the visible points on B_{ij} into the projected line structure described by CV_{ijk} . Since $V_{ij} \in CV_{ijk}$ implies that B_{ij} could only have been displaced and rotated on the supporting plane (since we have constrained it to be in a stable position) with respect to the camera, the total transformation from B_{ij} to V_{ij} can be written as follows:

$$V_{ij}(x_p^4, y_p^4, 1) = [T_C] [T_R] [T_D] B_{ij}(x_p^1, y_p^1, z_p^1, 1)$$

where

$[T_C]$ is the camera transformation matrix

$[T_R]$ is the rotation matrix about the z^1 -axis

$[T_D]$ is the displacement matrix on the x^1-y^1 (supporting) plane

Since $[T_C]$, B_{ij} and V_{ij} are all known, we can determine the displacement and rotation operator. This requires that we be able to match at least two points between the given projection and the stored CV whose 3D coordinates are known. The CV model must, therefore contain not only the 2D image coordinates of junctions (or some selected critical points) but must, as a minimum, contain the 3D coordinates of at least two such points.

We summarize briefly what the above transformation accomplishes for us in the context of recognition. Each CV is a mapping of the visible lines and junctions of a 3D body. The junctions are specified in image coordinates (2D) and the visible vertices in global coordinates (3D). The mapping is given by the calibration matrix $[T]$. Let us assume that we are given an arbitrary projection of the body and wish to test whether it is a member of a particular CVP. If we can match two points between the projection and the CV and, assuming that the 3D coordinates of these points are known, then we can determine a transformation which maps the 3D coordinates of the model (from which the CV was obtained) into this arbitrary projection.

Classification and generation of CVs

We shall develop in this section a scheme for classifying the CVs of an object. The classification is used both for storing the CVs of an object in an hierarchical manner and for narrowing down the possible choices for matching. It is evident that some CVs in certain specified stable positions convey more information than others. It is desirable that this be reflected in the classification hierarchy.

Since one criterion used for partitioning the projection set Q is the junction-line identity property, it is natural to develop a classification scheme based on the number and type of label associated with each junction. The labels used are the generalized line and junction labels which are applicable to both planar and curved surface objects [15]. A hierarchy of junctions can be established by noting that non-occluding lines forming a junction convey more information about the object than occluding lines. The eight generalized junction types developed in [15] can be ordered as follows:

Junction Type	Number of Regions per Line		
	Line 1	Line 2	Line 3
Y	2	2	2
W,S	1	2	1
V,C	1	2	
V,A	1	1	
T	{1}	2	1
	1	1	1

Note that junction types S,C and A are associated only with curved objects. We now define an ordered label set $L = \{M_Y, M_{WS}, M_{VC}, M_{VA}, M_T\}$ where M_X refers to the number of occurrences of junction type X in a CV. Characteristic views are now classified in terms of the ordered set L. Two CVs are placed in the same class if they have the same number for the most significant junction type. Thus $L = \{3,2,2,1,1\}$ is in the same class as $L = \{3,2,1,1,1\}$ even though the total number of junctions and lines in the CVs are different.

This type of classification permits a hierarchical, canonic representation of a 3D object in terms of a set of 2D perspective projections [16]. We illustrate this with the polyhedron of Figure 2a. This object has 10 CV classes. Of these, only the first one does not contain any T junctions. Listed in order of increasing degeneracy, the classes are as follows:

- (3,-,-,-,-)
- (2,-,-,-,-)
- (1,-,-,-,-)
- (0,5,-,-,-)
- (0,4,-,-,-)
- (0,3,-,-,-)
- (0,2,-,-,-)
- (0,0,0,7,-)
- (0,0,0,6,-)
- (0,0,0,4,-)

The CVs corresponding to the first class are illustrated in Figure 6. One member from each

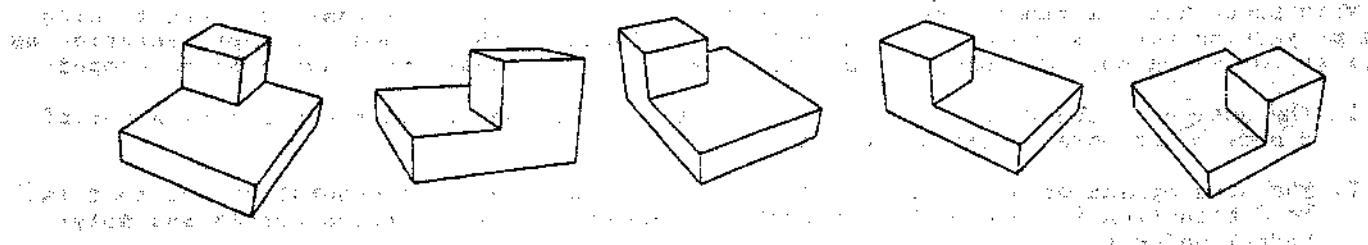


Figure 6. CVs of ordered label set $L = \{3, -, -, -, -\}$

of the remaining nine classes is shown in Figure 7 (a-i). For the purpose of representation, it is necessary to use only the non-degenerate CVs corresponding to each stable position. Although we have not placed any importance on the hierarchy of stable positions, this can be incorporated into the above structure if an object is known to appear more often in one stable position than another. Lastly, one must state that the exact number of CVs of an object that needs to be used for representation depends not only upon the object itself but also on the other 3D objects from which it must be distinguished.

Experimental results

An overall summary of the recognition scheme used is shown in Figure 8. We shall illustrate the intermediate processing steps used in the recognition scheme by a sequence of images which depict the processing of an actual object.

The digitized image of a bracket on a table top is shown in Figure 9(a). The object was digitized under normal room lighting at a 256x256 pixel resolution. The 3D model of this object was obtained by measurements made on the object and the surfaces were approximated by planes. An image of the resulting model with the vertices labeled is shown in Figure 9(b). The binary image resulting from applying an edge operator is shown in Figure 9(c). The binary image is then chain-encoded into a representation where the lines and junctions can be explicitly accessed for further processing [17]. The encoded image is shown in Figure 9(d).

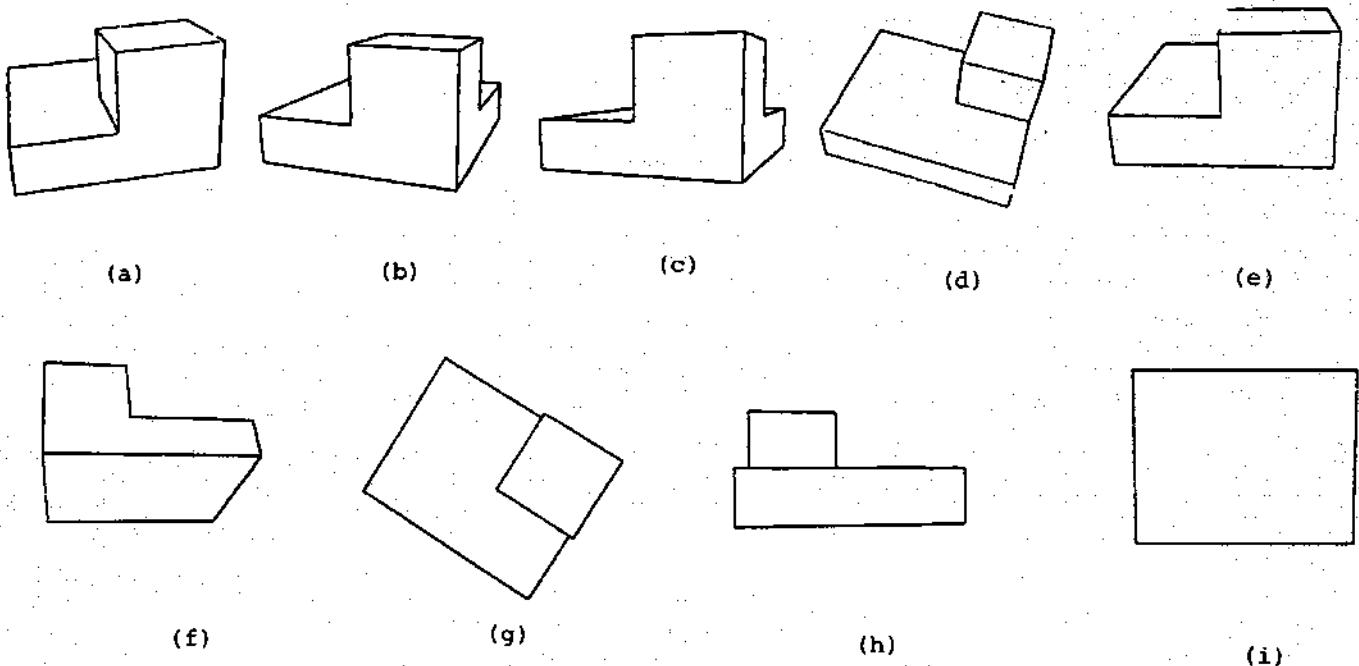


Figure 7. Representative CVs for polyhedron.

The next image, Figure 9(e) shows the line structure obtained after deleting dangling lines from junctions and isolated segments. The silhouette obtained after labeling is shown in Figure 9(f). The junctions used for correlating the line structure to the stored CV is shown in Figure 9(g). The V marks the W junctions used for determining the projection-to-CV transformation. The CV matched is shown in Figure 9(h). The computed projection from the 3D model that corresponds to the unknown object projection is shown in Figure 9(i).

Summary

This paper has described a new technique for modeling 3D objects based on partitioning the projection set. We have illustrated the usefulness of the method for representation as well as for matching. We summarize briefly the three key ideas that have been developed:

1. The notion of CVs which provide a systematic way to partition the projection set of a body based on associated label sets.
2. The development of a uniform approach in handling objects of general shape; that is, no distinction is made either in partitioning or labeling between curved and polyhedral objects.
3. The development of a projection-to-CV transformation which is used both for recognition and for verifying the reliability of the match.

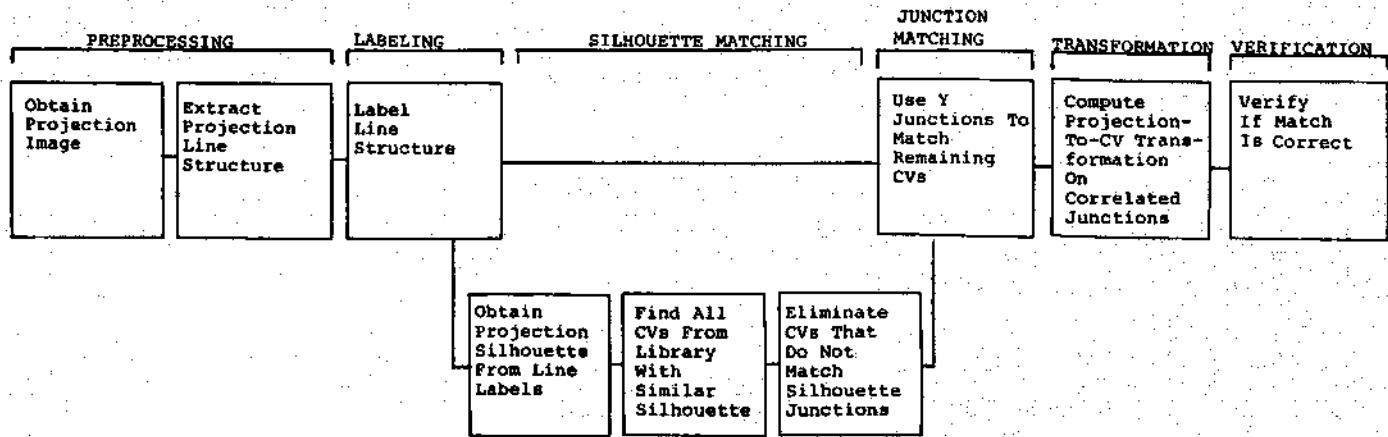


Figure 8. The recognition scheme.

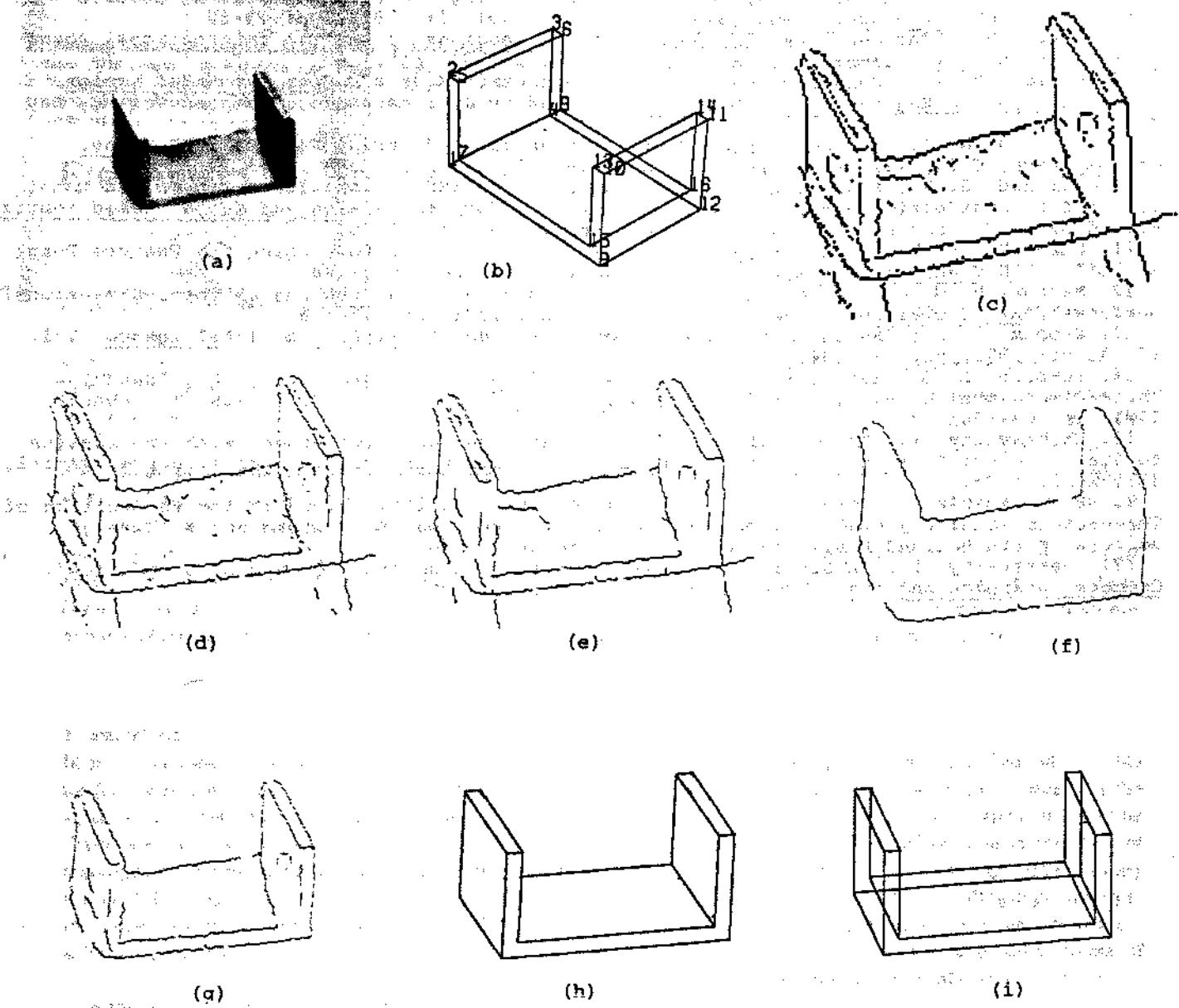


Figure 9. Steps in the processing of the image of a simple bracket.

Acknowledgment

The research reported here was supported by the National Science Foundation, Automation, Bioengineering and Sensing Systems Program, under grant ENG-7904821. This support is gratefully acknowledged.

References

1. Agin, G. J., "Hierarchical Representations of 3D Objects", Final Report SRI, SRI Project 1187, Stanford Research Institute, Stanford, CA, March 1977.
2. Marr, D. and Nishihara, H. K., "Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes", MIT A.I. Laboratory Memo 377, August 1976.
3. Srihari, S. N., "Representation of Three-Dimensional Digital Images", ACM Computing Surveys, Vol. 13, (4), December 1981.
4. Potmesil, M., "Generation of 3D Surfaces from Images of Pattern Illuminated Images", IEEE Computer Society Conference on PRIP, Chicago, Ill, July 1979.
5. Dane, C. and Bajcsy, R., "Three-Dimensional Segmentation Using the Gaussian Image and Spatial Information", IEEE Computer Society Conference on PRIP, Dallas, TX, August 1981.

6. Levin, J. Z., "Mathematical Models for Determining the Intersections of Quadric Surfaces", Computer Graphics and Image Processing, Vol. 11, 1979, pp. 73-87.
7. Baker, H., "Three-Dimensional Modeling", Proceedings of the 5th International Joint Conference on A. I., Cambridge, MA, August 1977, pp. 649-655.
8. Wesley, M. A., "Construction and Use of Geometric Models", Computer Aided Design Modeling, Systems Engineering, CAD Systems, edited by J. Encarnacao, Springer Verlag, New York, 1980.
9. Perkins, W. A., "A Model-Based Vision System for Industrial Parts", IEEE Trans. on Computers, C-27, (2), February 1978, pp. 126-143.
10. Holland, S. W., Rossol, L. and Ward, W. R., "CONSIGHT-1: A Vision Controlled Robot System for Transferring Parts from Belt Conveyors", Computer Vision and Sensor Based Robots, edited by G. C. Dodd and L. Rossol, Plenum Press, New York, 1979.
11. Chen, N., Birk, J. and Kelly, R., "Estimating Workpiece Pose Using the Feature Point Method", IEEE Trans. on Automatic Control, AC-25, (6), December 1980.
12. Barrow, H. G. and Tennenbaum, J. M., "Interpreting Line Drawings as Three-Dimensional Surfaces", Artificial Intelligence, Vol. 17, August 1981, pp. 75-116.
13. Brooks, R. A., "Symbolic Reasoning Among 3-D Models", Artificial Intelligence, Vol. 17, August 1981, pp. 285-348.
14. Shapiro, L. G., Haralick, R. M., Moriarty, J. D. and Mulgaonkar, P. G., "Matching Three-Dimensional Models", IEEE Computer Society Conference on PRIP, Dallas, TX, August 1981, pp. 534-541.
15. Chakravarty, I., "A Generalized Line and Junction Labelling Scheme with Application to Scene Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1, (2), April 1979.
16. H. Freeman and I. Chakravarty, "The Use of Characteristic Views in the Recognition of Three-Dimensional Objects", Pattern Recognition in Practice, E. Gelsema and L. Kanal, editors, North-Holland Publishing Company, Amsterdam, 1980.
17. Chakravarty, I., "A Single Pass Chain Generating Algorithm for Region Boundaries", Computer Graphics and Image Processing, (15), February 1981.

"Fleshing Out Projections" by M.A. Wesley and G. Markowsky from *IBM Journal of Research and Development*, Volume 25, Number 6, November 1981, pages 934-954. Copyright 1981 by International Business Machines Corporation, reprinted with permission.

M. A. Wesley

G. Markowsky*

Fleshing Out Projections

In an earlier paper, the authors presented an algorithm for finding all polyhedral solid objects with a given set of vertices and straight line edges (its wire frame). This paper extends the Wire Frame algorithm to find all solid polyhedral objects with a given set of two dimensional projections. These projections may contain depth information in the form of dashed and solid lines, may represent cross sections, and may be overall or detail views. The choice of labeling conventions in the projections determines the difficulty of the problem. It is shown that with certain conventions and projections the problem of fleshing out projections essentially reduces to the problem of fleshing out wire frames. Even if no labeling is used, the Projections algorithm presented here finds all solutions even though it is possible to construct simple examples with a very large number of solutions. Such examples have a large amount of symmetry and various accidental coincidences which typically do not occur in objects of practical interest. Because of its generality, the algorithm can handle pathological cases if they arise. This Projections algorithm, which has applications in the conversion of engineering drawings in a Computer Aided Design, Computer Aided Manufacturing (CAD/CAM) system, has been implemented. The algorithm has successfully found solutions to problems that are rather complex in terms of either the number of possible solutions or the inherent complexity of projections of objects of engineering interest.

1. Introduction

In an earlier paper [1] the authors presented an algorithm for finding all polyhedral solid objects with a given set of vertices and straight line edges (its wire frame). The Wire Frame algorithm was based on the concepts of algebraic topology and rigorous definitions of the geometric entities involved. It recognized that many solid objects may have the same wire frame and was able to find all possible solutions efficiently.

In this paper we extend the Wire Frame algorithm to polyhedral objects described by a set of two dimensional projections such as might be seen on an engineering drawing. The projection process may introduce another level of ambiguity into reconstruction problems and increases the possibility of there being many objects with the same set of projections. The Projections algorithm presented here can work with very little information, for

example, only two projections, and find all possible objects matching the data. However, it is seen that the number of solutions may be very large and that it may be reasonable to provide more information in the form of three or more projections, by labeling corresponding features in divers views, and by providing depth information. The Projections algorithm is able to make use of this extra information and can also accept other forms of advice, such as whether given points are inside material.

Quite apart from its mathematical interest, the algorithm has practical applications in the automatic conversion of digitized engineering drawings into solid volumetric representations of the geometry of objects. These solid volumetric representations become the basis for the simulation and synthesis of large parts of the design validation, analysis, manufacture, inspection, and documentation process [2, 3].

The subject of reconstruction of solid polyhedral objects from their projections has been studied over a period

*Consistent use of alphabetical ordering of authors' names tends to slight people whose names begin with letters towards the end of the alphabet. Thus, the order of names on this paper is not meant to pass judgement on the relative contributions of the authors, but rather to illustrate the fact that names appearing in alphabetical order is not a "natural law."

Copyright 1981 by International Business Machines Corporation. Copying is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract may be used without further permission in computer-based and other information-service systems. Permission to republish other excerpts should be obtained from the Editor.

of years. Early work [4-6] was largely based on labeling corresponding information in different views and requiring the user to conform to constraints on the manner of description of features such as faces. The historical trend has been to free the user of as many constraints as possible [7]. However, the relaxation of constraints has led to the possibility of multiple solutions to a given problem, and workers have tended to concentrate on heuristic approaches to find a probable solution. A recent paper [8] reports such a heuristic approach that allows complete freedom of input and has been implemented; another paper [9] outlines an approach that would allow certain views of cylindrical surfaces but does not include an implementation. None of this work appears to be based on formal geometric definitions and the concepts of algebraic topology. A closely related development path has been followed by workers in the fields of Computer Vision and Scene Analysis. This path has been based on vertex and edge configurations in a single view [10-12] and has generally been restricted to objects with trihedral vertices and views with no chance alignments; this approach has led to the Origami World [13] and a linear programming approach [14].

This paper presents a very general and complete approach based on the authors' previously published Wire Frame algorithm. In addressing the problem of constructing a solid object from a number of two dimensional views, it is shown that, on the one hand, complete labeling of edges and vertices leads to the previously published Wire Frame algorithm. On the other hand, the Projections algorithm described here is capable of working with no further information than the lines and points of the two dimensional projections and is able to enumerate all possible solutions to a given set of projections, with a cost commensurate with the number of solutions. The techniques presented are applicable when two or more projections are available. Of course, the one projection case has, in general, infinitely many solutions and is not discussed further in this paper. The chief advantage gained from providing more projections is quite naturally to reduce the number of possible ambiguities.

The Projections algorithm constructs polyhedral objects from projections containing only straight lines. The logical component of this algorithm is topological in nature and is, in principle, independent of whether the components are linear or nonlinear. While extension to objects with curved surfaces and projections with curved lines appears to be feasible, the ease of actually carrying out such an extension would depend greatly on the family of allowable curves and surfaces, as well as the projection conventions used.

The paper is organized as follows: Section 2 reviews the definitions of objects, faces, edges, and vertices used in the paper describing the Wire Frame algorithm [1] and then develops the basic results dealing with back projections and labeled projections. Section 3 outlines the original Wire Frame algorithm and describes the Basic Projections algorithm which handles the general case of unlabeled projections of wire frames of objects. Section 4 presents some extensions to the Basic Projections algorithm which enable it to make use of more general forms of input data. For example, various types of views (overall, detail, and cross section) and depth information distinguishing between visible and occulted lines are considered. In Section 5, some examples are given to clarify this discussion. These examples illustrate the execution of the algorithm in both the stylized world of geometric puzzles with multiple solutions and the practical world of engineering drawings. The engineering objects successfully constructed from their projections are sufficiently complicated that a human unfamiliar with the solid object generally has some difficulty envisioning it. Thus, the algorithm appears capable of handling real world problems.

2. Basic concepts and results

The basic concepts defined in this section are based on some fundamental topological ideas which are described in detail in [15]. Throughout the paper the standard topology in \mathbb{R}^3 and the induced topology on subsets of \mathbb{R}^3 are assumed. Vertices refer to points in \mathbb{R}^3 and edges refer to line segments defined by two points in \mathbb{R}^3 . The approach used in this section is to define faces, objects, wire frames, and projections, and then describe the consequences of these definitions.

Definition 1

A face, f , is the closure of a nonempty, bounded, connected, coplanar, open (in the relative topology) subset of \mathbb{R}^3 whose boundary (denoted by ∂f) is the union of a finite number of line segments. P_f is used to denote the unique plane which contains f . \square

Definition 2

An object, O , is the closure of a nonempty, bounded, open subset of \mathbb{R}^3 whose boundary (denoted by ∂O) is the union of a finite number of faces. \square

From the definitions above it is easy to see that the "cube," $\{x, y, z \in \mathbb{R}^3 | 0 \leq x \leq 1, 0 \leq y \leq 1, 0 \leq z \leq 1\}$ is an object and that $\{(1, y, z) \in \mathbb{R}^3 | 0 \leq y \leq 1, 0 \leq z \leq 1\}$ is one of its "square" faces. Starting off with open sets means that faces and objects have nontrivial interiors. Notice that it is not assumed that an object is the closure of a connected set. This allows objects that consist of disjoint "solids" or even objects which intersect only in

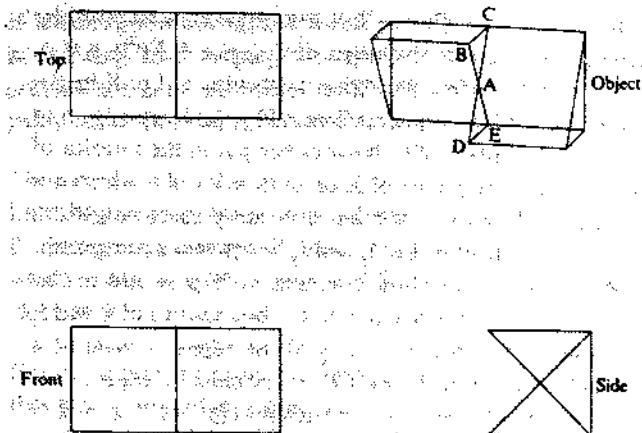


Figure 1. Examples of projections.

edges, etc. One can argue that this last case does not represent a "real" object, but in practice all sorts of strange objects can appear. Thus, we decided to handle the most general case possible. Furthermore, this generality does not exact any penalty other than creating a large number of solutions.

Another point worth noticing is that Definitions 1 and 2 allow many different representations of the boundaries of faces and objects by line segments and faces (respectively). However, there are canonical representations of the boundaries which correspond to one's intuitive notions about such things. To get to these representations it is necessary to introduce several additional concepts.

Definition 3.

- Let f be a face. The *vertices* of f , $V(f)$, are defined to be the set of all points for which two noncolinear line segments contained in ∂f can be found whose intersection is the given point.
- Let f be a face. The *edges* of f , $E(f)$, are defined to be the set of all line segments e , contained in ∂f , satisfying the following conditions:

- The endpoints of e belong to $V(f)$;
 - No interior point of e belongs to $V(f)$.
- Let O be an object. The *vertices* of O , $V(O)$, are defined to be the set of all points p for which faces $f_1, f_2, f_3 \subseteq \partial O$ can be found such that $\{p\} = f_1 \cap f_2 \cap f_3 = P_{f_1} \cap P_{f_2} \cap P_{f_3}$.
 - Let O be an object. The *edges* of O , $E(O)$, are defined to be the set of all line segments e , contained in ∂O , satisfying the following conditions:

- The endpoints of e belong to $V(O)$;
- No interior point of e belongs to $V(O)$;
- For every point p of e , two noncoplanar faces can be found, $f_1, f_2 \subseteq \partial O$ such that $p \in f_1 \cap f_2$.

(e) Let O be an object. The *wire frame* of O , $WF(O)$, is defined to be the ordered pair $(V(O), E(O))$. \square

It can be shown that the edges of an object can intersect only at vertices of the object, i.e., at their endpoints.

The Wire Frame algorithm detailed in [1] allows one to construct all possible objects which have a given wire frame. It happens to be true, but not immediately obvious from the definitions, that $V(f)$, $E(f)$, $V(O)$, and $E(O)$ are all finite and well-defined. These facts and others are discussed in greater detail in [1].

The Wire Frame algorithm described in [1] runs on any collection of points and line segments in \mathbb{R}^3 and either returns all objects having the given collection as their wire frame or shows that the given collection could not be a valid wire frame. In presenting the Projections algorithm the first things to consider are the projections of the wire frame of a valid object. At this point it is necessary to make clear exactly what is meant by a projection.

Definition 4.

Let O be an object, $P \subset \mathbb{R}^3$ a plane, and $\pi_P: \mathbb{R}^3 \rightarrow P$ the perpendicular projection. By the *P-projection* of O , denoted by $O \uparrow P$, is meant the ordered pair, $(V(O \uparrow P), E(O \uparrow P))$, of *P-vertices* and *P-edges* of O defined by the following process. Let E^* be the set of images under π_P of all edges of O which are not perpendicular to P . Then the *P-vertices* of O are those points of P which lie on at least two noncolinear line segments in E^* . The *P-edges* of O are those line segments of P which have elements of $V(O \uparrow P)$ as endpoints, have no points of $V(O \uparrow P)$ as interior points, and are subsets of unions of elements of E^* .

XY , YZ , and ZX are used to denote the planes $Z = 0$, $X = 0$, and $Y = 0$, respectively. \square

Figure 1 shows some of the things that can happen as a result of projection. The vertex A disappears in the front and top views. Furthermore, the edges AB , AC , AD , and AE do not appear as such in these views. Rather a single edge appears which is the union of the projections of the four aforementioned line segments. However, in the side view the vertex A projects into a vertex, and the projections of AB , AC , AD , and AE form distinct line segments.

At this point it seems appropriate to discuss the situations in which vertices of an object project into vertices in

a given projection. Note that if a vertex of a polyhedral object is the intersection of at least three noncoplanar line segments, the image of that vertex under any projection is the intersection of at least two noncolinear line segments and is thus a vertex in that projection. For convenience, vertices which are the intersections of at least three noncoplanar line segments are called *Class I vertices*. Thus, if two different projections of an object are given, the Class I vertices are a subset of the set of all intersections of all the perpendiculars erected at the vertices in each projection.

All vertices of an object which are not Class I are to be called *Class II vertices*. In Fig. 1, vertex A is Class II; all other vertices are Class I. In general, one cannot expect to recover Class II vertices simply by erecting perpendiculars and computing their intersections.

There are a number of properties of Class I and Class II vertices which are useful in recovering an object from its projections. The key observation, which is formalized below, is that the wire frame of \mathcal{O} can be recovered from the Class I vertices of \mathcal{O} and certain line segments joining these vertices.

Definition 5

The *skeleton*, $S(\mathcal{O})$, of an object \mathcal{O} is the ordered pair $(SV(\mathcal{O}), SE(\mathcal{O}))$ of *skeletal vertices* and *skeletal edges* where $SV(\mathcal{O})$ is the set of the Class I vertices of \mathcal{O} and $SE(\mathcal{O})$ is a set of line segments joining the elements of $SV(\mathcal{O})$. For $v_1, v_2 \in SV(\mathcal{O})$, there exists $w \in SE(\mathcal{O})$, joining v_1, v_2 iff there exists an edge or colinear sequence of edges of \mathcal{O} joining v_1 and v_2 and not containing any other Class I vertex. \square

Theorem 6

Let \mathcal{O} be an object. Then the wire frame of \mathcal{O} , $(V(\mathcal{O}), E(\mathcal{O}))$, can be recovered from the skeleton of \mathcal{O} , $(SV(\mathcal{O}), SE(\mathcal{O}))$, as follows. First, $V(\mathcal{O}) = V^*(\mathcal{O})$ where

$$V^*(\mathcal{O}) = SV(\mathcal{O}) \cup \{v | \{v\} = e_1 \cap e_2, e_1, e_2 \in SE(\mathcal{O})\}.$$

Thus, to get all vertices of \mathcal{O} it is enough to add all intersection points of skeletal edges to the skeletal vertices. Second, $E(\mathcal{O})$ is simply the set of line segments which result from partitioning the skeletal edges using their points of intersection.

Proof Observe that from Definition 5 it follows that every skeletal edge is the union of edges of \mathcal{O} . Thus, the intersection of two skeletal edges is a point of intersection of two edges. However, edges of \mathcal{O} intersect only in vertices of \mathcal{O} . Thus, $V^*(\mathcal{O}) \subseteq V(\mathcal{O})$.

It remains to show that $V(\mathcal{O}) \subseteq V^*(\mathcal{O})$. In particular it must only be demonstrated that every Class II vertex of \mathcal{O}

belongs to $V^*(\mathcal{O})$. To see this it is necessary to consider briefly the nature of the edges of \mathcal{O} . Let $e \in E(\mathcal{O})$, $p \in e$, and ℓ the infinite line through p containing e . Let X be the set of disjoint line segments formed by the intersection of ℓ and the boundary of \mathcal{O} . Now either p is in the interior of X (*i.e.*, there are points of X on both sides of p which are arbitrarily close to p) or p has arbitrarily close neighbors only to one side of it. Let f_1 and f_2 be the two noncoplanar faces whose intersection contains e . If p is not in the interior of X , then, since p is on the boundaries of f_1 and f_2 but is not in the interior of any of the edges, it must be a vertex of each of the faces, *i.e.*, there must be edges, $e_1 \in f_1, e_2 \in f_2$, not colinear with e such that $\{p\} = e \cap e_1 = e \cap e_2$. But in this case there are three noncoplanar edges through p , namely, e, e_1 , and e_2 . Thus, p is a Class I vertex.

The point of the preceding paragraph is to show that either a point, p , of an edge, e , is a Class I vertex or the line through p containing e has boundary points of \mathcal{O} arbitrarily close to p , *i.e.*, there exists a line segment $s \supseteq e$ contained in $\partial\mathcal{O}$ for which p is an interior point. In particular, an edge, e , containing a Class II vertex p can be extended to a line segment s lying in $\partial\mathcal{O}$ containing e whose endpoints are Class I vertices, *i.e.*, every edge of \mathcal{O} is contained in some skeletal edge. Since every vertex of \mathcal{O} must lie on at least three edges, every Class II vertex of \mathcal{O} must lie on at least two skeletal edges and hence $V(\mathcal{O}) = V^*(\mathcal{O})$.

Since every edge of \mathcal{O} lies in some skeletal edge and $V(\mathcal{O}) = V^*(\mathcal{O})$, it follows that the edges of \mathcal{O} are exactly the pieces into which the skeletal edges are partitioned by the vertices of \mathcal{O} . \square

Theorem 6 gives some insight into the working of the Projections algorithm. Back projection yields a *pseudo skeleton* consisting of a set of vertices which includes the Class I vertices and a set of edges. This pseudo skeleton is processed to produce a *pseudo wire frame*. In general, the pseudo skeleton and pseudo wire frame contain vertices and edges not in the skeleton and wire frame of the original object. However, they do contain all the vertices and a partition of the edges of the skeleton and wire frame of the original object. In fact, the additional complexity of the Projections algorithm is based on the fact that back projection generally yields many vertices and edges not in the original object. The Projections algorithm thus proceeds along the lines laid down by the Wire Frame algorithm, but with suitable modifications made to deal with surplus information.

The discussion of Class II vertices in the proof of Theorem 6 shows that they have various properties, one

of which appears as Theorem 7. Theorem 7 is very useful in showing that certain points which arise from back projection cannot be vertices of O . Example 4 later in the paper illustrates the power of this observation.

Theorem 7

Let O be an object and v a Class II vertex of O . Any plane, P , through v separates \mathbb{R}^3 into two components each of which contains interior points of O which are arbitrarily close to v .

Proof From the proof of Theorem 6 it follows that v is the intersection of two noncolinear line segments, e_1 and e_2 , which are unions of line segments of O ; are contained in the boundary of O ; and contain v as an interior point. Any plane through v not containing e_1 or e_2 is clearly going to contain interior points of O near v , and in this case this theorem is true. Also, there is only one plane, P , containing e_1 and e_2 . If all of O were to one side of P , there would be a contradiction, since at least four noncoplanar faces would go through v , but all the edges containing v would be coplanar. \square

The remainder of this section shows how much simpler things are when items are labeled or when special projections are used. The discussion of the unlabeled case is resumed in Section 3.

In mechanical drawing practice, one generally starts with $O|XY$, $O|YZ$, and $O|ZX$, although it is always possible to use other planes. In fact, as will now be shown, for each object O it is always possible to find a plane P such that π_P distinguishes all the elements of $WF(O)$.

Proposition 8

Let O be an object. Then there exists a plane P containing the origin for which π_P projects each element of $V(O)$ into a distinct vertex of $O|P$, elements of $E(O)$ project into distinct line segments which can intersect in at most one point, and no point in $V(O)$ projects into a projection of an element of $E(O)$ unless it is a member of it.

Proof The set of all planes in \mathbb{R}^3 containing the origin can be identified with the unit sphere, S^2 , in \mathbb{R}^3 , where each unit vector corresponds to the plane for which it is a unit normal. Clearly, in this manner exactly two points of S^2 correspond to each plane through the origin. In order for a projection π_P to map each vertex of O to a distinct member of $V(O|P)$, P cannot be perpendicular to any line which goes through at least two of the points of $V(O)$ and cannot be perpendicular to any plane containing all edges incident with a Class II vertex. Each of these restrictions rules out exactly one plane, i.e., two points on S^2 . Thus, in order to get an injection on $V(O)$, at most

$$2 \left[\binom{|V(O)|}{2} + |V(O)| \right]$$

points on S^2 must be avoided.

Two elements of $E(O)$ can have an intersection of more than one point in some projection if and only if they are coplanar. Furthermore, they can project with a nontrivial overlap only into planes which are perpendicular to the plane containing both of the elements of $E(O)$. The set of all planes through the origin perpendicular to a given plane corresponds to a great circle of S^2 . Thus, to get the desired behavior at most

$$\binom{|E(O)|}{2}$$

great circles on S^2 must be avoided.

To keep a point from projecting onto a line segment not containing it there are two cases to consider. First, the point and line segment might be colinear. In this case, one must avoid the plane perpendicular to the given line. Again this means avoiding two points. Thus, at most $2|V(O)| |E(O)|$ points must be avoided. Second, the point and line segment are not colinear. In this case it is enough to avoid all planes perpendicular to a given plane as before. Thus, at most $|V(O)| |E(O)|$ great circles on S^2 must be avoided.

Since points and great circles are nowhere dense in S^2 and the number of sets which must be avoided is finite, it follows from the Baire Category Theorem (see [15]) that there must be points of S^2 which do not lie in any of the forbidden sets. Using any such point yields a plane with the desired properties. \square

Definition 9

Let O be an object, P a plane in 3-space, and π_P the projection of 3-space onto P . Projection π_P is said to be a *distinguishing projection* for O if it has all the properties of Proposition 8. \square

Note that the proof of Proposition 8 shows that for a given object "most" projections are distinguishing projections since the nondistinguishing ones have a two dimensional measure of 0. The probability of picking a nondistinguishing projection at random is thus zero in an ideal model. However, in most practical situations there are only a finite number of choices for coordinates, and there is a nonzero probability of picking a nondistinguishing projection. Many objects of engineering interest have planar features aligned with the "natural" axes of the object, and the set of three standard views contains a maximum degree of concealment and self-alignment.

At this point it is worthwhile to consider two cases. In the first case, the image of each vertex in each projection carries the labels of all the vertices of O that project into it, i.e., the P -projections are labeled. In the second case, there are no labels on the vertices of the P -projection.

In the first case there is, quite naturally, significantly less ambiguity than in the second case. The following theorem shows exactly how much information can be recovered from labeled P -projections.

Theorem 10

Let P_1 and P_2 be two nonparallel planes in \mathbb{R}^3 , and let O be an object. Assume that the P_1 and P_2 projections of O are labeled. Then there is a unique set of points in \mathbb{R}^3 which can be $V(O)$. Furthermore, if either of the projections is distinguishing or if all the edges in at least one P -projection are labeled with the pairs of vertices they connect, then $WF(O)$ can be reconstructed uniquely. In this case, reconstructing objects from projections reduces to the problem of reconstructing objects from wire frames.

Proof If P_1 -vertices and P_2 -vertices are labeled, to reconstruct a point $x \in SV(O)$, the images of x under the two projections are found and perpendiculars erected at those points. Since P_1 and P_2 are not parallel, these perpendiculars can meet in at most one point. Since they both go through x , x can be recovered as their unique intersection point. In this way $SV(O)$ can be reconstructed uniquely, which, by Theorem 6, means that $V(O)$ can also be reconstructed uniquely. \square

Clearly, if the edges of at least one P -projection are labeled as described above, $E(O)$ can be uniquely reconstructed. If one of the projections is distinguishing, $E(O)$ can be reconstructed by joining together two points of $V(O)$ if and only if they are joined together in the distinguishing projection (or in both projections). \square

Thus, given a fairly small amount of information on projections, one can quickly and easily reconstruct a unique wire frame. In many practical situations, where the emphasis is on getting things done and not on creating puzzles, it seems quite likely that there will be ample information for constructing the correct wire frame easily. Unfortunately, there will also be many situations with inadequate information. The techniques developed for handling the unlabeled case are of great importance in such situations.

To complete the development of the labeled case, the situation in which there are no distinguishing projections must be discussed. Since this problem is a subset of the unlabeled case, the unlabeled case is considered next.

In the unlabeled case, there can be a number of distinguishing projections and it may not be possible to recover a wire frame uniquely. The following example illustrates this in the case of three distinguishing projections.

Example 11

Let O_1 be the tetrahedron with vertices $\{(1, 1, 1), (1, 2, 2), (2, 1, 2), (2, 2, 1)\}$ and O_2 the tetrahedron with vertices $\{(1, 1, 2), (1, 2, 1), (2, 1, 1), (2, 2, 2)\}$. The projections of O_1 and O_2 into the XY, XZ, and YZ planes are all distinguishing and are identical in each plane, but do not allow construction of a unique wire frame. Actually, the projections into the various planes are all essentially the same, i.e., by ignoring the coordinate which is fixed at 0 in each case, one gets the points $\{(1, 1), (1, 2), (2, 1), (2, 2)\}$ and the six possible lines between them, i.e., each projection looks like a square with both of its diagonals drawn in. In Section 5, the problem of reconstructing all objects for which all three standard projections look like a square with its diagonals is discussed in more detail. As shall be seen, there are surprisingly many solutions to this problem. \square

The above discussion shows that labeling projections can be very useful in reducing the difficulty of reconstructing objects from projections. The truth of the preceding sentence becomes even more apparent after the discussion of the algorithm for reconstructing objects from unlabeled projections in Section 3 and the discussion of the examples in Section 5.

3. Fleshing out unlabeled projections

In order to aid in the comprehension of this rather complex algorithm, a basic form of the algorithm, which accepts only limited data, is presented here (Section 3). The basic algorithm constructs all polyhedral solid objects whose wire frames have a given set of projections (or views). The extension of the algorithm to a more general set of projection forms (i.e., overall, detail, and cross section), and to the use of depth information to distinguish between visible and occulted edges, is deferred until Section 4. Since the Projections algorithm is an extension to the Wire Frame algorithm, the basic concepts of the Wire Frame algorithm and its terminology are reviewed first.

In the Wire Frame algorithm the input data [a wire frame, Fig. 2(a)] are processed to find all graphs containing more than two noncollinear edges. For each such graph, minimum enclosed areas are found and nested in a tree hierarchy. From this hierarchy candidate faces with an exterior boundary and possibly interior boundaries (i.e., a face may have holes) are constructed—these are

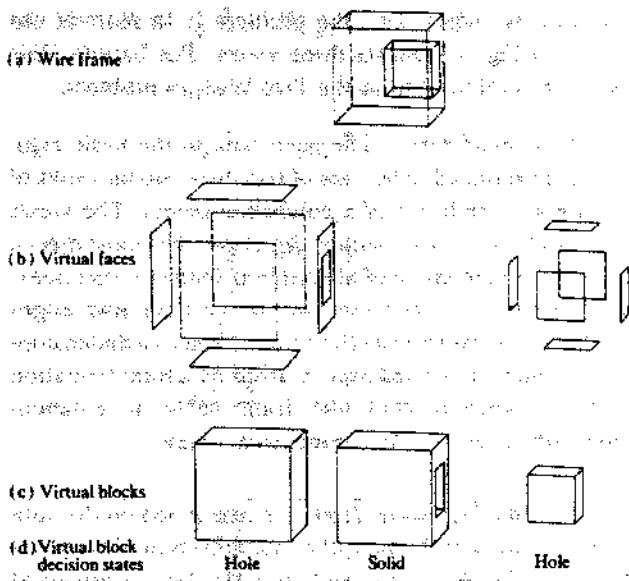


Figure 2. The Wire Frame algorithm in action.

minimum enclosed volumes are found and nested, again in a tree hierarchy. From this hierarchy, candidate volume regions called *virtual blocks* are found [Fig. 2(c)]. A final decision process assigns state solid or hole to each virtual block [Fig. 2(d)], glues the solid blocks together, and finds all possible solid objects with the input wire frame. Note that one virtual block is always an infinite envelope block (*i.e.*, it is inside out) and is always a hole.

The ability to handle all possible cases is embedded in the parts of the algorithm for finding enclosed regions (for example, bridges are ignored), for the handling of illegal intersections between virtual faces (Type I and Type II intersections, see below), and in the final decision process. The correctness of objects is derived from the use of directed edges and faces and from rules governing the number of times and directions with which edges and faces are used.

The several stages of the Projections algorithm are now described. Since many of these stages are quite similar to the corresponding stages of the Wire Frame algorithm, details are given about only those points which are different. The presentation is given in two parts: first, a brief outline of the stages, and second, a more detailed description of each stage.

The early stages (1, 2, and 3) of the Projections algorithm are concerned with converting, by means of a

back projection process, a set of projections of an object to a pseudo skeleton and thence to a pseudo wire frame for the object. This pseudo wire frame contains supersets of the vertices of all objects with the given projections. Furthermore, the edges of this pseudo wire frame partition the edges of all objects with the given projections. The existence of various edges and vertices in objects may be known for certain or may be uncertain. All components of the pseudo wire frame are consistent with all the views.

The later stages (*i.e.*, 4–7) apply an extended form of the Wire Frame algorithm to a pseudo wire frame to find all polyhedral solid objects with the given projections.

• Outline of the Basic Projections algorithm

1. Check input data The input data to the basic algorithm are assumed to be a set of at least two distinct parallel projections of the wire frame of a polyhedral object. Extensions to handle more general forms of input data are presented in Section 4. The data are checked for validity and reduced to canonical form with edges and vertices distinct and with edges intersecting only in vertices.

2. Construct pseudo vertex skeleton The vertices in each view are back projected to find all Class I vertices (*i.e.*, vertices formed by the intersection of noncoplanar edges) and some Class II vertices (*i.e.*, vertices formed by the intersection of only coplanar edges); at this point it is not possible to distinguish between vertex classes. The vertices discovered here, and the remainder of any Class II vertices missed in this stage and found in Stage 3, are called *candidate vertices*. While not all vertices of O may be recovered at this stage, enough are recovered to enable the recovery of all vertices after passing through the next stage. Note also that candidate vertices may not be vertices or even points of O .

3. Construct pseudo wire frame The vertices constructed in Stage 2 form a skeleton for the pseudo wire frame in the same sense that $WF(O)$ derives from $S(O)$. Edges are introduced based on the edges in the projections. These edges are checked for mutual internal intersections. Intersections are introduced as additional vertices and used to partition the edges. The remaining Class II vertices are constructed in this manner. The vertices constructed here and in Stage 2 are the set of candidate vertices (denoted $CV(O)$); and the final set of edges constructed in this stage is the set of candidate edges (denoted $CE(O)$). Together the candidate edges and vertices form the *pseudo wire frame*. The candidate vertices are a superset of $V(O)$, and the candidate edges partition the elements of $E(O)$. The edge connectivity of all vertices

is examined and the candidate edge and vertex lists edited. The editing process may remove impossible items, simplify colinear edges, and update the classification of vertices as Class I or II. Candidate edges and vertices which are the only possible candidates for some edges and vertices appearing in one of the projections are labeled as certain and must appear in a solution object; all others are labeled uncertain and may or may not appear in solution objects. For both candidate edges and vertices, cross reference lists are maintained between view edges and vertices and pseudo wire frame edges and vertices, and vice versa.

4. Construct virtual faces Beginning with the pseudo wire frame generated in Stage 3, all virtual faces are found in a manner analogous to that used in the Wire Frame algorithm. All uncertain edges are checked for containment in at least two noncoplanar virtual faces. Any edges not meeting this criterion are deleted and the virtual faces updated. Any impossible virtual faces (e.g., a certain edge piercing the interior of a virtual face) are deleted. The consequences of deletions are propagated until a stable condition is reached.

5. Introduce cutting edges Illegal intersections between two virtual faces such that both faces cannot exist in an object are handled by the introduction of a temporary *cutting edge* along their line of intersection. The cutting edge partitions the virtual face into smaller independent virtual faces and will be removed in the final stages. All the partitioning processes in the algorithm, be they of edges or faces, generate lists of siblings with common parent edge or face, and also lists of correlations between edges or faces which cannot co-exist in an object; these data structures are used in the final stages of the algorithm.

6. Construct virtual blocks Virtual faces are pieced together to form *virtual blocks* in exactly the same manner as in the Wire Frame algorithm.

7. Make decisions A depth-first decision process is used to assign solid or hole state to the virtual blocks and to find all objects with the given projections. The process ensures that all cutting edges disappear in solution objects (*i.e.*, that they are either totally surrounded by space or by material or they separate coplanar surfaces). Efficiency in the search process is obtained by careful pruning of the decision tree, for example, by recognizing that decisions involving partitioned edges and virtual faces may be propagated to the whole original edge or virtual face.

- **Detailed description of the Basic Projections algorithm**

To make the description of the algorithm more comprehensible, the example based on Fig. 1 is used to illustrate

the various stages, *i.e.*, the problem is to recover the object in Fig. 1 from its three views. For brevity, this problem is referred to as the Two Wedges problem.

1. Check input data The input data to the basic algorithm are assumed to be a set of two dimensional views of the whole wire frame of a polyhedral object. The views may be at arbitrary projection directions, but must meet a minimum requirement of at least two distinct projections. Each view is an ordered pair of vertices and edges (Definition 4) expressed relative to a local two dimensional coordinate frame and accompanied by a transformation matrix between the coordinate frame of the three dimensional object and the two dimensional view.

In this and later stages, tests are performed on the data input to a stage of the algorithm, for detection of inconsistencies in the data, for reduction of the data to canonical form for the stage, and to obtain information to be used in later stages. The exact choice of which tests to include depends on the characteristics of the input data and performance trade-offs between the cost of performing a test first, the usefulness of information generated for later stages, and the desirability of reporting errors before incurring the cost of executing the algorithm. These issues are not considered further here. However, it will be seen that the combinatorial problems of the projections algorithm may be very severe, and there is therefore a need to minimize the quantity of surplus information generated in the early stages of the algorithm.

2. Construct pseudo vertex skeleton As stated earlier, in this stage perpendiculars are erected at each vertex of each view. Then, only those vertices lying on at least two noncolinear perpendiculars and which are consistent with all other projections, *i.e.*, their images are either vertices or interior points of edges, are selected. As noted after Definition 4, all Class I vertices and possibly some Class II vertices are recovered. In order for the projections to be consistent, it is necessary that every P-vertex have at least one element of $CV(O)$ in its inverse image. This check may be performed as part of this stage. In addition, if some P-vertex has a unique element of $CV(O)$ in its inverse image, then that element of $CV(O)$ must actually be an element of $V(O)$. Such a vertex is assigned type certain, and all other vertices are assigned type uncertain.

Each intersection is tested to see if it coincides with a previously found vertex and, if not, is introduced as a new vertex. Each vertex found is accompanied by a list of cross references to the view-vertex pairs from which it has been generated. Conversely, for each view vertex, a list is formed of the wire frame vertices into which it projects.

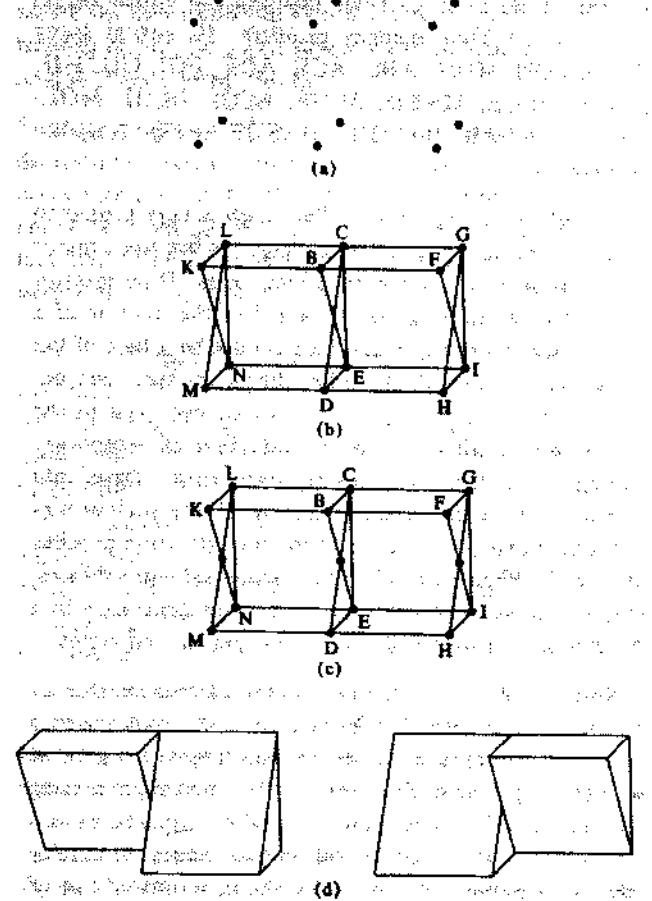


Figure 3 (a) The vertex pseudo skeleton of the Two Wedges problem. Edge recovery in the Two Wedges problem: (b) the pseudo skeleton and (c) the pseudo-wire frame. (d) The two solutions to the Two Wedges problem.

The pseudo vertex skeleton of the Two Wedges problem consists of 12 points: the 8 points corresponding to the vertices of a cuboid and 4 points corresponding to the mid-points of the 4 horizontal edges [see Fig. 3(a)].

3. Construct pseudo wire frame In this stage all pseudo skeletal edges are constructed as a prelude to constructing the pseudo wire frame. To do this, simply join two vertices in the pseudo vertex skeleton by an edge iff in every projection the images of these two vertices coincide or are joined by an edge or colinear set of edges and no other vertex of the pseudo vertex skeleton would be an interior point of the edge.

In general, these pseudo skeletal edges may intersect in mutually interior points. To obtain the pseudo wire frame from this skeleton it is only necessary to duplicate the techniques of Theorem 6, i.e., to introduce edges in the obvious way so that all edges have vertices as endpoints, that two edges intersect only in a vertex, and that no vertex be an interior point of an edge.

Note that the proof of Theorem 6 shows that $V(O) \subseteq CV(O)$ and that every edge of O can be written as the union of candidate edges.

Many of the checks of Stage 2 are used on the vertices produced in this stage. With modification these checks are used on candidate edges. Thus, it should be verified that every P-edge has some element of $CE(O)$ in its inverse image. In particular, if some P-edge has a unique inverse image, then that element of $CE(O)$ must be real, i.e., it must actually be an element of $E(O)$ and, like the rule for vertices above, is classified as type certain. At the end of this stage pruning operations are performed. All vertices with edge connectivity of degree ≤ 1 are removed, together with any incident edges. If the vertex has degree 2, the incident edges are checked for collinearity. If they are collinear, the vertex is removed and the two edges are merged into a single edge. If they are not collinear, they are removed together with the vertex. If a vertex of degree ≥ 3 has only coplanar edges, then any edges not having a collinear extension, and possibly also the vertex, are removed. Whenever edges are removed, the effects of the change are propagated until a stable configuration is achieved. In a similar manner to the vertices, cross reference lists are maintained from pseudo wire frame edges to view-edge pairs, and conversely, for each view edge, a cross reference list to the pseudo wire frame edges is formed.

Figures 3(b) and (c) show the results obtained during this stage in the case of the Two Wedges problem. Note that vertex A of the original figure appears in the pseudo wire frame exhibited in Fig. 3(c) but does not appear in the skeleton [Fig. 3(b)]. Note also that by Theorem 7 vertices J and O are clearly spurious since all solid material lies to one side of the planes KLN and FGL. However, these conditions cannot be derived until a later stage of the algorithm. \square

The stages described above are fairly straightforward. Before describing the later stages of the Projections algorithm it will be helpful to understand exactly what has been produced so far. The pseudo wire frame ($CV(O)$, $CE(O)$) looks like a wire frame. Indeed, in many cases ($CV(O)$, $CE(O)$) is exactly the wire frame of O and feeding ($CV(O)$, $CE(O)$) to the Wire Frame algorithm will yield the correct solutions directly. The important thing is to understand the way in which simply applying the Wire Frame algorithm to ($CV(O)$, $CE(O)$) can fail to find all solutions. The chief problem is that the original Wire Frame algorithm treats vertices and edges as real entities, whereas the pseudo wire frame contains uncertain edges and vertices, any of which may or may not exist in a solution. Any solid object having a subset of ($CV(O)$,

$CE(O)$) as its wire frame and producing the correct projections is a solution of the projections problem. Thus, the Wire Frame algorithm approach may fail to find all solutions of the projections problem (it may in fact fail to find any). The assumption of reality of edges and vertices is crucial to two places in the Wire Frame algorithm:

- Dealing with illegal intersections between virtual faces, and
- Making decisions.

Whenever an edge pierces a virtual face (a Type I intersection) in a legitimate wire frame problem, it is safe to drop the virtual face since it is known that the edge is "real" and that "real" edges cannot pierce faces which separate solid material from space (these are the only important faces). In the present situation, it might very well be that the edge is not real and should itself be dropped instead. Of course, if it is known that a particular edge is real (*i.e.*, certain), the algorithm can proceed as before.

In the Wire Frame algorithm the decision process was concerned with finding those combinations of virtual blocks which made every edge (except the cutting edges) an edge of a real object. In the case of the Projections algorithm it is necessary only to find combinations of virtual blocks with projections agreeing with the given projections. In general, this means that not every uncertain element of $(CV(O), CE(O))$ is actually a member of $(V(O), E(O))$. Thus, the decision procedure must be modified to check that every edge in each projection comes from a candidate edge which becomes a real edge in the corresponding solution.

Cutting edges were introduced in [1] to handle illegal intersections between virtual faces when no internal point of an edge from one face was contained in the interior of another face, but there were points common to the interior of both faces (a Type II intersection). This situation was interpreted as one where the two faces could not co-exist in the solution, and temporary edges—cutting edges—were introduced along the line of intersection of the two faces. The cutting edges partitioned the faces into nonintersecting sub-faces, which could be used to build more, smaller, virtual blocks. The decision process ensured that cutting edges did not remain in the final solutions. Although introduced originally for Type II intersections, cutting edges are applicable also to Type I intersections, and are particularly relevant to the case of uncertain edges.

4. Construct virtual faces. This stage is essentially identical with Stage 4 of the Wire Frame algorithm. As noted earlier, each candidate edge is checked to see

whether it lies in at least two noncoplanar virtual faces. Thus, in the Two Wedges problem, 19 virtual faces [KLO, LON, MON, ABC, ACE, ADE, FGJ, GIJ, HIJ, KLCB, NLCE, MNED, BCGF, ECGI, DEIH, MOLCAD, KONEAB, DACGJH, BAEIJF in Fig. 3(c)] are discovered.

5. Introduce cutting edges. This stage is very similar to its equivalent in the wire frame algorithm but has a minor modification to allow for uncertain edges. If an interior point of a certain edge is contained in the interior of a virtual face, then the virtual face cannot be a face of the object and is deleted. All other illegal intersections between virtual faces, *i.e.*, both faces cannot exist in the object, are handled by the introduction of temporary cutting edges. Cutting edges separate virtual faces into independent regions so far as the illegal intersection was concerned and are removed in the final decision process in Stage 7. When a virtual face is partitioned into subsfaces, mapping tables and correlation lists are generated in a manner similar to that described for partitioned edges.

Note that if records are kept in the correct manner all reprocessing of virtual faces is done with reference to a particular virtual face, rather than starting with a general wire frame problem. Furthermore, if, when reprocessing a virtual face, f , to determine the smaller virtual faces into which it is partitioned by the cutting edges, a cutting edge, e , is found which is not on the boundary of one of the smaller virtual faces, then it can be dropped together with any virtual face, g , whose intersection with f is e . Face g can be dropped since it is impossible for g to be a member of a virtual block. As usual, dropping a virtual face will in general have other repercussions which are exploited until a stable situation results. For brevity, virtual faces found in Stage 4 will be called *original virtual faces*. Those arising because of cutting edges will be called *new virtual faces*.

In the Two Wedges problem, two cutting edges [OA and AJ in Fig. 3(c)] are introduced. These two edges partition four virtual faces (CADHJG, BAEIJF, MOLCAD, KONEAB) into eight virtual faces (CAJG, ADHJ, BAJF, AEIJ, KOAB, ONEA, MOAD, OLCA).

6. Construct virtual blocks This stage is identical with the corresponding stage in the Wire Frame algorithm. In the Two Wedges problem, six finite virtual blocks are uncovered:

- B_1 :(MONEAD),
- B_2 :(NOLCAE),
- B_3 :(LOKBAC),
- B_4 :(DAEIJH),
- B_5 :(EACGJI),
- B_6 :(BACGJF),

where the description of virtual blocks is in terms of the labeling of Fig. 3(c). The seventh virtual-block, B_6 , is the unique infinite empty block.

7. Make decisions The set of virtual blocks is fed to a decision procedure, which is an extension of the decision procedure used in the Wire Frame algorithm. The differences between the two procedures revolve around the fact that the Projections algorithm is aware that not every vertex and edge must be real.

The chief difference consists of the fact that whenever the nature of a new virtual face is determined (*i.e.*, whether or not it separates solid material and space), the same determination can be made for all other new virtual faces which are subdivisions of the same original virtual face. Furthermore, as soon as it is determined (or assumed) that an original virtual face, f , does separate solid material and space, all original virtual faces sharing a cutting edge with f are forced to be spurious. This means that any pair of virtual blocks using any part of any virtual face "cutting" f as a common boundary must both be assigned the same state. Similarly, if a virtual face is known to be spurious, all virtual blocks using any part of it as a boundary must have the same state.

These facts speed up the decision procedure considerably and offset the greater number of virtual blocks that have been introduced. Similar arguments apply to entire edges which have been partitioned in Stage 3. In the final solution, no cutting edge can be a real edge. Of course, all decisions respect the fact that the final outcome must be consistent with the original projections.

In this stage virtual blocks are fitted together to generate all objects with the given projections. Basically, each virtual block may have solid or hole state and, when a state assignment has been made to each virtual block, an object is obtained. However, not all assignments of solid and hole yield the desired projections. An assignment of solid or hole to the virtual blocks yields an object with the correct wire frame iff

1. Every certain edge element $e \in E(G)$ belongs to two noncoplanar virtual faces f_1 and f_2 , each of which belongs to one virtual block assigned solid state and one assigned hole state.
2. No cutting edge belongs to two noncoplanar virtual faces f_1 and f_2 , each of which belongs to one virtual block assigned solid state and one assigned hole state.
3. Every uncertain edge element $e \in E(G)$ may be assigned either to state certain and obeys the rule for certain edges (1) above or to state not-visible and obeys the rule for cutting edges (2) above, in a manner consistent with the input projections.

The decision process is performed by assigning states in a virtual block state vector, whose elements are ordered *a priori*. The first element of the state vector is the unique infinite virtual block, which is assigned the empty state. For each edge, a list is formed of the faces containing the edge and the blocks they bound; this list is sorted around the edge and allows the angular sequence of block state transitions to be discovered.

The decision process proceeds as a depth first search in the virtual block decision space tree. At any node in the tree, the current state vector is checked for consistency and consequential states are assigned. Thus, although the state vector may have dimension of many hundreds, the consistency check may be expected to prune large sections of the tree, while the propagation of consequential states may be expected to reduce substantially the number of decisions to be made.

The checks for consistency are essentially those listed above. The consequential state assignments are performed to meet the following criteria:

- A certain edge with all except one containing block assigned the same state forces the remaining block to be assigned the opposite state.
- An uncertain edge totally surrounded by either all material or by all space becomes nonvisible; an uncertain edge contained in blocks producing exactly two coplanar state transitions around the edge becomes nonvisible; an uncertain edge contained by blocks of both hole and solid states and with at least two noncoplanar state transitions around the edge becomes certain.
- An uncertain edge that is the only edge remaining to create a view edge becomes certain.
- A cutting edge whose surrounding blocks have the same state, *i.e.*, both solid or both hole, spanning regions 180 degrees apart, allows the same state to be assigned to all blocks around the edge.
- A cutting edge whose surrounding blocks have the same state <180 degrees apart around the edge allows any intermediate blocks to be assigned to the same state.
- A new virtual face which is a real face, *i.e.*, it separates blocks of different states, and which is a subdivision of an original virtual face formed by cutting edges allows the same solid-hole relationship to be given to all blocks containing sibling faces from the original virtual face. Similar rules apply when the face is not real.
- An uncertain edge which becomes a certain edge and which is a subdivision of an original wire frame edge allows its sibling edges to be upgraded to certain state.

In some cases, particularly those where there are high degrees of symmetry and a limited number of views, giving rise to many highly correlated uncertain edges, there may be a very large number of objects producing the given projections. Thus, although the depth first search and also heuristic search approaches to this problem [8] allow a solution to be found efficiently, an exhaustive search must ultimately be used, and efficient pruning of the decision tree is very important. It is evident that, in the case of problems with multiple solutions, the provision of rather small amounts of extra information by the user, for example, labeling of some uncertain edges, and assigning states to points in 3-space, can resolve the ambiguities completely. Thus, in a practical system, the user may be requested to assist with extra information when requested. The basis for the system requesting extra information in the early stages of the algorithm is the preponderance of uncertain edges, discovered in Stage 3, and self intersection of uncertain edges, discovered in Stages 4 and 5.

At this point it can be appreciated that the use of cutting edges has allowed construction of a set of virtual blocks having the property that every solution of the projection problem can be built out of the virtual blocks in this set.

Stage 7 feeds into an output module which puts the output together in forms which can be understood by the user of the system. In our implementation of the algorithm, the output is in the form of a polyhedron for the Geometric Design Processor system [2].

In the case of the Two Wedges problem, this stage produces the two solutions shown in Fig. 3(d). The decision procedure works as follows in this case. Suppose that the search in this case deals with the virtual blocks B_0, \dots, B_6 in that order. B_0 is known to be empty. Thus, the first branch of the decision tree corresponds to determining the state of B_1 .

If B_1 is assumed to be solid, MOAD is seen to separate solid from space. This means that the entire virtual face MOLCAD must separate solid from space. In particular, B_2 must be solid and B_3 empty. Thus, the next step is to decide whether B_4 is solid or empty. Assuming that B_4 is solid forces B_5 to be solid and B_6 to be empty. However, the object resulting from making B_1, B_2, B_4, B_5 solid and B_0, B_3, B_6 empty clearly fails to have the right projections. Thus, the decision procedure backs up to the B_4 decisions and assigns hole to B_4 . This means that the new virtual face DAJH is spurious and that the original virtual face DACGJH is spurious. Thus, B_4 and B_6 must have the same state. If they are both assumed to be empty, the

object that results is just a simple wedge, which clearly has the wrong projections. Thus, B_4 and B_6 must both be assumed to be solid. The object that results is a left-right transform of the original object in Fig. 3 and clearly has the correct projections.

On the other hand, if B_1 is assigned hole, B_2 and B_3 must both be assigned the same state. Clearly, if B_2 and B_3 are also empty, it is impossible to obtain the correct front and top views. Thus, B_2 and B_3 must be solid in this case. Furthermore, assuming B_4 to be solid forces B_5 to be solid and B_6 to be empty. This yields the original object. Assuming B_4 to be empty forces B_5 and B_6 to have the same state. The objects that result are both wedges of differing width and are clearly not solutions.

It is clear that keeping track of the number of objects remaining in the inverse image of a projected artifact can be helpful in the decision procedure, i.e., if assigning a particular state to a given virtual block removes the last vertex or edge in the back projection of some vertex or edge, then that assignment can be rejected and its consequences need not be explored further. \square

The following section describes ways in which additional information can be extracted from various drawing conventions. The final section contains examples which should clarify the discussion in this and the next section.

4. Additional information from drawing conventions
Designers and draftsmen use a number of conventions and aids to clarify and help reduce ambiguity in engineering drawings. Extensions to the Basic Projections algorithm are presented in this section. These extensions cover two concepts: the generalization of the set of types of views to include overall, detail, and cross sectional, and the use of depth and detail information expressed by line types. The presentation is made within the context of the various stages of the algorithm presented previously.

• *Stages of the algorithm reconsidered*

1. Check input data

In extending the basic algorithm to handle several different types of view (i.e., overall, detail, and cross sectional), the central problem is to be able to relate information from the different types of views. This is achieved here by classification of the edges of the object into two types: gross and detail. The gross edges describe the main structure of the object; the detail edges add more information in regions where there is fine structure in the object.

The edges of the views are labeled with edge types according to an agreed drawing standard. For example,

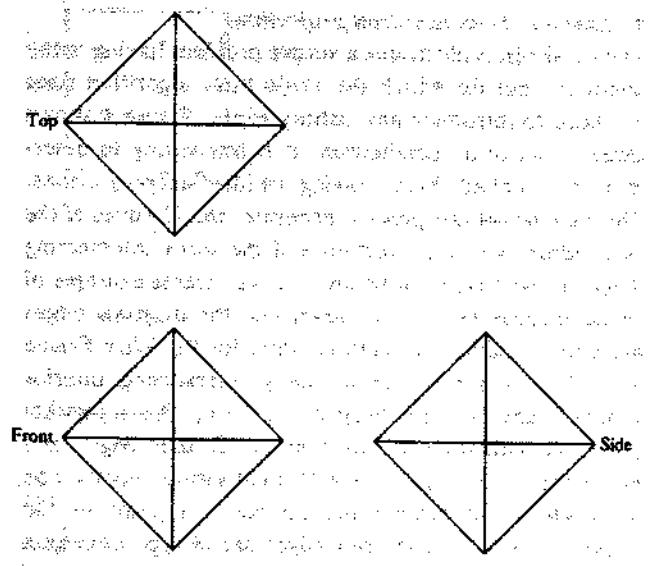


Figure 4 Three views of an object related to an octahedron.

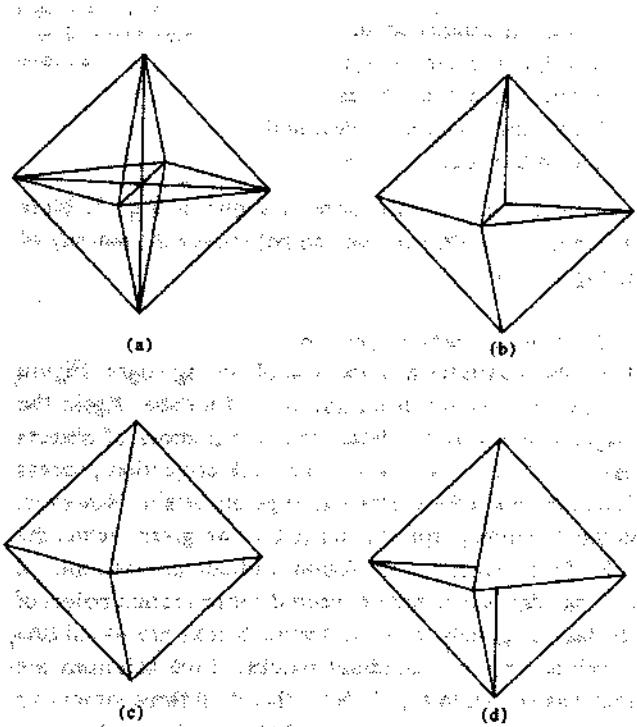


Figure 5 The solution to the problem of Figure 4: (a) the pseudo wire frame; all external edges are of type certain, internal edges are of type uncertain; (b, c, d) some of the 35 solid objects with the views of Fig. 4.

visible edges are generally drawn with line type *solid* and occulted edges with type *dashed*, which provides depth information. Another possibility, namely the omission of occulted edges, is not permitted; the Projections algorithm is based on geometric concepts and the premise that

all edges are shown in all projections. An algorithm that attempts to fill in missing information would have to be based on heuristic ideas of what a most likely object would be as well as on the concepts of geometry.

An overall view is a projection of the major features of the whole object onto a plane outside the object. The set of overall views of the object contain projections of only the gross edges. Thus, every gross edge of the object is represented as an edge or a vertex in every overall view. Similarly, every object vertex that is the intersection of gross edges appears as a vertex or a point in an edge in every overall view.

A detail view is a projection of a portion of the object. The view has a defined polyhedral boundary and two extents along the projection direction. The boundary and extents define a right prismatic region in 3-space. The detail view is a projection of all edges and vertices of the object contained in the region. A detail view contains projections of both the gross and detail edges, without distinction, contained within its defined region.

A cross sectional view may be either overall or detail. The view is a planar cross section normal to the projection direction. In this case the view transformation contains the location of the section plane in the coordinate frame of the object. Note that edges are shown at the cross section plane that may not be present in the object (they lie in surfaces of the object), and may not be shown in other views of the object.

2. Construct pseudo vertex skeleton

This stage proceeds in a manner similar to before. However, somewhat greater care must be taken to treat the various projections consistently. Intersections between back projections of vertices from appropriate pairs of different views are considered candidate vertices. Appropriate means noncollinear projection directions and the same type of view, i.e., both overall or both detail. In the case of pairs of detail views, the intersection point must lie within the intersection of their respective prismatic regions. In the case of a cross sectional view, the intersection point must lie in the halfspace defined by the section plane and projection direction. Also, a cross sectional view generates a set of vertices and edges in the plane of the view.

3. Construct pseudo wire frame

This stage is essentially unchanged from Stage 3 in Section 3. However, the following is a very useful observation: whenever a view shows two noncollinear solid (i.e., visible) lines intersecting internally in a point, p, then there must be some vertex of O visible in the appropriate direction which projects onto p and which has only visible edges incident with it corresponding to

the solid lines incident with p. In particular, if in moving along the perpendicular from p one first encounters candidate vertices which are clearly not vertices of O (see discussion of Stage 3 in Section 3), then these vertices and all incident edges may be discarded. To appreciate the power of this observation see Example 4.

4. Construct virtual faces

This stage is essentially the same as Stage 4 in Section 3. However, it is possible at this point to use line type depth information to edit out some type II vertices and uncertain candidate edges, as well as to extract additional information for use at a later time.

The cross reference lists from view edges to edges in the wire frame are concatenated with the list of original (*i.e.*, before any partitioning) virtual faces and sorted by distance along the projection direction from the mid-point of the view edge. For any edge that is visible, *i.e.*, not dashed, the nearest pseudo wire frame edge is identified. Any interposing virtual faces cannot exist and are deleted. For an edge to be dashed, there must be at least one occulting virtual face in the projection direction. If there is only one such face, then it must be a real face separating solid material from space, and since the projection is from outside, the directedness of the face is known. This information is fed forward to the decision process as initial certain states of blocks and faces. As before, the consequences must be fully propagated.

5 and 6. Introduce cutting edges and form virtual blocks

These stages are the same as in Section 3.

7. Make decisions

This stage again is very similar to the corresponding stages described in Section 3. Clearly, however, the decision procedure must accommodate the drawing conventions in the correct manner. It is fairly apparent how this is to be done. Thus, for example, in the case that occulted edges are represented explicitly in views, each view edge must contain a visible edge in the view projection direction, and each nonvisible view edge must be occulted by an interposed face in the view projection direction. □

The examples in the next section illustrate the points made above. As shall be seen, pathological features do not appear to be common in objects of practical interest.

5. Examples

To clarify the discussion in Sections 3 and 4, several examples are presented in this section. The examples are chosen to illustrate particular features of the algorithm and some of the performance trade-offs involved in providing extra information.

• Example 1—octahedron projections

The octahedron illustrates a simple problem having many solutions, but for which the Projections algorithm does not need to introduce any cutting edges. Figure 4 shows three views of an octahedron. It is interesting to determine the set of all objects having the identical projections. The back projection process generates the 12 edges of the octahedron with type certain and the three intersecting diagonals with type uncertain. In a wire frame example of an octahedron [1] it was shown that the diagonal edges must be introduced as cutting edges for the Wire Frame algorithm to handle the mutually intersecting interior virtual faces. In the Projections case, the algorithm proceeds with no need to generate further edges and enters the decision process with eight virtual blocks, one for each octant around the intersection point of the diagonals. Since the interior edges are of type uncertain and the exterior are all of type certain, any selection of octants such that no two hole octants share a face is a solution. The decision process finds 35 solutions:

- 1 with all octants solid,
- 8 with one octant a hole,
- 16 with two octants holes,
- 8 with three octants holes, and
- 2 with four octants holes.

A sampling of these solutions is shown in Fig. 5. Note that in this case dashed lines do not reduce the amount of ambiguity. □

• Example 2—cube projections

The cube illustrates a simple use of cutting edges. Figure 6 shows two views, front and top, of a cube. Again the Projections algorithm determines the number of objects having the same two views. The back projection process finds the cube edges, albeit as type uncertain. However, in the direction perpendicular to the two given views, the cube face diagonals are found without intersection. A cutting edge is inserted between the intersection points of the face diagonals, and five virtual blocks are found (the envelope and four quadrant blocks). Five solutions are found as shown in Fig. 7. Note that if all three views of a cube were furnished, there would be a unique solution to this projections problem. □

• Example 3—Two Y's problem

Figure 8 shows a well known mechanical drawing puzzle: find all objects having the top and front views shown. Because of the way edges line up in the two views, the back projection process finds the pseudo skeleton with 29 edges and 12 vertices shown in Fig 9. Intersections of the edges yield three additional vertices where the diagonals intersect, and intersections of virtual faces yield eight cutting edges. The final pseudo wire frame is shown in

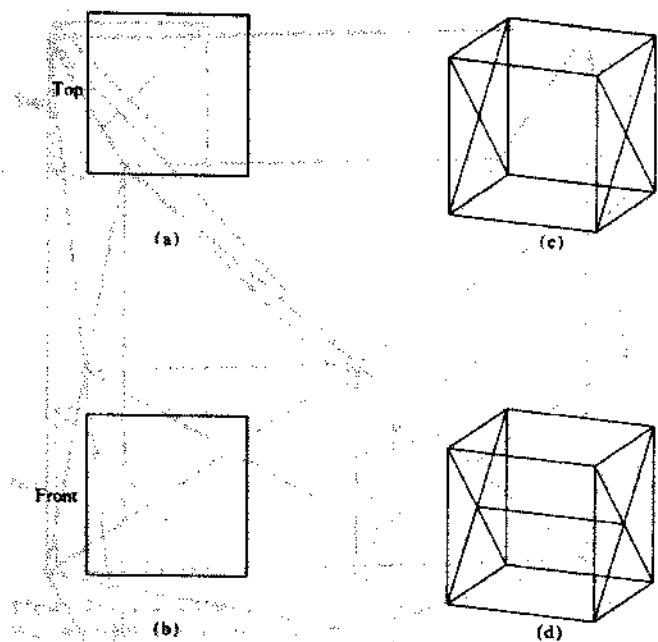


Figure 6. (a, b) Two views of an object related to a cube; (c) the pseudo wire frame; (d) the pseudo wire frame with a cutting edge inserted.

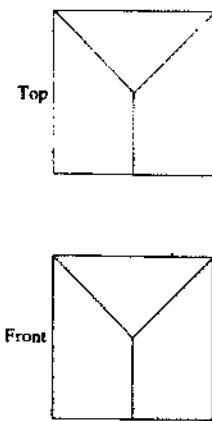


Figure 8. The Two Y's, a well known mechanical drawing puzzle: front and top views of an object.

tions of Section 4 (*i.e.*, all lines in the views are assumed to be solid, that is, visible) there are seven solutions. These are shown in Fig. 12.

The Two Y's problem is very sensitive to numerical considerations. If the branch point of one of the Y's is moved from the center of its view, there are no solutions to the corresponding projections problem. □

• Example 4—Three X's problem

The Three X's problem illustrates vividly the savings that can result from the use of depth information. Figure 13 shows an apparently minor modification to the problem of Fig. 4; the object is now clearly contained within a cube. However, further investigation shows that the solution process becomes surprisingly complex. The back projection process produces nine vertices—the cube vertices (uncertain) and its midpoint (certain) and thirty-two edges, all uncertain—the twelve cube edges, twelve face diagonal edges (initially with type II intersections, but later changed to mutually exclusive intersections), and eight cube diagonals from the midpoint. Note that, in contrast to the situation with Fig. 4, none of the edges found are of type certain and that ambiguities can be expected to stem from this lack of definite information. The pseudo skeleton that is obtained by back projection is shown in Fig. 14.

In the case without depth information, *i.e.*, all edges drawn regardless of occultation, the partitioning process, of intersecting edges to generate sub-edges and virtual faces to generate sub-virtual faces with cutting edges, divides space into many small regions. A total of 96 internal virtual blocks are found and the decision process uncovers 38 065 solutions. Clearly, searching a 96-level decision tree for 38 065 solutions is a complex process.

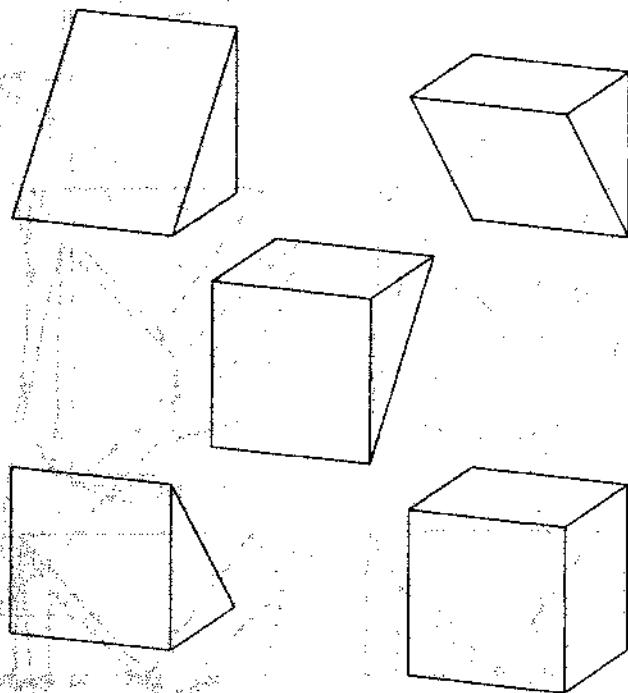


Figure 7. Objects with the views shown in Fig. 6 (a, b).

Fig. 10. The 16 internal virtual blocks found in Stage 7 are shown in Fig. 11. Under the assumptions of Section 3 there are 55 solutions to this problem. Under the assumption

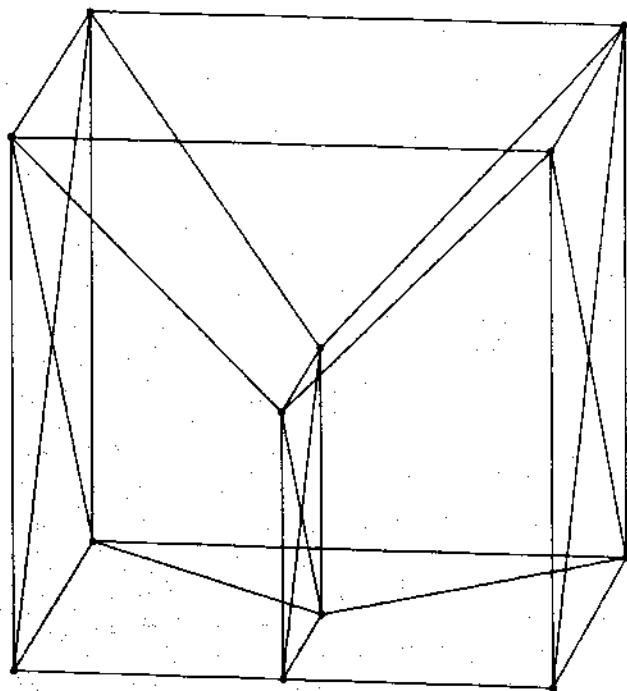


Figure 9 The pseudo skeleton of the Two Y's problem.

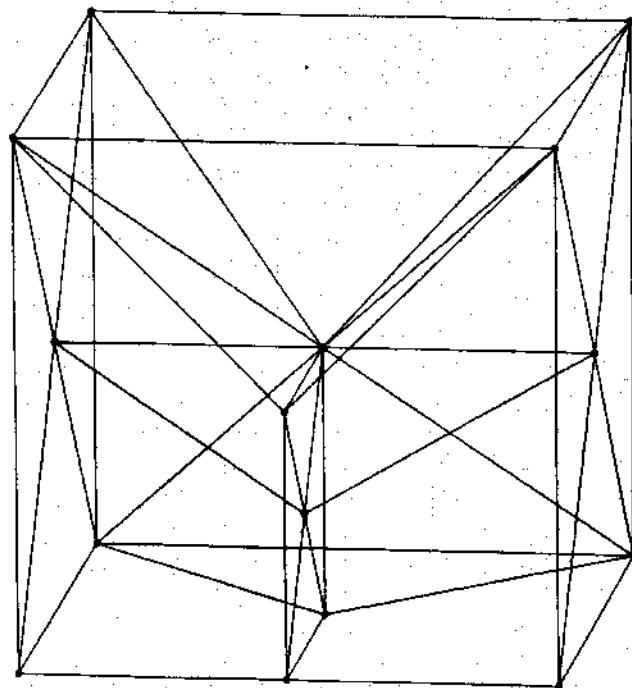


Figure 10 The pseudo wire frame with cutting edges added.

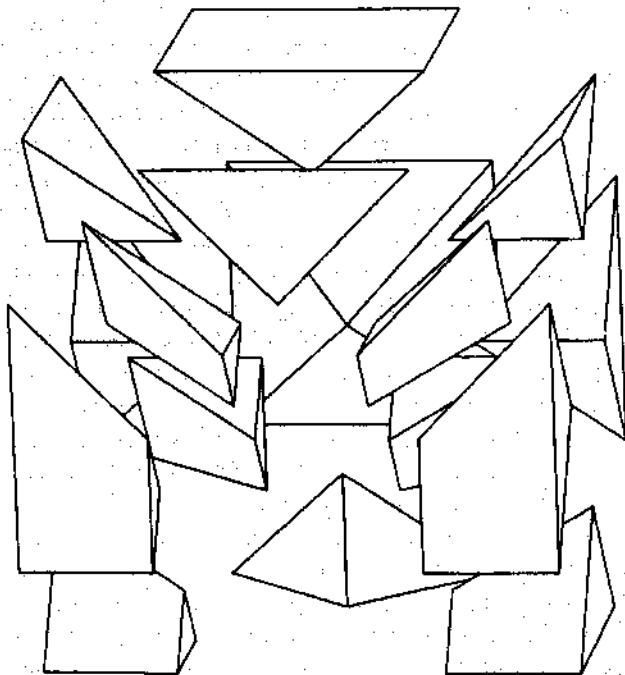


Figure 11 Sixteen virtual blocks found from the two views of Fig. 8.

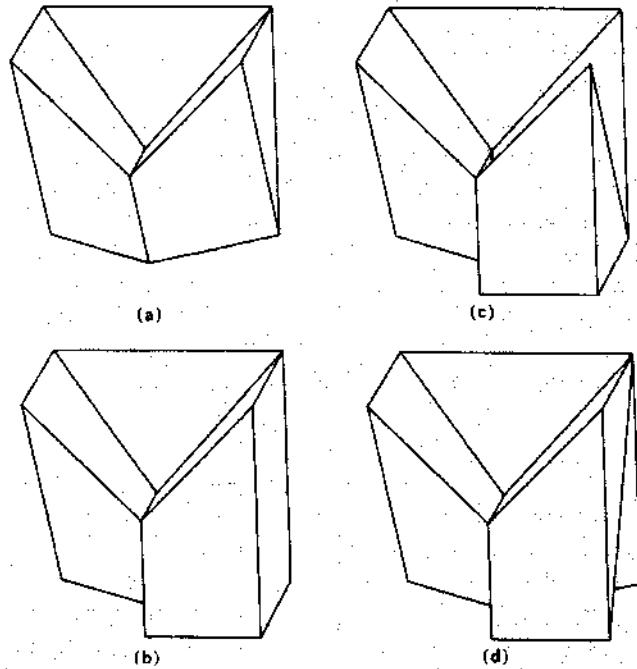


Figure 12 Objects with the two views of Fig. 8: (a) is symmetric; (b, c, d) are asymmetric and each is typical of a pair of objects.

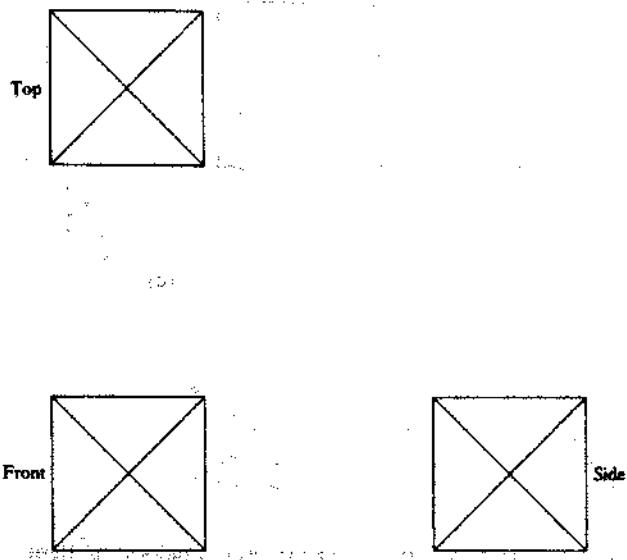


Figure 13 The Three X's problem: three views of an object whose extent is bounded by a cube.

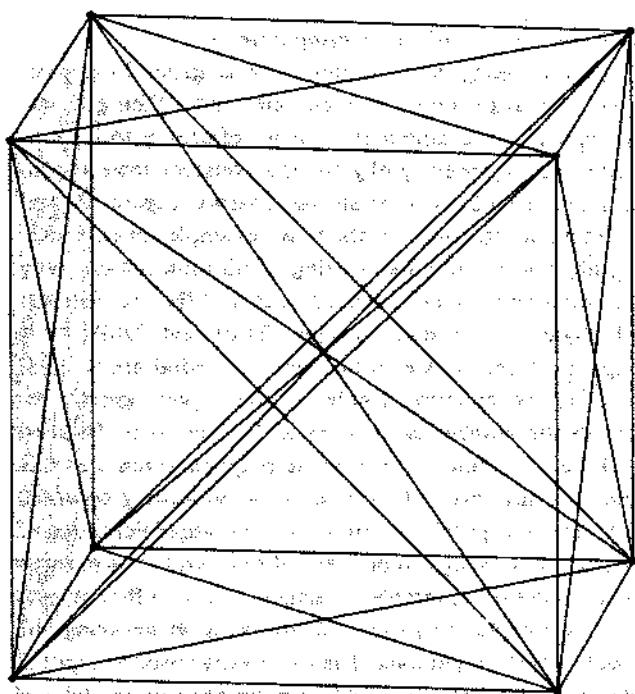


Figure 14 The pseudo skeleton for the Three X's problem.

The solution is made practicable by making heavy use of the mappings and correlations between original faces and edges and their partitioned forms. One solution, picked at random, is shown in Fig. 15. The object is hard to

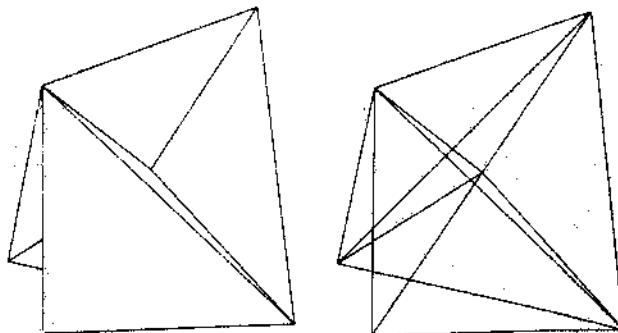


Figure 15 One of 38,065 objects found with the views of Fig. 13 and assuming that all edges are shown in each projection. The object is based on three tetrahedra.

understand, even with a model in one's hand. It is a set of three tetrahedra, a pair with a common face and a third with edge contact only, i.e., it is decomposable into two disjoint objects. The solutions found could be filtered to reject unstable objects of this form, but this test has not been executed.

The analysis of the case with depth information shows the power of Theorem 7 when used in Stage 3. Each of the three views shows solid lines intersecting in the center point. Following a perpendicular from any of the center points of any view leads first to a point in the center of a face of the cube containing the pseudo wire frame. This vertex cannot be a Class I vertex since all candidate edges incident with it are coplanar. It also cannot be a Class II vertex since all solid material lies to one side of the plane containing the face in question. Thus, the center points and all diagonal edges may be discarded from the front, top, and appropriate side faces of the cube. Furthermore, in Stage 4 corresponding faces of the cube are also discarded since they would obscure a vertex and lines in the interior. With these faces discarded, three of the leading edges of the cube must be discarded also since they no longer contain at least two noncoplanar virtual faces.

After these reductions, the algorithm goes on to find the ten solutions shown in Fig. 16. The solutions may be considered as being based on the union of three pyramids, as shown in Fig. 16(a). In all solutions, the view of the objects in the projection directions are the four triangular faces of the union of the three pyramids. The distinguishing features between the solutions are cavities in the "rear"; the viewpoint for the solutions in Figs. 16(b)–(g) is chosen to illustrate these cavities. The solutions are grouped as follows:

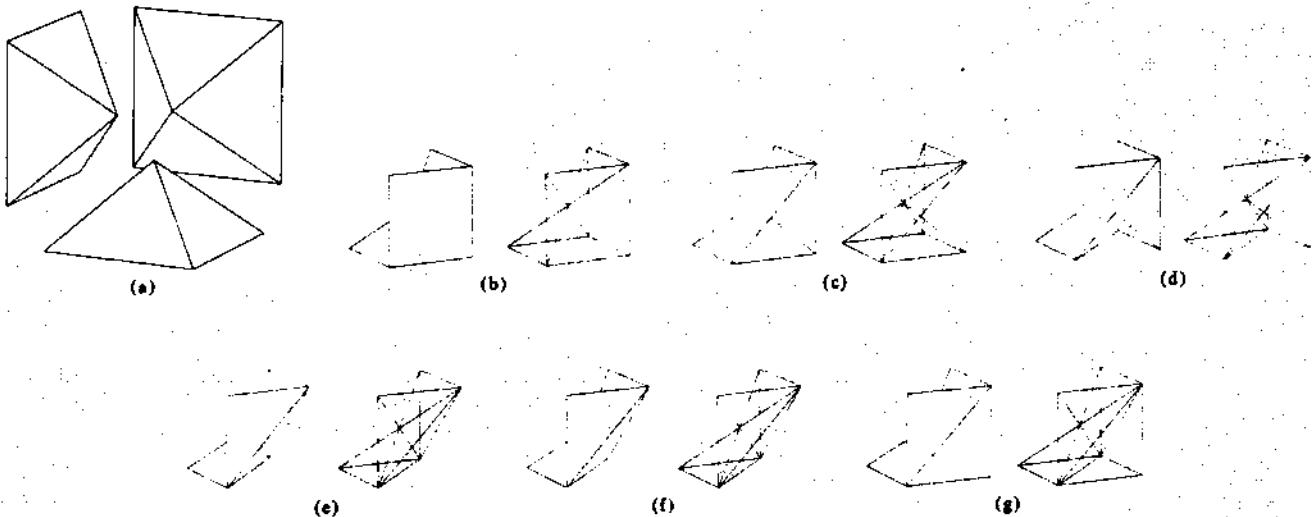


Figure 16 All ten solutions to the Three X's problem in dashed line mode: (a) three pyramids forming the basic solutions; (b) three pyramid solution; (c) one pyramid bisected; (d) two pyramids bisected; (e) all three pyramids bisected; (f) solution (b) cut by plane containing the diagonals of the square faces; (g) solution (b) with an internal tetrahedral cavity.

- Figure 16(b) shows all three pyramids complete,
- Figure 16(c) shows one pyramid bisected and is one of a set of three solutions,
- Figure 16(d) shows two pyramids bisected and is one of a set of three solutions,
- Figure 16(e) shows all three pyramids bisected,
- Figure 16(f) shows the object (b) above cut by the plane containing the diagonals of the three square faces;
- Figure 16(g) shows the object (b) with an internal tetrahedral cavity just visible as diagonal edges of the square faces.

• Example 5—Two Ramps problem

The Two Ramps problem illustrates the effectiveness of the pruning operations of Stage 3. Figure 17 shows this well known two view puzzle problem reputed to have twelve solutions. The back projection process produces an array of three by four, i.e., twelve, vertices on the left-hand face and an array of two by four for the right-hand face. Twelve edges are found linking the left and right sides. However, the number of possible edges in the end faces, i.e., in the direction normal to the two given views, is large; see Fig. 18. Fortunately many of those in the left-hand face are rejected by the edge and virtual face connectivity test at the end of Stage 3 (Fig. 19). Some 108 internal virtual blocks are found and 107 distinct solutions. Only 12 of the solutions, however, pass the stable object criterion. Some of the solutions are shown in Fig. 20. □

• Example 6—real engineering objects

After developing the algorithm to be as general as possible, and proving it with problems chosen for their geometric difficulty and ambiguities, it is refreshing to look at some real engineering objects and consider their reconstruction from their three standard views. Figures 21 and 22, parts (a), (b), and (c), show two examples of engineering objects. Even without using depth information, only one solution is found to each object, and the reconstructed objects are shown in Figs. 21(d) and 22(d). It is apparent from the views that the polyhedral approximations of the cylindrical holes in the objects greatly increase the number of vertices and edges to be handled and that the projections of these polyhedral features can lead to many small edges in the view, indicating potential for numerical problems. However, our implementation of the Projections algorithm does not have problems in these areas with these examples. Further, it is clear that objects of this complexity raise real problems in ensuring the validity of the input data. The three views used as input in this example were obtained from an existing model and were therefore guaranteed to be correct. A human generating these views directly would have some difficulty ensuring their correctness and self consistency. The Projections algorithm in its present form does not attempt to handle incorrect (or incomplete) data. □

6. Summary

The Projections algorithm presented in this paper finds all polyhedral objects O with a given set of projections. It has

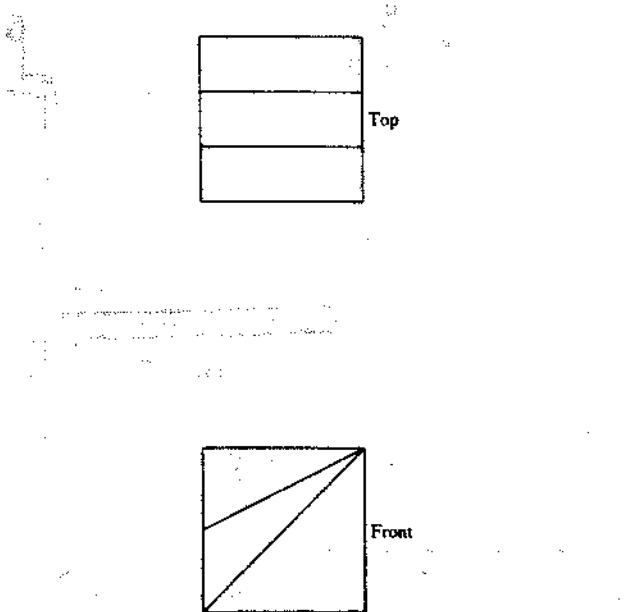


Figure 17. The Two Ramps problem: two views of an object.

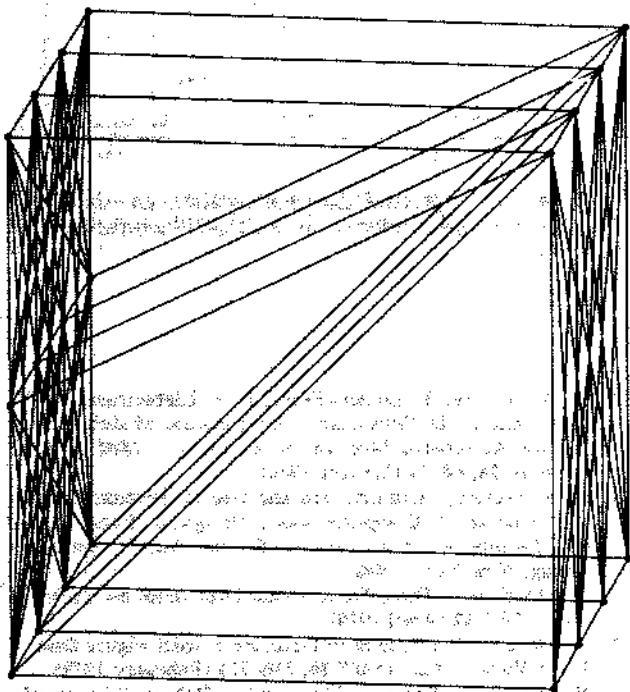


Figure 18. The pseudo skeleton of the Two Ramps problem.

been shown that, if the projections are labeled, the problem may be solved by the Wire Frame algorithm [1]; in the unlabeled case an extended form of the Wire Frame algorithm, the Projections algorithm, is needed.

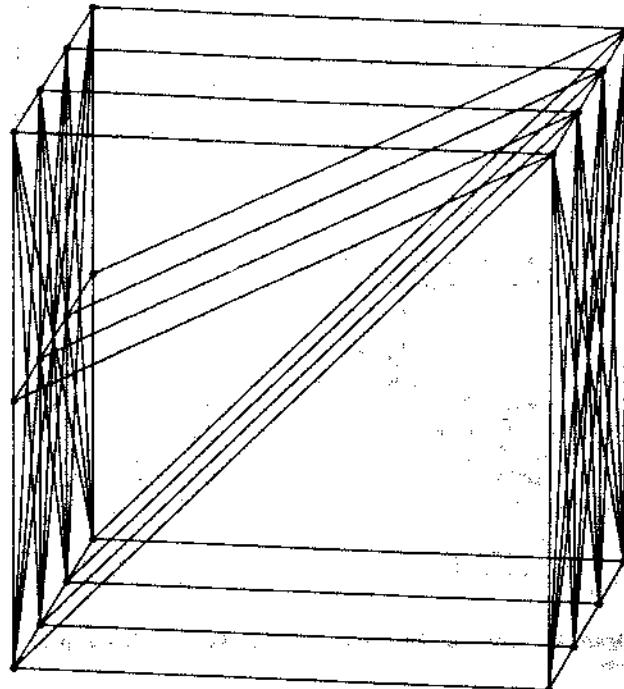


Figure 19. Pseudo wire frame after pruning in Stage 3; note the reduction of edges in the left-hand face.

It has been shown that an inverse projection process may be used to construct a superset of Class I vertices of \mathcal{O} (vertices contained in at least three noncoplanar edges) together with a superset of unions of edges of \mathcal{O} . These edges and vertices constitute the skeleton of \mathcal{O} .

It has also been shown that a superset of the Class II vertices of \mathcal{O} (vertices contained only in a set of coplanar edges) may be found as intersections of skeleton edges. These updated vertices and edges constitute a pseudo wire frame.

A pseudo wire frame differs from a wire frame in that it contains supersets of the edges and vertices of the wire frame. Some of these elements have been identified uniquely and have type certain; the rest are of type uncertain. Any object whose wire frame is composed of the certain elements of the pseudo wire frame and any subset of the uncertain elements and produces the correct projections is a solution.

The pseudo wire frame is processed to find candidate faces (virtual faces). Virtual faces are connected to enclose volume regions (virtual blocks). A depth-first decision process with heavy pruning is used to find all state assignments of hole or solid to virtual blocks that produce solid objects with the correct projections.

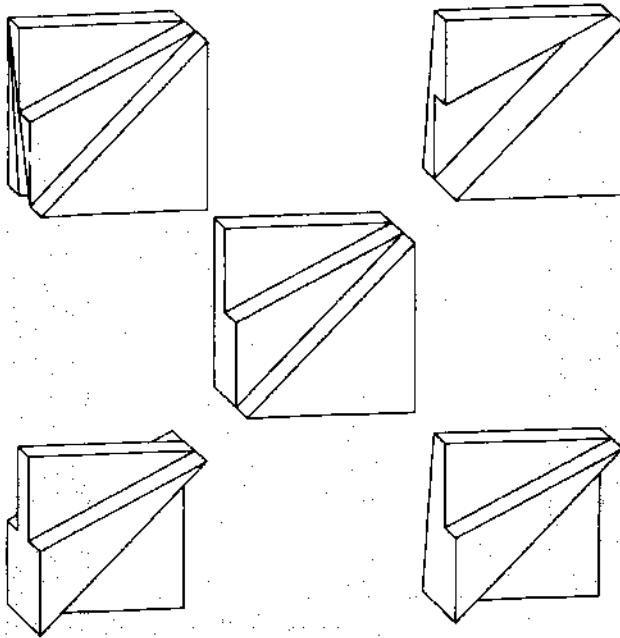


Figure 20 Some of the 107 solutions to the Two Ramps problem.

The Basic Projections algorithm accepts projections of the wire frame of \mathcal{O} ; extensions handle a more general set of projection types (detail, overall, and cross sectional) and projection conventions such as depth information obtained from occulted edges in a projection being shown as dashed.

The Projections algorithm has been implemented and its operation has been illustrated by a set of examples. These examples have shown that problems of a mechanical drawing puzzle nature, which typically have high degrees of symmetry leading to large numbers of uncertain elements in the pseudo wire frame, can have very large numbers of solutions. On the other hand, engineering objects, with projections sufficiently complex to require careful thought from a human, have been run and have produced unique solutions.

Acknowledgments

Our thanks are due to D. D. Grossman for encouraging us to tackle the projection reconstruction problem. A. F. Nightingale suggested the Two Ramps problem, and V. J. Milenkovic implemented the stable object test.

References

- George Markowsky and Michael A. Wesley, "Fleshing Out Wire Frames," *IBM J. Res. Develop.* 24, 582-597 (September 1980).

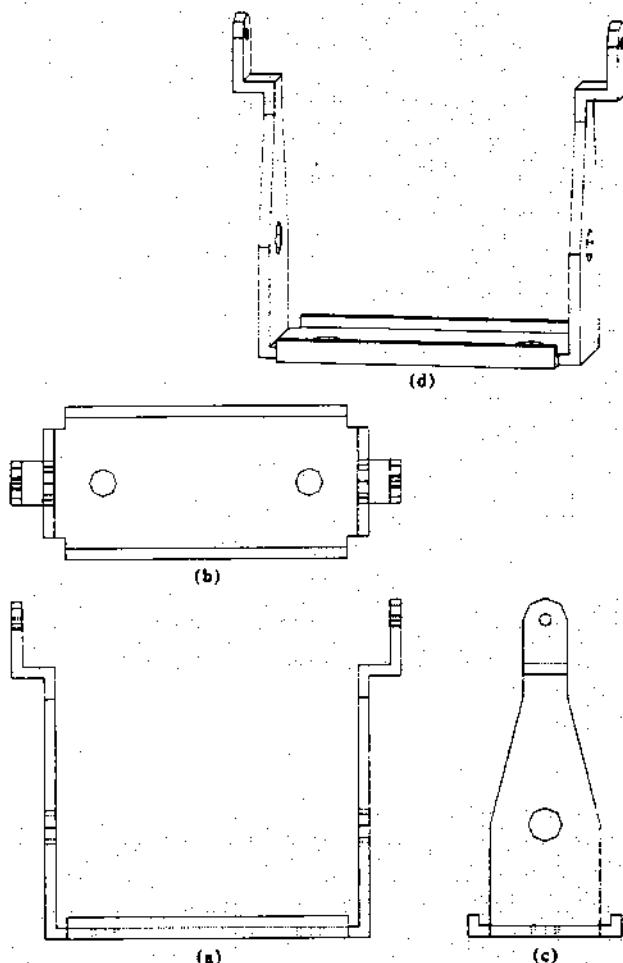


Figure 21 The Projections algorithm applied to an engineering object: (a, b, c) three views; (d) unique object constructed from the views.

- M. A. Wesley, T. Lozano-Pérez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman, "A Geometric Modeling System for Automated Mechanical Assembly," *IBM J. Res. Develop.* 24, 64-74 (January 1980).
- M. A. Wesley, "Construction and Use of Geometric Models," Chapter 2, *Computer Aided Design*, J. Encarnacao, Ed., *Lecture Notes in Computer Science No. 89*, Springer-Verlag, New York, 1980.
- I. Sutherland, "Three Dimensional Data Input by Tablet," *Proc. IEEE* 62 (April 1974).
- M. Idesawa, "A System to Generate a Solid Figure from a Three View," *Bull. JSME* 16, 216-225 (February 1973).
- M. Idesawa, T. Soma, E. Goto, and S. Shibata, "Automatic Input of Line Drawing and Generation of Solid Figure from Three-View Data," *Proceedings of the International Joint Computer Symposium 1975*, pp. 304-311.
- G. Lafue, "Recognition of Three-Dimensional Objects from Orthographic Views," *Proceedings 3rd Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing, ACM/SIGGRAPH*, July 1976, pp. 103-108.
- K. Preiss, "Constructing the 3-D Representation of a Plane-Faced Object from a Digitized Engineering Drawing," *Proceedings of International Artificial Intelligence Conference, Brighton, England*, April 1980.

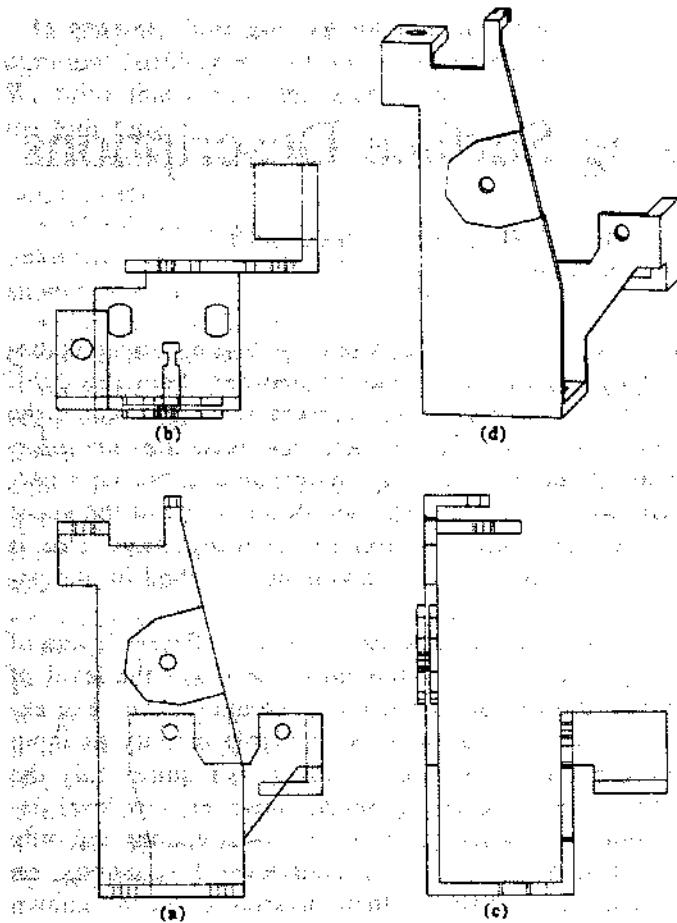


Figure 22. The Projections algorithm applied to an engineering object: (a, b, c) three views; (d) unique object constructed from the views.

9. T. C. Woo and J. M. Hammer, "Reconstruction of Three-Dimensional Designs from Orthographic Projections," *Proceedings of 9th CIRP Conference*, Cranfield Institute of Technology, Cranfield, England, 1977.
10. D. A. Huffman, "Impossible Objects as Nonsense Sentences," *Machine Intelligence 6*, B. Meltzer and D. Michie, Eds., Edinburgh University Press, Edinburgh, Scotland, 1971, pp. 295-324.
11. M. B. Clowes, "On Seeing Things," *Artificial Intelligence 2*, 79-116 (1971).
12. D. Waltz, "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Vision*, P. H. Winston, Ed., McGraw-Hill Book Co., Inc., New York, 1975, pp. 19-91.
13. T. Kanade, "A Theory of Origami World," Report No. CMU-CS-78-144, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, September 1978.
14. K. Sugihara, "Mathematical Structures of Line Drawings of Polyhedra—Towards Man-Machine Communication by Means of Line Drawings," Third Laboratory of the Department of Information Science, Faculty of Engineering, Nagoya University, Nagoya, Japan, May 1981.
15. J. G. Hocking and G. S. Young, *Topology*, Addison-Wesley Publishing Co., Reading, MA, 1961.

Received April 27, 1981; revised June 2, 1981.

The authors are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.

PAISLEY COLLEGE
OF TECHNOLOGY
LIBRARY

Recognizing 3-D Objects Using Surface Descriptions

TING-JUN FAN, MEMBER, IEEE, GERARD MEDIONI, MEMBER, IEEE, AND RAMAKANT NEVATIA, SENIOR MEMBER, IEEE

Abstract—Object recognition is a key problem in machine vision. The issues to be addressed relate to the representation of the data and the method used to establish correspondences. We present a system which takes as input dense range data and automatically produces a symbolic description of the objects in the scene in terms of their visible surface patches. This segmented representation may be viewed as a graph whose nodes capture information about the individual surface patches and whose links represent the relationships between them, such as occlusion and connectivity. Based on these relations, a graph for a given scene is decomposed into subgraphs corresponding to different objects.

For the purpose of matching, a model is represented by a set of such descriptions from multiple viewing angles, typically 4–6. Models can therefore be acquired and represented automatically.

Matching between the objects in a scene and the models is performed by three modules: the *screener*, in which we find the most likely candidate views for each object, the *graph matcher*, which performs a detailed comparison between the potential matching graphs and computes the 3-D transformation between them, and the *analyzer*, which takes a critical look at the results and proposes to split and merge object graphs. We present results on a variety of scenes containing multiple complex objects with occlusion.

Index Terms—Object description, object recognition, range image analysis.

I. INTRODUCTION

RECOGNIZING objects in a scene is a primary goal of computer vision. The difficulty of this task depends on several factors such as: the number and complexity of objects in the scene, the number of objects in the model database, and the amount of *a priori* information about the scene. The appropriate techniques for object recognition depend on the difficulty of the task. In our case, we attempt to recognize rather complex objects in range images.

The simplest tasks assume that objects are unoccluded and occur in one or more of standard viewing positions. This essentially reduces to a 2-D recognition task. In another set of tasks, model-based techniques are used.

Manuscript received July 22, 1988; revised May 1, 1989. Recommended for acceptance by O. D. Faugeras. This work was supported by the Defense Advanced Research Projects Agency under Contract F33615-87-C-1436, monitored by the Air Force Wright Aeronautical Laboratories, DARPA Order No. 3119.

T.-J. Fan was with the Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, Los Angeles, CA 90089. He is now with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

G. Medioni and R. Nevatia are with the Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, Los Angeles, CA 90089.

IEEE Log Number 8929959.

There, objects are recognized by finding some distinguishing features and relations between them [6], [15]. However, to handle more complex situations where the number of models may be large and little *a priori* information about the scene may be available, it becomes necessary to first use sophisticated descriptions of the scene and then use these descriptions for recognition. This is the scenario assumed in the system described in this paper.

One can match two scenes at many different levels of descriptions with some tradeoffs: the lower the level of the descriptions, the easier it is to compute them. For example, a range array may be available directly as input and an Extended Gaussian Image [19] requires only the computation of surface normals. However, low level descriptions are not invariant to viewing directions and little tolerant to occlusion. The higher level descriptions, on the other hand, maintain their invariance but the known algorithms to compute them are often weak and error-prone. The appropriate level of description to be used for matching thus depends on the expected variations in the scenes and on the state-of-art in computing descriptions of the scene.

We have decided, in this paper, to use object descriptions in terms of their surfaces. The surface of an object is described by segmenting it into *surface patches* and the complete description consists of the description of each patch separately, and their interrelationships. Such a description can be viewed as an *attributed graph* with the patches as the *nodes* and relations between them as the *links*. The segmentation and description of the surface is based on measured curvature properties of the surface. We describe this process briefly in Section III, complete details are given in [12], [13].

We believe that our chosen representation has many advantages for scene and object matching. The description is *rich*, so that similar objects can be identified, *stable*, so that local changes do not radically alter it, and has *local support* so that partially visible objects can be identified. It also enables us to recreate, from its features, a shape reasonably close to the original one. The surface descriptions are much higher level than pointwise or edge descriptions, but not as high as volumetric descriptions. The surface descriptions are invariant for smaller changes in viewing angle than would be the case for volume descriptions. However, techniques for volume descriptions are not yet fully developed, whereas our previous work allows us to have high-quality surface descriptions.

In general, there are two steps involved in object recognition: *building models* and *recognizing scene objects*. We think that a good recognition system should provide the following.

- It should automatically build object models from range images.
- The descriptions used for models and for scenes of unknown objects should be compatible, or at least it should be easy to go from one to the other.
- The search should be efficient in finding correspondences between models and scene objects.

We have decided to use the same description format for both model and scene objects. As a result, the computation of the model descriptions can be achieved automatically, and there is no need to *translate* one description into another. We have tested our system on a number of scenes containing multiple complex objects with occlusion, and obtained very good recognition results. Some examples are given later in Section V.

In the next section, we review existing object recognition systems. Section III provides a brief summary of how we go from a dense range map to a set of surface patches, and how these patches are further organized into partial objects. Section IV presents the details of our recognition system, which uses the descriptions given above. Section V shows results of our recognition system on real range images. Finally, Section VI summarizes our contribution.

II. SURVEY OF RECOGNITION SYSTEMS

In this section, we review the most successful object recognition systems to date, and point out how they address the issues of model representation, choice of scene features, and matching methods. We also discuss their limitations.

A. Extended Gaussian Image (EGI)

Horn *et al.* use multiview EGI models to recognize 3-D objects [19]–[21]. Each model object is represented by its mapping on the Gaussian sphere, and each scene object is also represented by an EGI. The EGI's of scene objects and model views are compared. To constrain the search space, the two EGI's are first aligned along the directions of minimum EGI mass inertia, then a match measure is specified by comparing the similarity in their mass distributions. The model that maximizes this measure is chosen as the matched model. This method has the advantage that EGI can be computed directly; no complicated description stage is needed. The main disadvantage is that EGI is sensitive to occlusion and is unique for convex objects only. Furthermore, when multiple objects are present in a single scene, it would be necessary to segment the EGI into regions corresponding to separate objects, and it is not clear how to achieve this, except for simple-shaped objects.

B. 3DPO

Horaud and Bolles [18] and Bolles *et al.* [6] developed the 3DPO system for recognizing and locating 3-D parts

in range data. The model consists of two parts: an augmented CAD model and a feature classification network. The CAD model describes edges, surfaces, vertices, and their relations. The feature-classification network classifies observable features by type and size. The system recognizes unknown objects by searching for features that match features for some model, for example, a cylindrical curve with a given radius. Then objects are hypothesized by determining whether a pair of observed segments are consistent with a given model's feature. This system has been shown to recognize objects in highly complex scenes, but with very few models. Since the models are represented in 3-D, it is very difficult to compute them automatically, therefore a CAD model is required that usually needs help from a user, it also needs a very complex network. Furthermore, this system relies heavily on detecting circular arcs and straight dihedral edges, so the shape of the objects it can recognize is restricted.

C. Grimson and Lozano-Pérez

Grimson and Lozano-Pérez [15], [16] discussed how local measurements of 3-D positions and surface normals can be used to identify and locate objects from among a set of unknown objects. Models consist of polyhedral objects represented by their planar faces. The information about these faces (such as their equations) and the relations between faces (such as distance) are also computed. Sparse range or tactile data of 3-D objects are used as scene features. The matching process contains two steps: in the first step, a set of feasible interpretations of the sensory data is constructed. Interpretations consist of pairings of each sensed point with some object surface of one of the models. Interpretations inconsistent with local constraints are discarded. In the second step, the feasible interpretations are verified by a transformation test. An interpretation is accepted if it can be used to solve for a transformation that would place each sensed point on an object surface. Only polyhedral objects or objects with a sufficient number of planar surfaces can be used in this system.

D. Ikeuchi

Ikeuchi [22] developed a method for object recognition in bin-picking tasks. Object models are generated under various viewer directions, apparent shapes are then classified into groups. Since this system is mainly designed for tasks for bin-picking, only one type of object, which is the same one as in the model, appears in the scene. The same surface features used in models are extracted and classified by the help of the model. In the recognition process, an *interpretation tree* is generated according to various model views. The orientation and location of the scene object are then decided by comparing their surface features and classified by the interpretation tree. Strictly speaking, this is not a recognition system, but a system which identifies a given 3-D object to be picked up.

E. Faugeras and Hebert

Faugeras and Hebert [14] developed a system to recognize and locate rigid objects in 3-D space. Model objects are represented in terms of linear features such as points, lines, and planes. The same features such as significant points, lines, and planes are used to describe scene objects. The system uses *rigidity constraints* to guide the matching process. At first, possible pairings between model and scene features are established, the transformation is estimated using quaternions. Then, further matches are predicted and verified by the rigidity constraints. This system has been shown to recognize fairly complex objects, but in scenes without occlusion from other objects. As the segmentation of objects in this system results in a large number of surface patches, and since the segmentation is not guided, we believe it is going to be highly sensitive to small changes in the surfaces.

F. Oshima and Shirai

Oshima and Shirai [25], [26] developed a model-based recognition system for objects with planar and curved surfaces. Each model is represented by a *relational-feature graph* whose nodes represent planar or smoothly curved surfaces, and links represent relations between adjacent surfaces. Matching is achieved by a combination of data-driven and model-driven searching processes. At first, *kernel nodes* which consist of large, planar surfaces with no occlusion are extracted. Next, an exhaustive search of all model graphs is performed and those containing regions which match the kernel nodes are selected as candidates. Finally, a *depth-first search* is applied to build the correspondences for remaining surfaces. Since only one view is used for each model object, if objects may be viewed from multiple directions, then a separate relational graph must be constructed for each view, and these models must be treated independently by the matching process. Furthermore, no occlusion is allowed for curved surfaces.

G. Nevatia and Binford

Nevatia and Binford [24] developed a system in mid-1970's that used generalized cones as the basic representation. This system uses range data as input. Model acquisition and scene description follow the same procedure, hence the models can be acquired automatically. This system also handles articulation of parts of objects. It is tested on objects of the complexity of a doll and a horse. This system also uses an *indexing* scheme based on gross features so that recognition does not require a search through all models in the database. The main limitation of this system would come from the ability to generate appropriate generalized cone descriptions for complex objects. This is a difficult problem as we discuss later.

A. ACRONYM

Brooks [9], [10] developed an image understanding system called ACRONYM which uses generalized cylind-

ers for descriptions of model and scene objects. The model objects are represented by hierarchical graphs of primitive volumes described by generalized cylinders (GC). The user constructs a tree for each object, where nodes contain parts of objects represented by GC, and links represent their subpart relation. The user is also required to construct a model class hierarchy called a *restriction graph*. This graph contains sets of constraints for different classes of objects and is used later to guide the match between model and scene objects. *Ribbons* and *ellipses* are used to describe scene objects. The descriptions are finally represented by an *observation graph* whose nodes contain ribbon and ellipse descriptions and links specify spatial relations between nodes. Matching is performed at two levels: first, predicted ribbons must match image ribbons, and second, these local matches must be globally consistent. The main restriction of ACRONYM is that models and restriction graphs are constructed by the user, which is very expensive and restricts the possibility of automatic model building. Furthermore, since both models and scenes are represented by GC's that consist of ribbons and ellipses, the shape of model and scene objects is restricted; in addition, the viewing direction is assumed to be approximately known.

Other contributions in object recognition are due to Bolles and Cain [5], Bhanu [4], Ayache [1]. A more detailed survey can be found in [2], [7], [11].

III. COMPUTING SYMBOLIC SCENE DESCRIPTIONS

As discussed in Section I, we have chosen segmented surface descriptions for representing objects in this work. Discontinuities such as *jump boundaries*, *limbs*, and *creases* are chosen to segment surfaces as they are explicit features of the surfaces of 3-D objects. Our method consists of two stages. In the first stage, surfaces objects are segmented at discontinuities which can be detected by examining the *zero-crossings* and *extremal* values of surface curvature measures. Then these detected discontinuities are used to segment a complex surface into simpler meaningful components called *surface patches* or *patches*. These patches can then be approximated by simple surface models. Finally, these surface patches are grouped into meaningful 3-D objects, and *attributed graphs* are generated to describe these objects.

A. Extracting and Describing Surface Patches

Given a complex surface, we wish to decompose it into simple surface patches. Our approach is to determine the boundaries of the surfaces by computing *local* properties and then inferring the patches from them. Similar ideas for segmentation have been used in [3], [8]. In particular, we seek to find the following kinds of boundaries.

1) *Jump boundaries*—where the surface undergoes a discontinuity.

2) *Creases*—which correspond to surface orientation discontinuities.

We infer these boundaries from curvature properties of the surface. In particular, we use curvature *zero-crossings*

and *extrema* [12]. A *jump boundary* creates a zero-crossing of the curvature in a direction normal to that of the boundary; a *crease* causes a local extremum of the curvature at that point. Crease boundaries may also create zero-crossings away from the location of the boundary itself.

These features, when connected using contiguity, give us *partial boundaries* for patches in which the surface should be segmented but not necessarily a *complete segmentation*. These partial boundaries are then completed by simple extension [12]; we have found this process to be satisfactory for the large number of examples we have tested it on. The resulting *regions* are assumed to correspond to elementary surface patches. These regions could be segmented further, either based on the region shape or on the results of surface fitting; we have not found it necessary to do so for the examples we have tried.

The surface patches are approximated by a second-order polynomial in (x, y, z) , whose coefficients are computed by a least-squares method. The details are given in [12]. Figs. 1 and 2 illustrate this early processing on a synthetic and a real range scene, respectively.¹ In both figures, (a) shows the shaded image, (b) shows the curves obtained by our feature detection process, and (c) shows the regions bounded by the previous curves.

B. Object Inference

At the end of the above process, we have a symbolic representation of a scene as a *graph* whose nodes represent the patches and whose links express geometric relationships between patches, making it an appropriate level to use in a matching process. We can further group these patches into *objects*, or rather visible faces of objects, by reasoning on the type of connections between adjacent patches. This higher level of description facilitates the matching process. The procedure used to obtain these *objects* is described in detail in [12], we only provide the outline below.

Up until now, we have classified the boundaries into two classes only: jump boundaries and creases. Once each patch is approximated by an analytic function, it becomes possible to distinguish true jump boundaries from *limbs* (also called axial contour generators [28]), for which the normal to the surface becomes perpendicular to the viewing direction. This distinction is important for matching, as limbs are *not* intrinsic properties of an object, but depend on the viewing direction. We therefore end up with the following set of four labels: *convex creases* (+), *concave creases* (-), *limbs* (L), and *jumps* (J).

¹Most researchers display range images by encoding depth by grey level, but this produces images with very poor dynamic range; in the rest of the paper, we present range images by borrowing a technique from computer graphics: we assume that the object is Lambertian, compute the normal to the surface in a small (3×3) neighborhood, and generate a reflectance image in which the intensity is inversely proportional to the angle between the light source and the surface normal. (The brightness of a point on the surface is proportional to the cosine of the angle between the light source and the surface normal under the Lambertian assumption.)

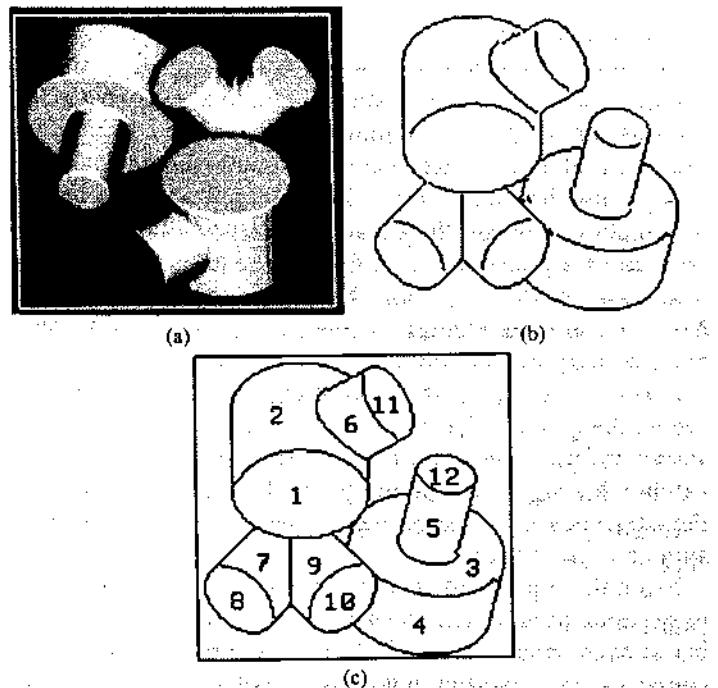


Fig. 1. Segmentation of a synthetic range image: (a) original scene, (b) detected discontinuities, (c) result of segmentation.

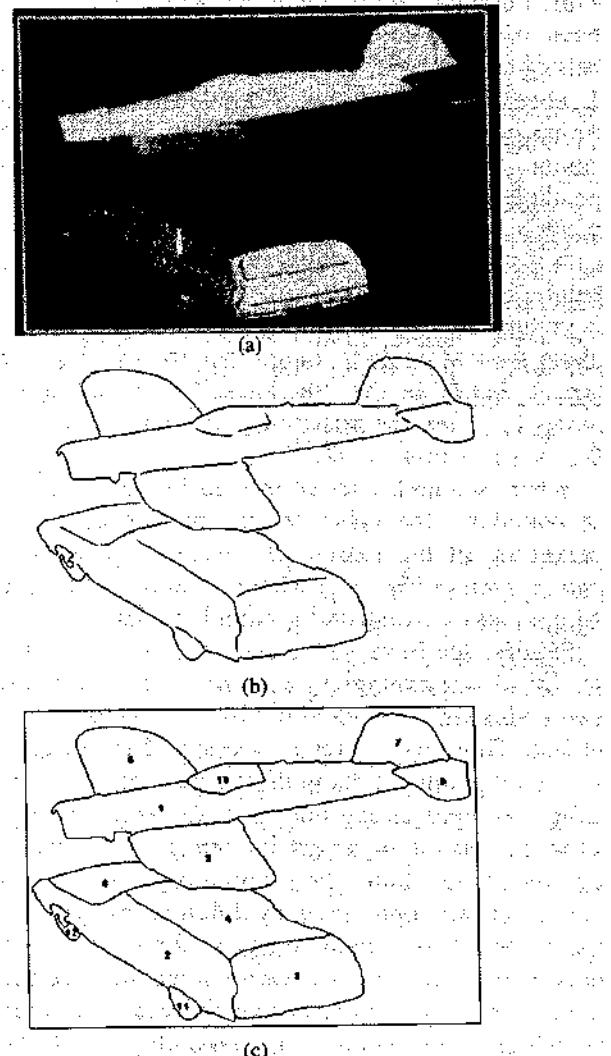


Fig. 2. Segmentation of a real range image: (a) original scene, (b) detected discontinuities, (c) result of segmentation.

The match primitives are composed of graphs and subgraphs. We create a *node* for each surface patch, which contains the *unary* information about the surface patch such as the shape, orientation, and location. Then, for each pair of nodes that share a common boundary, we create a *link* between them. This link contains the *binary* information between these two nodes such as the boundaries and the *possibility* that these two nodes (which represent two surface patches) belong to the same object. Note that one node contains one surface patch only, but one link may include multiple boundaries.

Surfaces are only parts of solid objects. In a segmented scene, the type of adjacency between two patches conveys strong information regarding whether or not these two patches belong to the same object. Based on our labels, the adjacency information we can derive is that of *occlusion* or *connectivity*.

From the type of adjacency relationships, it is possible to generate hypotheses about objects. We do this by looking at each triplet (S_i, S_j, b_k) of surface patches S_i and S_j connected by boundary b with the label k . Whenever k is a convex crease, we directly conclude that S_i and S_j belong to the same object. Such a strong conclusion, however, cannot be made about the other junction types. We have chosen to compute the possibility p that two patches belong to the same object. This number p , between 0 and 1, encodes our belief that two patches belong to the same object. p is 1 for a convex crease, 0.75 for a concave crease, and between 0 and 0.5 for a limb or jump, depending on the distance between the two patches in the vicinity of this boundary. The precise value in the last case is computed as follows: let D be the smallest distance between the two patches S_i and S_j , along the common boundary b_k (for simplicity, we compute the distance along the line of sight, rather than the true Euclidean distance). Let D' be the "thickness" of the occluding patch along b_k . Then the possibility of connection is given by $0.5 \times (1 - \min(1, D/D'))$.

When two nodes are connected by more than one type of boundary, the value of the link between them is the maximum of the individual values (an example of this can be seen in Fig. 1(c) where regions 3 and 5 are linked by a concave crease and a jump boundary).

Finally, the links with p less than some threshold (0.3 for all of our examples) are removed, which means the two nodes are considered not likely to belong to the same object. Thus, we obtain a partition of the original graph into a set of subgraphs with no links between them, each subgraph representing one (partial) object. Figs. 3 and 4 show the results of object inference on the previous images where (a) shows the inferred objects in which different objects are represented by different textures, (b) shows the graph where circles represent the nodes whose numbers refer to the patch numbers shown in the result of segmentation, and lines represent links with a value corresponding to the connection possibility p . It should be noted that the grouping may not be perfect, as is the case in Fig. 4 where the right wing of the plane is inferred as

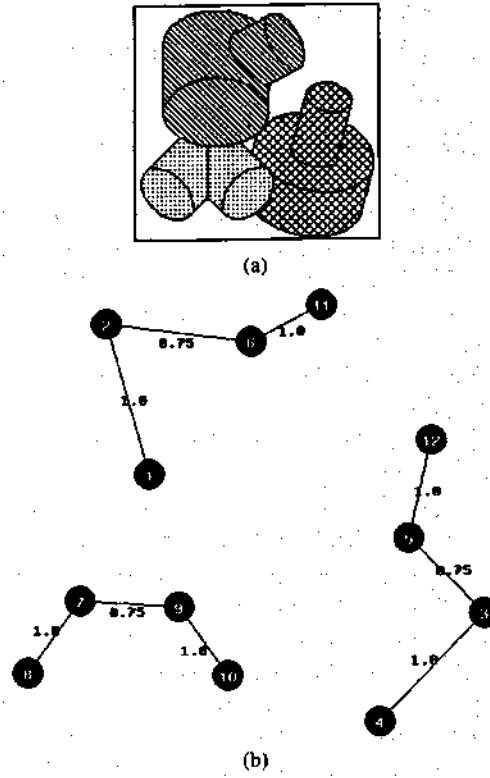


Fig. 3. Objects and graphs for the synthetic range image: (a) inferred objects, (b) graphs.

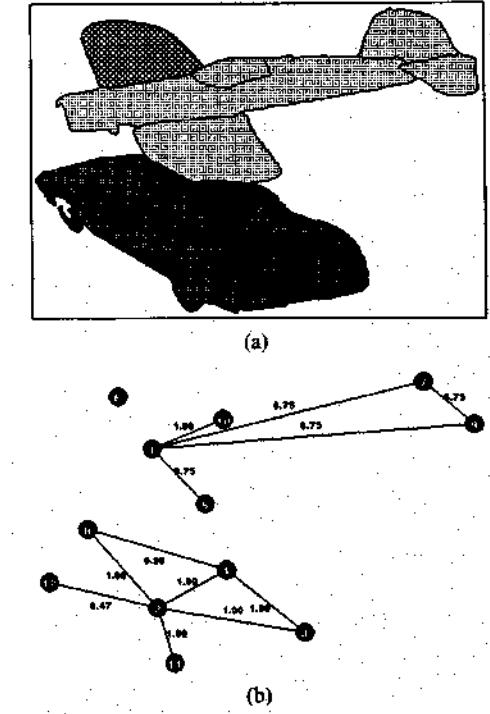


Fig. 4. Objects and graphs for the real range image: (a) inferred objects, (b) graphs.

a separate object because of occlusion. Our matching system needs to handle such errors in description.

C. Representing Objects by Attributed Graphs

Each object is represented by an *attributed* graph in which a node n_i contains the geometric information about

the corresponding surface patch, and a link I_{ij} between connected nodes n_i and n_j represents the relationships between them. The choice of these attributes is largely intuitive. However, their descriptive power and robustness has been demonstrated by testing on a large number of examples with no changes to the parameter values. Some of the attributes (such as the averages of the principal curvatures, given below) are global and could change with varying amounts of occlusion. However, as explained later, these attributes are used only to select a subset of likely models in the matching process; recognition itself involves evaluating a geometrical transformation between the data and the model and is not directly dependent on these gross measures. The attributes we use are described below.

1) Node Attributes: For each visible surface patch n_i , we compute the following:

- **Visible area $A(i)$:** This is the 3-D area of the visible surface patch. It is computed from the image of the patch in 2-D and the knowledge of surface normal at each point of the patch, by making a piece-wise planar approximation of each pixel. The 3-D area is given by $\sum_j 1/\cos(\alpha_j)$, where α_j is the angle between the surface normal and the line of sight at the j th pixel and the sum is over all pixels in the image region corresponding to the surface.

- **Orientation $\vec{n}(i)$:**

- For planar surfaces: the direction of the normal.
- For cylindrical or conic surfaces: the direction of the axis.
- For other surfaces: the direction of least curvature, which is defined as follows: let $\kappa_\theta(p)$ denote the curvature at point p in direction θ , and let $\kappa_\theta = \sum_{p \in n_i} |\kappa_\theta(p)|$. Then the orientation is chosen as the direction θ for which κ_θ is the smallest, the direction along which the patch is the least curved.

- **Average of Principal Curvatures $K_1(i)$ and $K_2(i)$:** $K_1(i)$ is the average of the maximum curvature at each point of the patch n_i ; similarly $K_2(i)$ is the average of the minimum curvature at each point. Note that these are not the same as the "mean" curvatures commonly used in differential geometry.

- For planar surfaces: $K_1(i) = K_2(i) = 0$.
- For other surfaces: let $\vec{n}(i)$ represent the orientation of the surface, and V_1 the projection of $\vec{n}(i)$ on the x - y -plane. Let V_2 be the vector on the x - y -plane perpendicular to V_1 . Then $K_1(i)$ is the average curvature at every point of the surface patch along the direction of V_1 and $K_2(i)$ that of V_2 .

Hence $K_1(i)$ and $K_2(i)$ reflect the flatness of the patch.

- **Estimated Ratio of Occlusion $R(i)$:** A surface patch n_i is said to be occluded by another surface patch n_j if there exists a limb or jump contour c_{ij} between n_i and n_j where the depth value of n_i in the vicinity of c_{ij} is less than that of n_j (note that depth is encoded such that the higher the value, the closer the point is to the viewer). Let $L_{\text{occ}}(i)$ denote the total length of set of boundaries of n_i that are occluded by other patches, and $L_{\text{tot}}(i)$ be the total length

of the boundaries of n_i , then the estimated ratio of occlusion R of n_i is equal to $L_{\text{occ}}(i)/L_{\text{tot}}(i)$.

- **Centroid $\bar{C}(i)$:** The centroid of n_i is given by the average of the (3-D) coordinates of the visible surface points belonging to n_i .

2) Link Attributes: For each pair of nodes n_i and n_j connected by at least one boundary, the link I_{ij} expresses the following.

- **The Type of Adjacency $t(i, j)$:** The adjacency between n_i and n_j can be any combination or none of the following:

- n_i is occluded by n_j at a jump or a limb,
- n_i is occluded by n_j at a jump or a limb,
- n_i and n_j are connected by a convex crease,
- n_i and n_j are connected by a concave crease.

- **The Connection Possibility $p(i, j)$ as Given Earlier:**

- $p = 1$ if n_i and n_j are connected by a convex crease,
- otherwise,
- $p = 0.75$ if n_i and n_j are connected by a concave crease, otherwise;
- $0.3 \leq p \leq 0.5$ if n_i occludes (is occluded by) n_j .

These attributes that we associate to nodes and links are the ones which we found most appropriate for establishing correspondences.

IV. OBJECT RECOGNITION

Matching between the objects in a scene and the database of the models is performed by three modules: the *screener*, in which we find the most likely candidate views for each object, the *graph matcher*, which performs a detailed comparison between the potential matching graphs and computes the 3-D transformation between them, and the *analyzer*, which takes a critical look at the results and proposes to split and merge object graphs. Since the matching program is rather intricate, we start by giving an overview of the core ideas, then describe each module in detail. Finally, we present a detailed case study on one of our test scenes in Section V.

A. Overview of the Matching Process

We have decided to use multiview surface models in our recognition system. Each model consists of several views (2–6, chosen by us based on the complexity of the object). These views are taken so that most of the significant surfaces of the model objects are contained in at least one of these views.

We want to recognize objects in occluded scenes. Each scene S_j may contain multiple (unknown) objects $S_j^1, S_j^2, \dots, S_j^{N_j}$ with self and mutual occlusion. The scene is processed and described using the method presented in the previous section. Each scene object S_j^k is represented by an attributed graph.

We have access to a database of several known objects, called *models*, M_1, M_2, \dots, M_N . Each model M_i consists of several views $M_i^1, M_i^2, \dots, M_i^{N_i}$, and each view is represented by an attributed graph computed as before.

The goal of the matching process is to find, for each object in the scene, the most similar model view. The ob-

ject is then recognized as the model which contains that view.

The matching process contains three major blocks: the *screener*, the *graph matcher*, and the *analyzer*. The top-level block diagrams of these three modules are shown in Figs. 5, 6, and 7, respectively.

- **Screener:** This module serves to find the likely model view candidates for each object in the scene. It is a fast search involving the computation of coarse differences in properties of the nodes of the graphs. The output is an ordered list (with at most 5 elements) of candidates. The details are given in Section IV-B.

- **Graph Matcher:** Once this list of candidates has been computed for each scene object, we perform an extensive comparison between the graphs representing the model view and the object, until one match is found to be “good enough” (defined later), or there are no candidates left.

The strongest constraint imposed by the matching process involves the geometric transform (rotation and translation) of a rigid object. However, this constraint cannot be applied until *after* a candidate match has been computed! As a result, we rely on weaker, partial constraints, and on an incremental estimate of the true transform.

The graph matching procedure consists of finding the pairs (*model node, object node*) forming the largest set consistent with a single rigid 3-D transform. We begin by finding all the possible pairs $\langle m, s \rangle$ where m and s are the model and object nodes, respectively. Then we compare the attributes of these nodes. For instance, a planar patch in the model cannot correspond to a cylindrical patch in the object. Once we have these pairs, we incrementally group them into sets consistent with a 3-D transform. This is done by tree search, updating the transformation as we go. The details of this process are given in Section IV-C.

- **Analyzer:** The previous steps can be improved with critical feedback. Two possible deficiencies of previous steps are:

- The segmentation of the scene graph may not be perfect, especially if objects are close together.
- The heuristic arguments invoked during matching may have led to wrong pairings or discarded valid ones.

There is a module in our system, called the *analyzer*, which tries to detect and correct such errors by merging unmatched objects with existing matches, or by splitting objects with many unmatched nodes into smaller objects. The details are given in Section IV-D.

B. Module 1: Screener

In principle, the number of model views and scene objects may be large, and evaluating each pair to find possible correspondences would be prohibitively expensive. Instead, we use a heuristic method to order the model views for every scene object S_j , according to coarse differences between S_j and these views, and use only the highly ranked views for detailed matching. This process significantly reduces searching time.

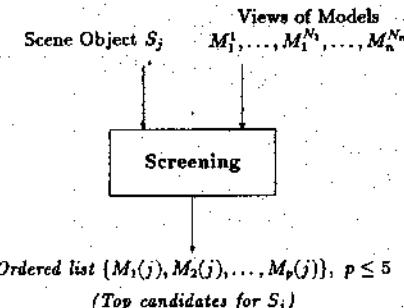


Fig. 5. Block diagram for the screener.

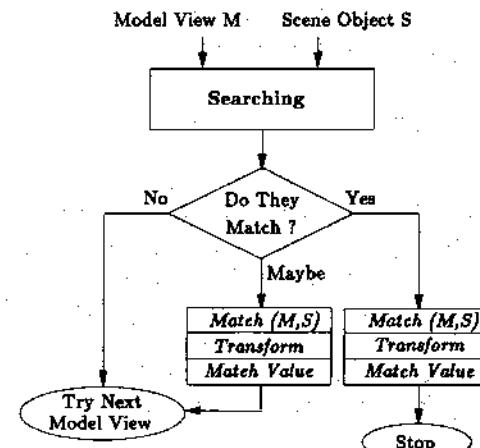


Fig. 6. Block diagram for the graph matcher.

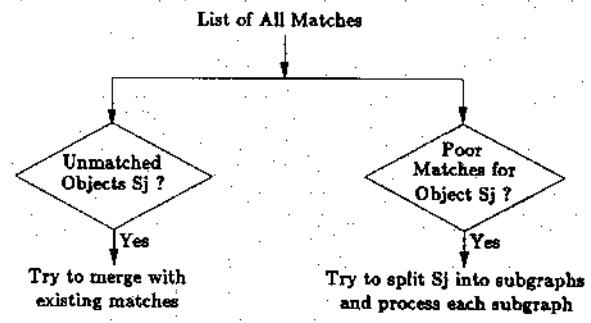


Fig. 7. Block diagram for the analyzer.

To measure the difference between two graphs, we first introduce a normalized measure between 0 and 1 as follows:

$$d(a, b) = \frac{|a - b|}{\max(a, b)} \quad (1)$$

The following *differences* are computed between each model view M and scene object S :

- **Number of Nodes:** Let $N(M, S) = d(N_M, N_S)$ where N_M and N_S denote the number of nodes in M and S , respectively.

- **Number of Planar Nodes (Surface Patches):** Let $R(M, S) = d(R_M, R_S)$ where R_M and R_S denote the number of planar nodes in M and S , respectively.

- **The Visible 3-D Area of the Largest Node:** Let $A(M, S) = d(A_M, A_S)$ where A_M and A_S denote the visible 3-D area of the largest nodes in M and S , respectively.

If any of the following conditions occurs, \mathcal{M} and S are considered not similar enough and the model view \mathcal{M} is discarded:

- $N(\mathcal{M}, S) > 40$ percent
- $R(\mathcal{M}, S) > 30$ percent
- $A(\mathcal{M}, S) > 30$ percent.

If they pass the above test, the *difference* between \mathcal{M} and S is measured by

$$\alpha(\mathcal{M}, S) = N(\mathcal{M}, S) + R(\mathcal{M}, S) + A(\mathcal{M}, S). \quad (2)$$

A model-view \mathcal{M}_i is said to be more similar to scene object S than model view \mathcal{M}_j if $\alpha(\mathcal{M}_i, S)$ is smaller than $\alpha(\mathcal{M}_j, S)$.

The output of the screening module is an ordered list $P(S) = \{M_1, M_2, \dots, M_N\}$ of model views such that $\alpha(\mathcal{M}_i, S) \leq \alpha(\mathcal{M}_j, S)$ for all $i < j$. In addition, we impose the restriction that $N \leq 5$.

C. Module 2: Graph Matcher

The purpose of this module is to find, from a small list of candidate model views produced by the screening module, the most likely view, if any, to correspond to a given scene object. The problem is therefore to find the largest subgraph in the model view for which every node maps onto a node of the object graph according to the geometric operations which transform the view onto the object.

Exhaustive search on all possible sets of pairs is an exponential process, so we perform a two-stage depth-first exploration on this tree, whose block diagram is shown in Fig. 8.

a) *Computing All The Possible Pairs*: For each pair (m, s) , where m and s are model and scene nodes, respectively, we check whether or not they are compatible (according to relation ξ_0 defined later), and if they are, we assign a measure of goodness to this pair.

b) *Search Stage 1*: We try to incrementally build a set with 4 pairs. This is done by depth-first tree search: we expand, that is, we try to add a pair to, the (ordered) pairs of this tree in a depth-first manner until a "good enough" set of pairs is found. Since we cannot enforce the compatibility relation based on the true 3-D transform, we rely instead on weaker constraints ($\xi_1 - \xi_6$ as defined later) and on the current estimate of the true transform (ξ_7). We associate a measure of goodness with each set; if a set of size 4 has a high enough measure, the search terminates, otherwise it continues. This restriction to four pairs is imposed in order to focus the search on the most promising paths only.

c) *Search Stage 2*: We now look for the largest set containing the *best* set found in the previous step. The search tree is expanded from that set only, in depth-first fashion, using the same compatibility constraints as before. It may appear that the search is too focused, and that we may miss promising nodes at earlier stages, but one should remember that we are interested in all the pairs in the path from the root to a leaf, regardless of their order. Therefore, if an unexpanded pair at an earlier level of the tree

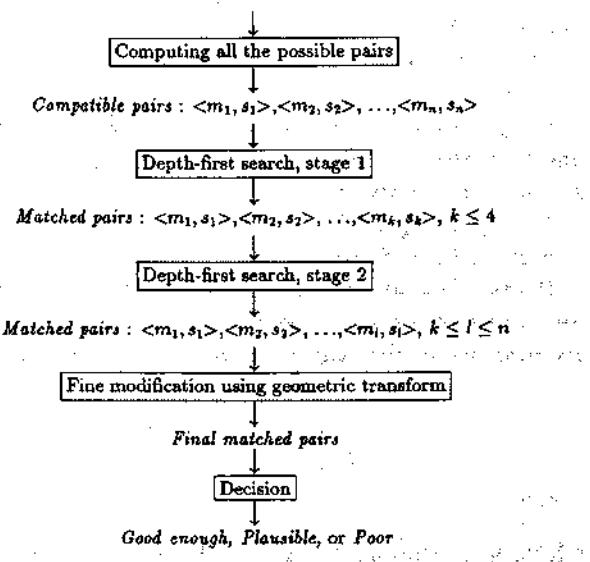


Fig. 8. Block diagram for the graph matcher.

is compatible with the expanded path, it will eventually be appended to it.

In both stage 1 and stage 2, the search terminates if one of the following conditions occurs.

i) The current path is "good enough" (defined precisely in Section IV-C-5).

ii) There is no more possible expansion.

iii) The number of expanded nodes is large enough.

d) *Fine Modification*: We now have a good estimate of the actual transform, and can therefore enforce a strong compatibility constraint between pairs in a set of matches. As a result, we may include pairs that were rejected by the approximate constraints, or reject pairs that were included. We update the transform and the measure of goodness after such modifications.

e) *Decision*: We have two thresholds, \mathcal{J}_1 and \mathcal{J}_2 , with $\mathcal{J}_1 < \mathcal{J}_2$, on the measure of goodness \mathcal{J} of the maximal set of matches.

- If $\mathcal{J} < \mathcal{J}_1$, the object is considered not to correspond to the model.
- If $\mathcal{J} \geq \mathcal{J}_2$, the match is considered good, the object matches the model, and the search stops.
- If $\mathcal{J}_1 \leq \mathcal{J} < \mathcal{J}_2$, the match is considered plausible, but the next candidate model views are also evaluated. Finally, the match with the highest \mathcal{J} is selected.

We now give in detail the definition of our compatibility relationships, our similarity measures, and the fine modification procedure.

1) *Compatibility Between Nodes of Model View and Scene Graph (ξ_0)*: In measuring the similarity between two nodes m of the model view and s of the scene object, we first compute the normalized measure (1) of the difference for each of the following properties:

a) $d_{m,s}(1) = d(A_m, A_s)$, where A_m and A_s represent the 3-D visible area of m and s , respectively.

b) $d_{m,s}(2) = d(K_m, K_s)$, where K represents the average curvature K_1 .

c) $d_{m,s}(3) = d(k_m, k_s)$, where k represents the average curvature K_2 .

The nodes m and s are said to be ξ_0 -compatible if and only if:

- $d_{m,s}(1) < 0.30 + 0.70 \times \max(R_m, R_s)$, where R_m and R_s represent the estimated ratios of occlusion of nodes m and s , respectively.
- $d_{m,s}(2) < 0.30$.
- $d_{m,s}(3) < 0.30$.

If any of the above criteria does not hold, the two nodes are considered not ξ_0 -compatible, otherwise, the *similarity measure* of the two nodes m and s is defined by

$$d_{m,s} = \frac{\sum_{i=1}^3 w_i d_{m,s}(i)}{\sum_{i=1}^3 w_i} \quad (3)$$

where w_i represents the weight for each item $D_{m,s}(i)$. In our experiments, we chose $w_2 = 2$, and $w_1 = w_3 = 1$.

At the end of the above process, all the ξ_0 -compatible pairs are ordered according to their similarity measure. Then, in the graph matching module, the pairs are examined according to this order.

2) *Compatibility Between Two Pairs of Matching Nodes*: Everytime a pair of nodes $\langle m_i, s_i \rangle$ is selected, it is compared to all the already matched pairs $\langle m_j, s_j \rangle$ using a compatibility constraint. If this constraint is not satisfied, the chosen pair $\langle m_i, s_i \rangle$ is discarded. The constraint contains the following *consistency checks*.

a) *Uniqueness Consistency (ξ_1)*: $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$ are said to be ξ_1 -compatible if and only if $m_i \neq m_j$ and $s_i \neq s_j$.

b) *Connection Consistency (ξ_2)*: Let l_1 and l_2 represent the links between m_i and m_j and between s_i and s_j , respectively. Then the types (i.e., limb, jump, convex, or concave) of l_1 and l_2 , denoted as t_1 and t_2 , respectively, are said to be ξ_2 -compatible if and only if one of the following satisfies:

- t_1 is equal to t_2 , or
- one of them is a jump and the other one is a convex crease (a change of view point may transform one into the other)
- either one or both of them is NULL (the nodes are not adjacent, this is possible especially when shadows occur).

Note that l_1 and l_2 may have multiple types, as explained in Section III-B earlier. In this case, we require that *any* of the multiple types match.

c) *Direction Consistency (ξ_3)*: Let θ_1 and θ_2 denote the angles between the orientation of $\langle m_i, m_j \rangle$ and $\langle s_i, s_j \rangle$, and let $\theta = |\theta_1 - \theta_2|$, then the pairs $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$ are said to be ξ_3 -compatible if and only if θ is less than a fixed threshold θ_{tol} . Here we choose $\theta_{tol} = 25^\circ$.

d) *Distance Consistency (ξ_4)*: Let L_1 and L_2 denote the distance between the centroid of inertia of m_i and m_j , and s_i and s_j , respectively. Let

$$L = \frac{|L_1 - L_2|}{\max(L_1, L_2)}, \quad (4)$$

then, the pairs $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$ are said to be ξ_4 -compatible if and only if L is less than a threshold. Here we choose the threshold as 0.30.

e) *3-D Geometry Consistency (ξ_5)*: For all the matched pairs $\langle m_k, s_k \rangle$ other than $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$, let \bar{U}_{ij} , \bar{V}_{ij} , \bar{U}_{ik} , and \bar{V}_{ik} represent the vector connecting the centroid of m_i to that of m_j , s_i to s_j , m_i to m_k , and s_i to s_k , respectively. Let θ_1 and θ_2 denote the *directed angle* from \bar{U}_{ij} to \bar{U}_{ik} , and from \bar{V}_{ij} to \bar{V}_{ik} , respectively. And let

$$\theta = |\theta_1 - \theta_2|. \quad (5)$$

Then the three pairs $\langle m_i, s_i \rangle$, $\langle m_j, s_j \rangle$, and $\langle m_k, s_k \rangle$ are said to be ξ_5 compatible if and only if θ is less than a threshold. Here we choose the threshold to be 25° . This consistency check is mainly used to remove a match between two objects which are geometrically similar by mirror effect, as shown in Fig. 9.

If these four consistency conditions are fulfilled, we say that $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$ are mutually consistent. This only happens, however, when no abrupt changes occur as a result of a difference in viewing angle. To take into account such changes, we define an additional condition as follows.

• *Enclosure (ξ_6)*: To determine whether a surface patch m is (*partially*) *enclosed* in 2-D by another surface patch s , we start from the center point $C = (x, y)$ of m , where x and y are the first two coordinates of the centroid of m , and search outward in 8 directions, each 45° apart, if 6 or more out of the 8 searches encounter any point in surface s , we say m is enclosed by s . With this definition, we accept two pairs even if they fail all of the consistency conditions above except for condition 1, as long as m_i encloses (is enclosed by) m_j and s_i encloses (is enclosed by) s_j .

The motivation for this constraint is to allow looser matching of smaller regions contained inside larger and already matched regions (such as pushbuttons on the telephone in one of the examples shown later). The constraints ξ_2 – ξ_5 tend to be not reliable in such cases. Note that the matches accepted at this stage must still agree with the geometric transformation computed later.

It should be noted that the above criteria are not perfect, especially when parts of the objects are occluded, and mostly serve to prune the search tree. The true compatibility between nodes is established by the computation of the geometric transform.

3) *Computing the Geometric Transform*: Computing the geometric transform between matched objects not only indicates how to bring matched objects in correspondence, but also helps to verify the matching process. If the error in the transform for the current partial match is too large, the match should be abandoned.

The matching process gives the correspondences between surfaces of model view \mathcal{M} and scene object \mathcal{S} . By extracting the orientation and location of the corresponding surfaces, we can compute the (geometric) transform which brings \mathcal{M} in registration with \mathcal{S} . Here we introduce a noniterative method in which the axis of the rotation is first computed using the orientation of the matched nodes,

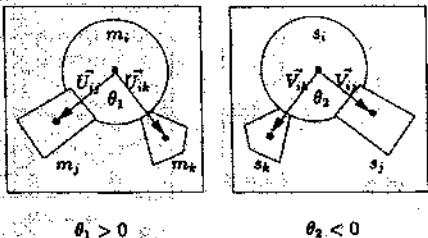


Fig. 9. Geometric similarity by mirror effect.

then the angle of the rotation is obtained using a two-level search. Finally, the translation is computed from the centroids of the matched nodes, using a least-square method. Since this method is noniterative, we believe it is faster than iterative methods which are used by many researchers [14], [17]. The details of the geometric transform computation are given below.

The rigid transform between objects can be represented in homogeneous coordinates by a 4×4 matrix as follows:

$$\begin{pmatrix} R_{(3 \times 3)} & T_{(3 \times 1)} \\ 0_{(1 \times 3)} & 1 \end{pmatrix} \quad (6)$$

where R represents the rotation and T the translation, respectively. A rotation can be represented by a rotation of angle θ about a unit vector k (axis of revolution) located at the origin. Let k_x , k_y , and k_z represent the three components of the axis k ; then the relation between the coefficients of R and θ , k_x , k_y , k_z can be expressed as follows [27]:

$$R(k, \theta) = \begin{pmatrix} k_x k_x v + c & k_y k_x v - k_z s & k_z k_x v + k_y s \\ k_x k_y v + k_z s & k_y k_y v + c & k_z k_y v - k_x s \\ k_x k_z v - k_y s & k_y k_z v + k_x s & k_z k_z v + c \end{pmatrix} \quad (7)$$

where $s = \sin \theta$, $c = \cos \theta$, and $v = 1 - c$.

To find the axis k , the orientation vectors for each matched node pairs are first retrieved. Let $\langle m_i, s_i \rangle$, $i = 1, \dots, N$ be the matched node pairs between two objects M and S where $m_i \in M$, $s_i \in S$, and N is the number of matched node pairs, respectively. Let $\langle P_i, Q_i \rangle$ denote the orientation vectors of the matched node pair $\langle m_i, s_i \rangle$, as shown in Fig. 10 for $i = 1, 2$, where both orientation vectors have been brought to the origin O . Assuming the rotated angle is $\theta = 2\alpha$ as indicated in Fig. 10, it is easy to show that the axis of revolution k for P_1 and Q_1 must lie on the plane L_1 that bisects the angle $P_1 O Q_1$. In other words, the angle between P_1 and its projection on L_1 is equal to that between Q_1 and L_1 , and this angle is equal to α (unless P_1 and Q_1 coincide at k , in this case, however, L_1 is an arbitrary plane that contains k). The same reasoning can be applied to all the other $\langle P_i, Q_i \rangle$ pairs. Thus to find the axis k , we only have to find all the planes L_i , then k is at their intersections. In our implementation, we find all the intersections k_{ij} of each two pairs L_i and L_j , then for each k_{ij} we compute the sum of the angle be-

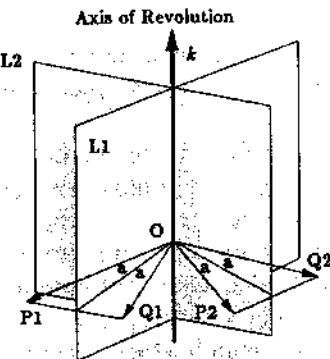


Fig. 10. Computing transforms.

tween all other k_{ij} , the one with the least sum is our choice of axis k .

The angle θ is found as follows. Let $\Theta(a, b)$ denote the angle between vector a and b . Given a rotation θ , for each matched pair $\langle m_i, s_i \rangle$, $\Theta(R(\theta)P_i, Q_i)$ is computed where $R(\theta)P_i$ represent the vector of P_i after rotating it by an angle θ . Then the best θ is found by minimizing the following equation:

$$\theta = \sum_i \Theta(R(\theta)P_i, Q_i). \quad (8)$$

After the rotation matrix R is computed, the translation T can be obtained easily. Let $\langle E_i, C_i \rangle$ denote the centroids of each matched pair $\langle m_i, s_i \rangle$ and $C'_i = R(\theta)E_i$. Then the translation T is as follows:

$$\frac{1}{N} \sum_i C_i - C'_i. \quad (9)$$

In our experiments, we have found that this computation gives highly accurate results; an example is shown later (in Section V-B).

4) Modifications Based on the Geometric Transform: Since the compatibility constraints presented previously are not perfect, some corresponding nodes may not have fulfilled the conditions imposed and may therefore have been rejected. Using the transform for the match allows us to rectify this situation. To achieve this, the transform between the model view and the scene object is computed first, using the current match L , then the model object is transformed and *superimposed* on the scene. Each surface s of the scene object is then checked by the following constraint ξ_7 :

- If s is not yet matched and there exists an unmatched model surface m whose transformed surface m' is close enough to s , then include $\langle m, s \rangle$ into the match L . Surfaces m' and s are said to be close enough if and only if the distance between the centroids of m' and s are less than twice the smaller *average width* of m and s , where the average width W of a surface m is computed as follows:

$$W = \frac{w_1 + 3w_2}{4} \quad (10)$$

where w_1 and w_2 indicate the distance from the 2-D center of the surface m to its farthest and closest boundaries, respectively.

- If s is a matched node and the corresponding transformed model node m' is not close enough, then $\langle m, s \rangle$ is removed from the match L .

After the modification, the transform is recomputed.

5) *Measuring the Goodness of a Match:* Throughout the graph matching procedure, a *match value* \mathcal{H} is attached to the current match which is computed as follows:

$$\mathcal{H} = \frac{\max(N_m, N_s) + \max(A_m, A_s)}{2} \quad (11)$$

where N_m , N_s , A_m , and A_s represent the ratio between the number of matched model nodes and the number of total model nodes, the number of matched scene nodes and the number of total scene nodes, the 3-D area of matched model nodes and that of all the model nodes, and the 3-D area of matched scene nodes and that of all the scene nodes, respectively. The highest value of \mathcal{H} is 1, which represents a complete match, and the lowest value of \mathcal{H} is 0, which means that nothing is matched. A match is considered "good enough" when \mathcal{H} reaches $\mathcal{H}_2 = 0.80$, and the graph matching step is immediately terminated. Otherwise, the search continues until the search tree can no longer be expanded. Finally, if the match value \mathcal{H} of the resulting match is less than $\mathcal{H}_1 = 0.60$, the match is discarded.

D. Module 3: Analyzer

Input scenes may contain multiple objects, and, as we have mentioned before, the object inference step may not produce perfect results. For example, when two objects touch, it is possible that we would consider them as one object instead of two, i.e., the result of object inference may produce just one *object shell* (i.e., a single graph) for these two touching objects. On the other hand, due to occlusions, shadows, or a special view point, real objects may appear as separate pieces in range image. In this case, more than one *object shell* is generated for different parts of the same object. In both cases, the match between model views and such objects will not be satisfactory. To correct this, a refining process which *splits* and/or *merges* objects according to current matches and their geometric relationships is applied next; we call this module the *analyzer*.

1) *Splitting Objects:* Suppose that our scene contains two objects A and B which are inferred as one object shell (represented by graph S). Since we allow only one model view being matched for one object shell, let us assume that the model view \mathfrak{Q} matches S (as the result after step 2) where \mathfrak{Q} is the view of the model object which is similar to scene object A . We also assume that there is another model view \mathfrak{G} whose model is similar to scene object B . At this time, the match between \mathfrak{G} and B has not yet been found. Fig. 11(a) illustrates this situation, where all surface patches are numbered for convenience. From the figure, we see that patch 3 of the scene is touching

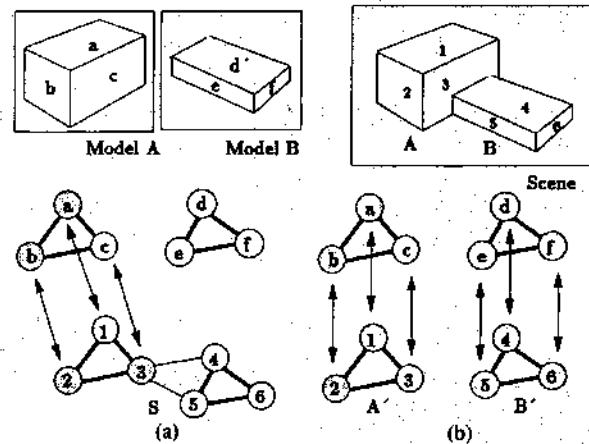


Fig. 11. Splitting objects.

patches 4 and 5. The scene graph S and the model graphs \mathfrak{Q} and \mathfrak{G} are also shown. The shaded nodes represent the currently matched nodes (the correspondences between two matched nodes are indicated by bidirectional arrows). After matching, we find that half of the nodes in S are not matched (surfaces 4, 5, and 6); however, by examining S , it is possible to *split* the two objects in the scene.

The following two rules are followed in splitting objects.

a) R_1 : Splitting can only be achieved by removing links for which p is strictly less than 1.

b) R_2 : The set of already matched patches should not be split into subsets.

By looking at the scene and the graph S , we know that surfaces 1-2, 2-3, 1-3, 4-5, 4-6, and 5-6 are connected by convex creases ($p = 1$) and from Rule 1 they can not be split (displayed by thick lines); however, surfaces 3-4 and 3-5 are connected by concave creases ($p = 0.75$) and may be split (displayed by thin lines). By removing the links 3-4 and 3-5 from S , we obtain two separate graphs \mathfrak{Q}' (which contains surfaces 1, 2, and 3) and \mathfrak{G}' (which contains surfaces 4, 5, and 6). Furthermore, by looking at the match we notice that the *matched surfaces* (surfaces 1, 2, and 3) are contained only in \mathfrak{Q}' , thus the result of splitting does not break Rule 2.

After splitting, we now can match scene object \mathfrak{G}' to model view \mathfrak{G} and the result is shown in Fig. 11(b).

From the above reasoning, we have derived the following procedure to split objects:

a) After graph searching, if there exists a *matched* scene object S such that more than 40 percent of its nodes are not matched, try to split it.

b) Split S according to the two rules R_1 and R_2 stated above. Let S_1 represent the split object which contains all the already matched nodes, and S_2, S_3, \dots, S_N represent the remaining split objects sorted in the descending order of number of containing nodes.

c) *Reconnect* S_2, S_3, \dots, S_N into one object S' , try to match it with model views, if no model views can be matched, remove S_N (which contains the least number of nodes) and retry the match. The idea is that we want to match from the largest possible objects.

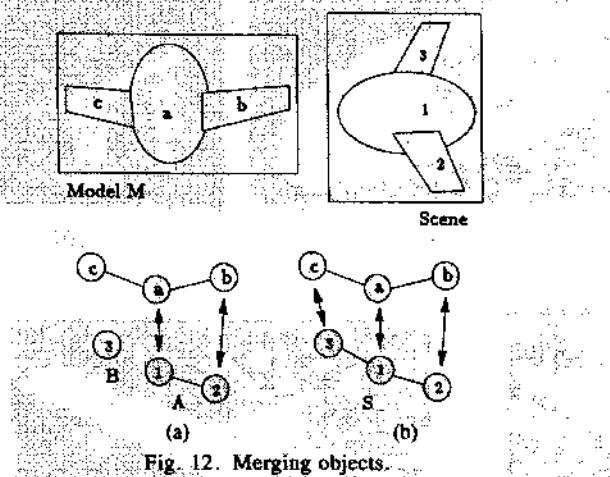


Fig. 12. Merging objects.

- d) Repeat steps a)-c) until no more splitting is achievable.

2) *Merging Objects*: Suppose that our scene contains only one object S ; however, due to self occlusion, it is inferred as two object shells (represented by graphs G and \mathcal{G} , respectively). Also assume that after graph searching, G is matched to a model view M (which is supposed to match the entire object S) and \mathcal{G} is not matched (usually because it contains too few nodes). This is illustrated in Fig. 12(a) where G contains two *matched* surfaces 1 and 2, \mathcal{G} contains one *unmatched* surface 3, and model view M contains three surfaces a , b , and c . In the current match, surfaces a and b match surfaces 1 and 2, respectively, while surface c is not matched (which is supposed to match surface 3 of \mathcal{G}).

To refine the match, merging can be used to bring separate pieces that actually belong to the same object. The idea is that by merging two objects G and \mathcal{G} into one object S (which matches M), not only the similarity constraints between corresponding *matching* nodes ($\langle c, 3 \rangle$ in this example) should be satisfied, but the binary constraints and the transform constraints between each pair of matching nodes ($\langle a, 1 \rangle$, $\langle b, 2 \rangle$, and $\langle c, 3 \rangle$ in this example) should also be satisfied.

In the example we show here, G and \mathcal{G} are merged into S and the new match result is shown in Fig. 12(b).

From the above reasoning, we have derived a general procedure to merge objects as follows:

- a) After graph searching, try to merge any *unmatched* scene object \mathcal{G} .
- b) Select among matched objects the one which is geometrically closest to \mathcal{G} . Let G , M , and L denote the selected object, its matched model view, and the match between G and M , respectively.
- c) Use the graph searching process presented in the previous section to find the correspondences between the nodes in \mathcal{G} and the *unmatched* nodes in M , assuming that the match L has already been established. More matches can be added to L only if the resulting match meets the constraints of node similarity, binary constraints between each pair of matching pixels, and the transformation constraint.

- d) Repeat steps a)-c) until either no more unmatched scene objects exist or no more merging is achievable.

V. EXPERIMENTAL RESULTS

In this section, we show the results and an evaluation of our recognition process, using a database of 10 objects, resulting in 32 views. We first show a detailed case study, then show results on a few more scenes.

A. The Models

In this work, we have selected 10 objects to build models. Then scenes which consist of these objects are acquired and recognized using these models. The model objects include a car, a chair, a telephone, a table, a mask, a hatchback car, a wagon, two boats, and an airplane. In order to save space, we only show *one* view for each of these objects in Fig. 13, but it is important to remember that the database consists of *all* the views for all the objects, 32 in all.

Our data comes from an active range finder using a light-stripe and triangulation technique, it is described in detail in [23]. The system consists of a laser, a video camera, a video monitor, a terminal, and a computer-driven rotary table.

Our composite scenes were obtained by first scanning each object in an arbitrary position and orientation, then combining these objects (synthetically) to generate the scene as it would have appeared if we had scanned it. This is necessary because of the technical difficulty in actually scanning the scene using a rotary table, which would create too many shadow regions.

B. A Detailed Case Study

In order to better understand how the system works, we take one of our test scenes, shown in Fig. 2(a), and follow each step of the system as it processes this scene. Table I shows the computation times for the various steps of processing on a Symbolics 3645 computer (with a floating point accelerator and 1 Megaword of memory). Note that the recognition takes only a small fraction of the total time. This is consistent with our design philosophy that good representations vastly simplify recognition. Also note that most of the time is spent on relatively simple, local computations that can be performed in parallel in a straightforward way.

1) *Search Nodes Expanded in Recognition*: As mentioned before, the object inference cannot be expected to always generate perfect results. In the scene, the right wing of the airplane is segmented into a separate object as the result of object inference, shown in Fig. 14, where different textures represent different objects. Thus, at the beginning of the recognition process, the scene consists of three "objects": the airplane without the right wing, the wagon, and the wing.

The first step of the recognition process is to screen the model views for each of the three objects. The single wing, which consists of only one surface patch, does not find any candidate model views, thus is ignored tempo-

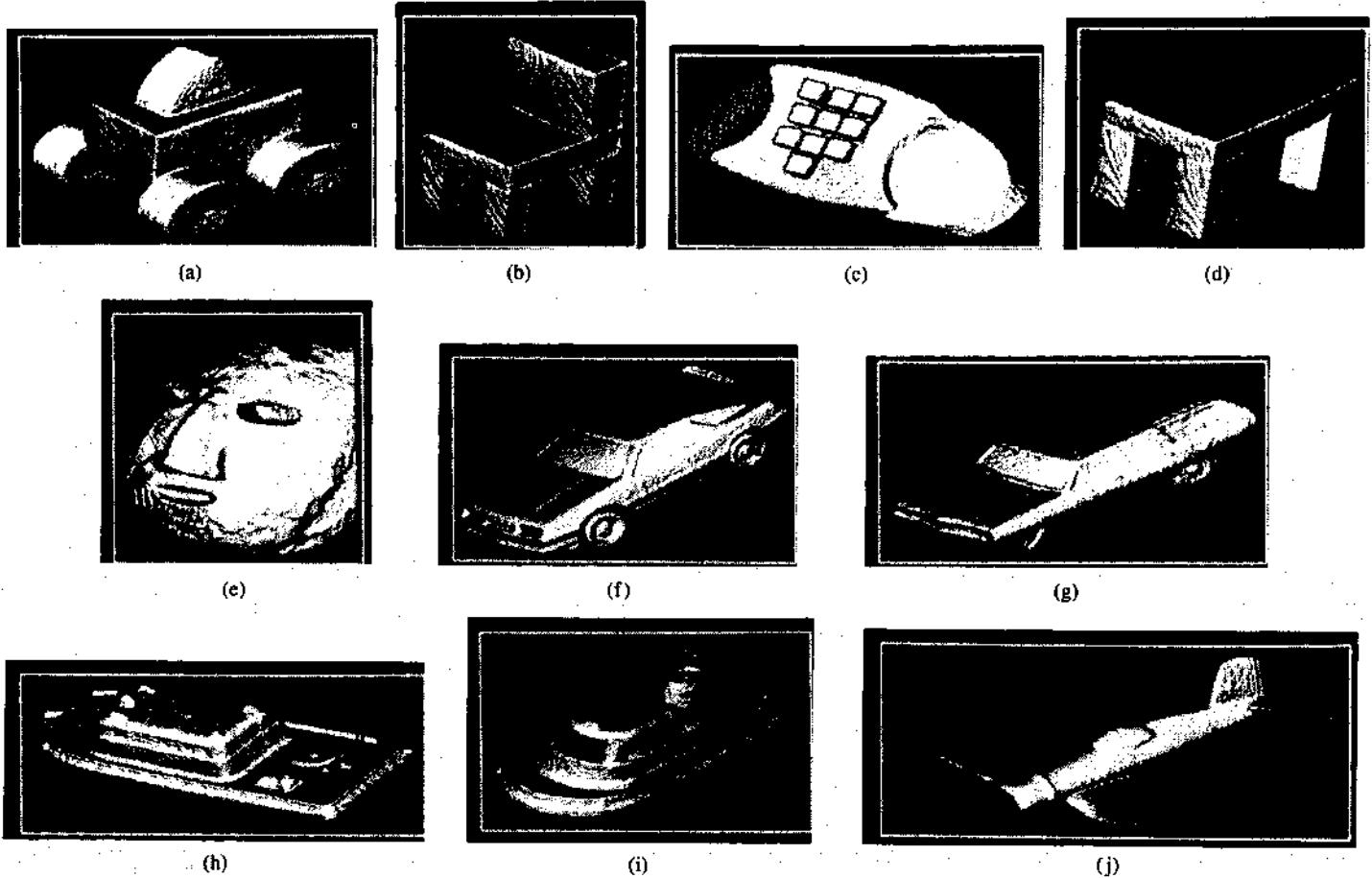


Fig. 13. Model views: (a) model car (4 views), (b) model chair (4 views), (c) model phone (4 views), (d) model table (4 views), (e) model mask (6 views), (f) model hatchback (2 views), (g) model wagon (2 views), (h) model Boat 1 (2 views), (i) model Boat 2 (2 views), (j) model plane (2 views).

TABLE I
DETAILED TIMING INFORMATION FOR THE FIRST SCENE

Step	Overall
CV Time: 693 seconds	17%
Gaussian convolution	50%
$L \circ G$ convolution	50%
SG Time: 433 seconds	10%
Zero-crossing detection	18%
Extrema detection	22%
Space grouping	17%
Surface segmentation	43%
AP Time: 2851 seconds	68%
Planar surface approximation	7%
Quadratic surface approximation	41%
Approximation error computation	52%
GR Time: 37 seconds	1%
Computation of graph	50%
Object inference	50%
RC Time: 147 seconds	4%
Screener (Module 1)	1%
Graph Matcher (Module 2)	92%
Analyzer (Module 3)	7%

CV : Time for Gaussian and LoG convolution
 SG : Time for feature detection and segmentation
 AP : Time for surface approximation
 GR : Time for computation of graph and object inference
 RC : Time for recognition

rarily. The screening results for the other two objects are as follows:

- The wagon: 1) plane, view 1, 2) hatchback, view 2, 3) wagon, view 1, 4) chair, view 2, 5) wagon, view 2.

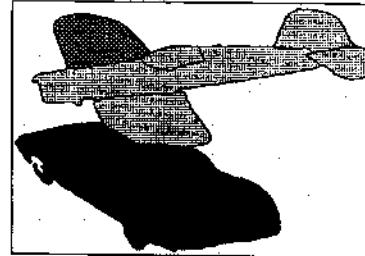


Fig. 14. Result of object inference for the first scene.

- The plane: 1) plane, view 1, 2) plane, view 2, 3) chair, view 1.

The second step examines each of the selected candidates in the order shown above. As the result of graph matching, the first four candidates for the wagon are discarded. In this case study, we show the search trees expanded by matching between the scene object "wagon" and the second view of the model "wagon," and between the scene object "airplane" and the first view of the model "plane," where both models are selected as the final match.

Fig. 15 shows the surface patches of the first view of the model plane, the second view of the model wagon, and the scene, respectively, with each surface patch (node) having a unique number. Figs. 16 and 17 show the search trees in matching the wagon and the airplane, respec-

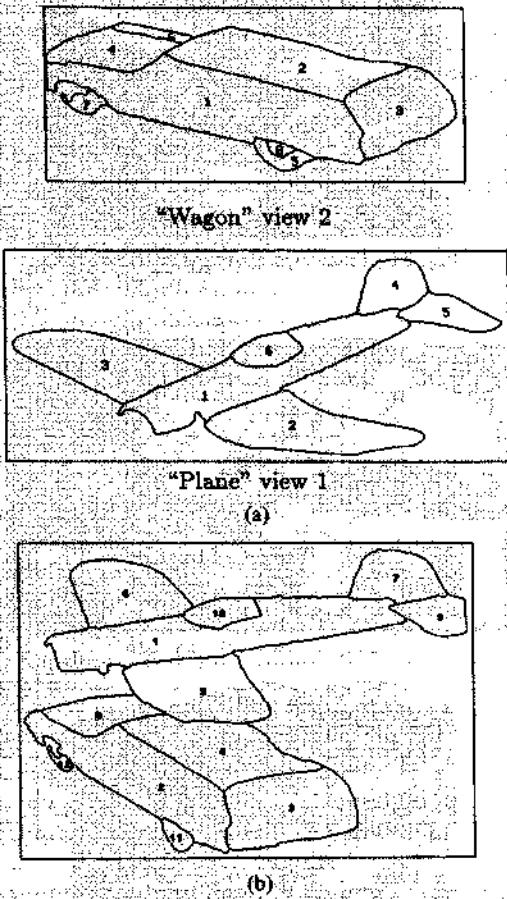


Fig. 15. Node numbers of the models and the first scene: (a) matched model views, (b) matched objects.

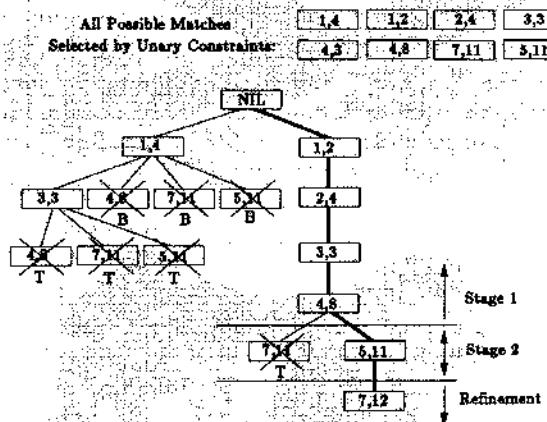


Fig. 16. Search tree for the wagon.

tively. In these figures, each tree node, represented by a pair of numbers inside a box a, b , indicates a possible match between model node a and scene node b . The order of expansion is from left to right, then top to bottom. In Fig. 16, for example, the order of expansion is $[1, 4]$, $[3, 3]$, $[4, 8]$ at level 2, $[7, 11]$ at level 2, $[5, 11]$ at level 2, $[4, 8]$ at level 3, $[7, 11]$ at level 3, $[5, 11]$ at level 3, $[1, 2]$, and so on. A cross over a tree node indicates that this match is rejected, either by binary constraints (marked "B") or by transform constraints (marked

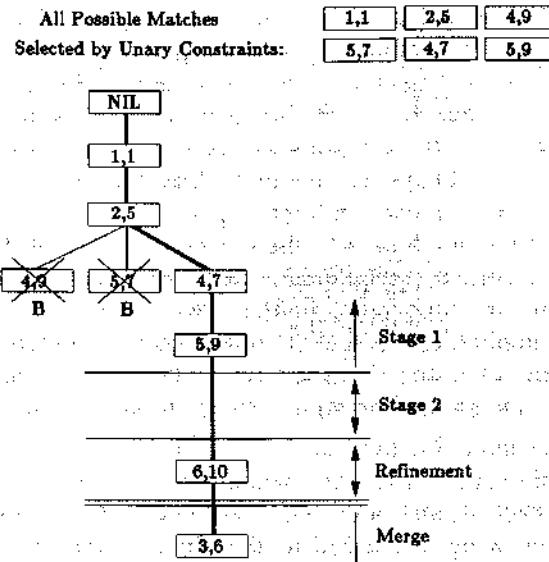


Fig. 17. Search tree for the airplane.

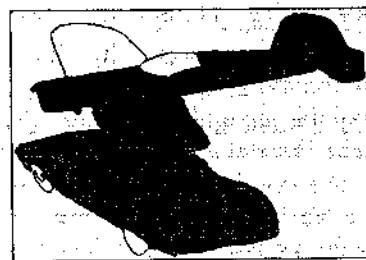


Fig. 18. Nodes expanded in stage 1.

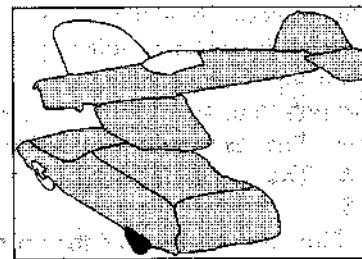


Fig. 19. Nodes expanded in stage 2.

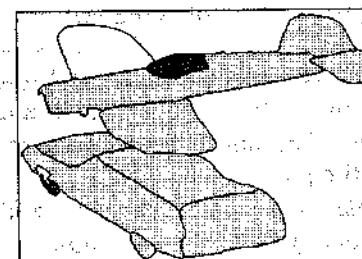


Fig. 20. Nodes expanded after refinement.

"T"). Note that the unary constraints are applied to select all the possible match pairs *before* the search begins. The final selected matches are indicated by thicker links. Figs. 18, 19, and 20 show the scene nodes expanded in

each stage where heavily shaded regions indicate currently expanded patches (nodes) and lightly shaded regions indicate expanded patches at previous stages. Note that we show both scene objects (wagon and airplane) in the same figures, but in fact they are processed in sequence. From these figures we see that the major (larger) surfaces are always selected in the first stage while smaller surfaces are matched in later stages.

After the graph search, the correspondences of the wagon are correctly established while the right wing of the airplane has not been recognized. In the next step, merging is applied to the scenes. Since the wing (scene node 6) is closer to the airplane than to the wagon, and there is only one unmatched node in the matched airplane model (model node 3), the match [3, 6] is thus selected as a possible match. The transform and binary constraints are then checked, and in this case, both of them are satisfied, thus the wing is merged to the airplane and the match [3, 6] is included. It should be noted that to verify the transform constraint, we simply transform the model views of the airplane and *superimpose* it on the scene, which is shown in Fig. 21, and then find that the two nodes [3, 6] are close enough. Fig. 22 shows the expanded node in the merging step.

Fig. 23 shows the recognition results where Fig. 23(a) shows the matched model views and Fig. 23(b) shows the matched scene objects. In this figure, corresponding textures are used to represent corresponding objects, and corresponding numbers are used for matched nodes. Table II summarizes the recognition results. The entries are to be interpreted as follows.

- *Object*: The object in the scene.
- *Model*: The selected model view by the screener. The views are sorted by order of selection.
- *Nodes Expanded*: The number of *search nodes* expanded in the search tree. It is limited to 100.
- *Max. Depth*: The maximum depth of the search tree.
- *Max. Width*: The maximum width of the search tree, it is limited to 5.
- *Decision*: The final decision of the recognition:
 - 1) *Matched*: The object and the model view are considered matched (good enough in Module 2 (Graph matcher), i.e., match value $J\mathcal{C} \geq 0.8$). If this happens, the remaining selected model views are ignored.
 - 2) *Ignored*: The view is ignored because a good-enough match has already been found.
 - 3) *Rejected*: The match is too poor to be accepted ($J\mathcal{C} < 0.6$).
 - 4) *Plausible*: The match between the object and the view is acceptable; however, it is not good enough to ignore remaining candidate views ($0.6 \leq J\mathcal{C} < 0.8$). In this case, the match and its match value $J\mathcal{C}$ are kept for further comparison. If a good-enough match is found during the subsequent exploration, this plausible match is discarded, otherwise, the plausible match with the largest $J\mathcal{C}$ is selected as the final match.

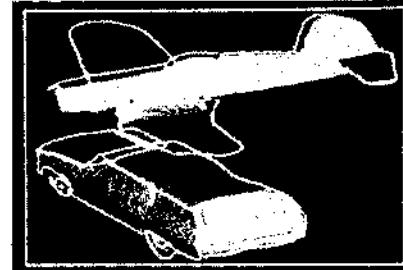


Fig. 21. Transform results of the first scene.

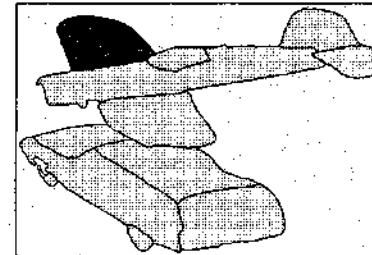
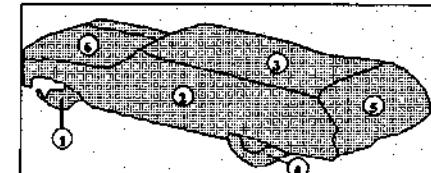
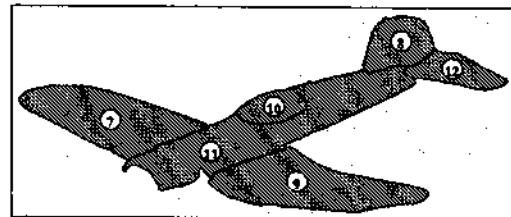


Fig. 22. Nodes expanded after merging.

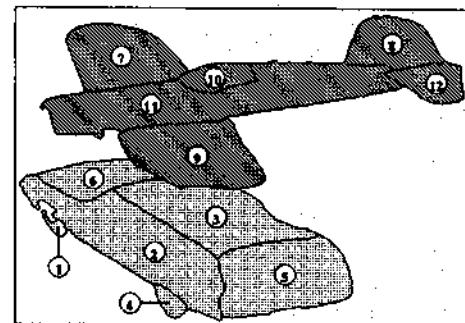


"Wagon" view 2



"Plane" view 1

(a)



(b)

Fig. 23. Result of recognition on the first scene: (a) matched model views, (b) matched objects.

C. Results for Other Scenes

We have performed successful experiments on several such composite scenes. The complete set can be found in [13]; here, we only present results for two other scenes in order to save space. They are shown in Figs. 24 and 25,

TABLE II
SUMMARY FOR THE FIRST SCENE

Object	Model	Expanded Nodes	Max. Depth	Max. Width	Decision
The wagon	plane, view 1	8	2	3	Rejected
	hatchback, view 2	3	2	2	Rejected
	wagon, view 1	4	1	2	Rejected
	chair, view 2	5	2	2	Rejected
The plane	wagon, view 2	15	6	4	Matched
	plane, view 1	7	5	3	Matched
	plane, view 2	-	-	-	Ignored
	chair, view 1	-	-	-	Ignored

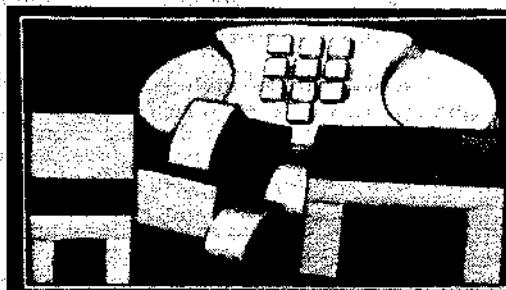


Fig. 24. The second scene.

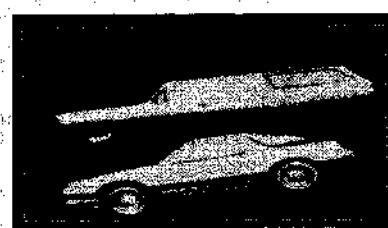


Fig. 25. The third scene.

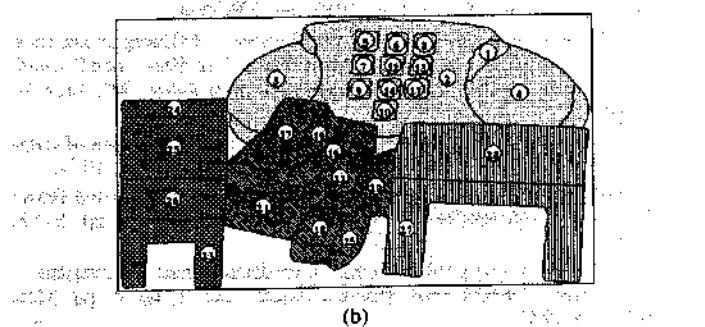
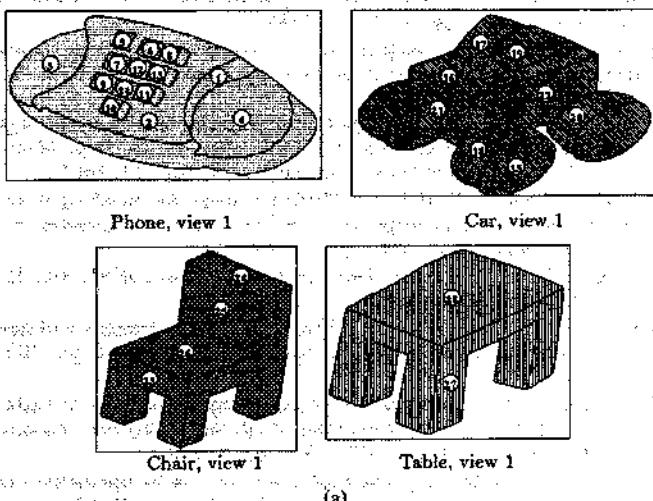
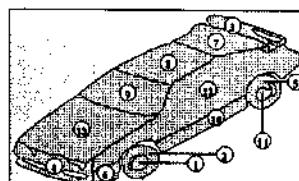
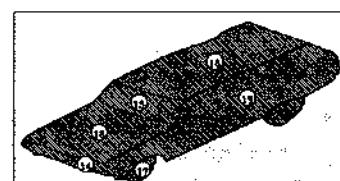


Fig. 26. Result of recognition on the second scene: (a) matched model views, (b) matched objects.

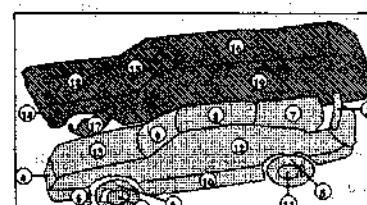


"Hatchback" view 1



"Wagon" view 1

(a)



(b)

Fig. 27. Result of recognition on the third scene: (a) matched model views, (b) matched objects.

TABLE III
SUMMARY FOR THE SECOND SCENE

Object	Model	Expanded Nodes	Max. Depth	Max. Width	Decision
The phone	phone, view 1	20	10	4	Matched
	phone, view 2	-	-	-	Ignored
	chair, view 3	-	-	-	Ignored
	hatchback, view 1	-	-	-	Ignored
The car	hatchback, view 2	5	2	2	Plausible
	plane, view 2	3	2	2	Rejected
	chair, view 1	17	3	3	Plausible
	car, view 1	18	4	5	Matched
	plane, view 1	-	-	-	Ignored
The chair	chair, view 1	7	4	3	Matched
The table	table, view 1	2	2	1	Matched
	mask, view 6	-	-	-	Ignored

TABLE IV
SUMMARY FOR THE THIRD SCENE

Object	Model	Expanded Nodes	Max. Depth	Max. Width	Decision
The hatchback	table, view 2	2	1	2	Rejected
	chair, view 3	14	3	5	Rejected
	hatchback, view 1	15	8	3	Matched
The wagon	wagon, view 1	7	4	2	Matched
	wagon, view 2	-	-	-	Ignored
	chair, view 2	-	-	-	Ignored
	hatchback, view 2	-	-	-	Ignored
	chair, view 1	-	-	-	Ignored

together with the recognition results shown in Figs. 26 and 27. Tables III and IV summarize the recognition process.

It should be noted that if a good-enough match cannot be found after graph matching, further analysis may be necessary to find the best match from the plausible matches. Several possibilities include increasing the resolution, focusing on one part, or changing viewing angles.

Several conclusions can be drawn from the following results.

- Most of the correct matches are found at the first or second selected views, this proves that our screening is powerful.

- Whenever a correct view is selected, the number of expanded nodes is small. This proves that our graph matching is efficient.
- The inferred objects do correspond to the physical objects, which indicates that our object inference and splitting/merging methods are powerful.

VI. CONCLUSION

This paper provides a complete method to describe and recognize 3-D objects, using the surface information of these objects. The key contributions are summarized as follows.

• *It provides a complete system to describe and recognize 3-D objects.* We present a set of techniques to retrieve significant features of 3-D objects, describe them, match the descriptions, build the models in multi-views, and finally recognize them.

• *It is data-driven in that no a priori scene knowledge is required.* The descriptions of the objects are computed without any knowledge about existing models, which is important when the environment is unknown, or the number of interesting objects is large.

• *Moderately complex objects can be well described and matched.* A large number of different objects were tested and good results were obtained. These objects vary in sizes, shape, and complexity. Some of them are highly symmetric while others are different from every view point. This shows the generality and robustness of our method.

• *Partially occluded objects can be well described and matched.* Occlusion is one of the major problems in computer vision. The method presented here tackles this problem in a very effective way. More than two thirds of the surfaces can be heavily occluded or completely missing and a plausible match is still achievable.

Our system can be improved in several ways:

- Better segmentation methods will help. Specifically, additional methods for segmenting "smooth" surfaces are needed.
- Under heavy occlusion, it may be necessary to further group fragmented surfaces before matching is attempted.
- It would be nice to have a single, 3-D volumetric representation of the models, rather than the multiple views. The multiple views could then be computed from the single model. Alternately, volume descriptions may be computed from the data. The latter is a difficult problem, but we believe that our surface descriptions are a key step for this process also.

REFERENCES

- [1] N. Ayache, "A model-based vision system to identify and locate partially visible industrial parts," in *Proc. Conf. IEEE Computer Vision and Pattern Recognition*, Washington, DC, 1983, pp. 492-494.
- [2] P. J. Besl and R. C. Jain, "Three-dimensional object recognition," *ACM Comput. Surveys*, vol. 17, no. 1, pp. 75-145, Mar. 1985.
- [3] P. J. Besl and R. C. Jain, "Segmentation through variable-order surface fitting," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 2, pp. 167-192, Mar. 1988.
- [4] B. Bhanu, "Representation and shape matching of 3-D objects," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 3, pp. 340-350, May 1984.
- [5] R. C. Bolles and R. A. Cain, "Recognizing and locating partially visible objects: The local feature-focus method," *Int. J. Robotics Res.*, vol. 1, no. 3, pp. 637-643, 1982.
- [6] R. C. Bolles and P. Horaud, "3DPO: A three-dimensional part orientation system," *Int. J. Robotics Res.*, vol. 5, no. 3, pp. 3-26, Fall 1986.
- [7] M. Brady, "Computational approaches to image understanding," *ACM Comput. Surveys*, vol. 14, no. 1, pp. 3-71, Mar. 1982.
- [8] M. Brady, J. Ponce, A. Yuille, and H. Asada, "Describing surfaces," in H. Hanafusa and H. Inoue, editors, *Proc. 2nd Int. Symp. Robotics Research*, H. Hanafusa and H. Inoue, Eds. Cambridge, MA: M.I.T. Press, 1985.
- [9] R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intell.*, vol. 17, pp. 285-348, 1981.
- [10] —, "Model-based three-dimensional interpretations of two-dimensional images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 2, pp. 140-150, Mar. 1983.
- [11] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Comput. Surveys*, vol. 18, no. 1, pp. 67-108, Mar. 1986.
- [12] T. J. Fan, G. Medioni, and R. Nevatia, "Segmented descriptions of 3-D surfaces," *IEEE J. Robotics Automation*, pp. 527-538, Dec. 1987.
- [13] T. J. Fan, "Describing and recognizing 3-D objects using surface properties," Ph.D. dissertation, Dep. Comput. Sci., Univ. Southern California, Los Angeles, Aug. 1988.
- [14] O. D. Faugeras and M. Hebert, "The representation recognition and locating of 3-D objects," *Int. J. Robotics Res.*, vol. 5, no. 3, pp. 27-52, Fall 1986.
- [15] W. E. L. Grimson and T. Lozano-Pérez, "Model-based recognition and localization from sparse range or tactile data," *Int. J. Robotics Res.*, vol. 3, no. 3, pp. 3-35, Fall 1984.
- [16] —, "Localizing overlapping parts by searching the interpretation tree," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 4, July 1987.
- [17] K. T. Gunnarsson, "Optimal part localization by data base matching with sparse data and dense data," Ph.D. dissertation, Dep. Mech. Eng., Carnegie-Mellon Univ., Pittsburgh, PA, Apr. 27, 1987.
- [18] P. Horaud and R. C. Bolles, "3DPO's strategy for matching three-dimensional objects in range data," in *Proc. Int. Conf. Robotics*, Atlanta, GA, Mar. 13-15, 1984, pp. 78-85.
- [19] B. K. P. Horn, "Extended Gaussian images," *Proc. IEEE*, vol. 72, pp. 1656-1678, Dec. 1984.
- [20] B. K. P. Horn and K. Ikeuchi, "The mechanical manipulation of randomly oriented parts," *Science America*, vol. 251, no. 2, pp. 100-111, Aug. 1984.
- [21] K. Ikeuchi, "Recognition of 3-D objects using the extended Gaussian image," in *Proc. 7th Int. Joint Conf. Artificial Intelligence*, Vancouver, B.C., Canada, Aug. 24-28, 1981, pp. 595-600.
- [22] —, "Precompiling a geometrical model into an interpretation tree for object recognition in bin-picking tasks," in *Proc. DARPA Image Understanding Workshop*, Feb. 1987, pp. 321-339.
- [23] J. L. Jezouin, P. Saint-Marc, and G. Medioni, "Building an accurate range finder with off the shelf components," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI, June 5-9, 1988.
- [24] R. Nevatia and T. O. Binford, "Description and recognition of complex-curved objects," *Artificial Intell.*, vol. 8, pp. 77-98, 1977.
- [25] M. Oshima and Y. Shirai, "A scene description method using three-dimensional information," *Pattern Recognition*, vol. 11, pp. 9-17, 1979.
- [26] —, "Object recognition using three-dimensional information," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 3, no. 4, pp. 353-361, July 1983.
- [27] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge, MA: M.I.T. Press, 1984.

- [28] K. Rao and R. Nevatia, "Generalized cone descriptions from sparse 3-D data," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami Beach, FL, June 22-26, 1986, pp. 256-263.



Ting-Jun Fan (M'88) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, in 1980, the M.S. degree in computer engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1982, and the M.S. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, in 1986 and 1988, respectively.

He is currently a Research Staff Member at IBM Thomas J. Watson Research Center, Yorktown Heights, NY. His research interests include computer graphics, user interface design, computer animation, and computer vision.

Dr. Fan is a member of the Association for Computing Machinery.



Gerard Medioni (S'82-M'83) received the Diplome d'Ingenieur Civil from the Ecole Nationale Supérieure des Télécommunications, Paris, France, in 1977, and the M.S. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, in 1980 and 1983, respectively.

He is currently an Assistant Professor in the Departments of Electrical Engineering and Computer Science, University of Southern California. His research interests include computer vision, artificial intelligence, and robotics.

Dr. Medioni is a member of the Association for Computing Machinery and the American Association for the Advancement of Science.

Ramakant Nevatia (S'71-M'74-SM'86), for a photograph and biography, see this issue, p. 1139.

REVIEW OF COMPUTER VISION AND PATTERN RECOGNITION IN 1986
Ting-Jun Fan, *IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA*
(Received 27 January 1987; accepted 10 March 1987)
Computer vision and pattern recognition have been two major fields of research in computer science during the past two decades. In the last few years, there has been a significant increase in the number of publications in these areas. This review article attempts to provide an overview of the major developments in these fields during 1986. It is organized into four main sections: computer vision, pattern recognition, computer graphics, and computer vision applications. The first section discusses the basic principles and methods of computer vision, including image acquisition, feature extraction, and scene interpretation. The second section covers the various techniques used in pattern recognition, such as statistical methods, neural networks, and rule-based systems. The third section deals with the use of computers in generating and manipulating graphical images. The fourth section focuses on the application of computer vision and pattern recognition in various domains, such as robotics, medical imaging, and remote sensing. The article concludes with a summary of the current trends and future directions in these fields.

Computer vision and pattern recognition have been two major fields of research in computer science during the past two decades. In the last few years, there has been a significant increase in the number of publications in these areas. This review article attempts to provide an overview of the major developments in these fields during 1986. It is organized into four main sections: computer vision, pattern recognition, computer graphics, and computer vision applications. The first section discusses the basic principles and methods of computer vision, including image acquisition, feature extraction, and scene interpretation. The second section covers the various techniques used in pattern recognition, such as statistical methods, neural networks, and rule-based systems. The third section deals with the use of computers in generating and manipulating graphical images. The fourth section focuses on the application of computer vision and pattern recognition in various domains, such as robotics, medical imaging, and remote sensing. The article concludes with a summary of the current trends and future directions in these fields.

"Model-Based Recognition in Robot Vision" by R.T. Chin and C.R. Dyer from *Computing Surveys*, Volume 18, Number 1, March 1986, pages 67-108. Copyright 1986, Association for Computing Machinery, Inc., reprinted with permission.

Model-Based Recognition in Robot Vision

ROLAND T. CHIN

Electrical and Computer Engineering Department, University of Wisconsin, Madison, Wisconsin 53706

CHARLES R. DYER

Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706

This paper presents a comparative study and survey of model-based object-recognition algorithms for robot vision. The goal of these algorithms is to recognize the identity, position, and orientation of randomly oriented industrial parts. In one form this is commonly referred to as the "bin-picking" problem, in which the parts to be recognized are presented in a jumbled bin. The paper is organized according to 2-D, $2\frac{1}{2}$ -D, and 3-D object representations, which are used as the basis for the recognition algorithms. Three central issues common to each category, namely, feature extraction, modeling, and matching, are examined in detail. An evaluation and comparison of existing industrial part-recognition systems and algorithms is given, providing insights for progress toward future robot vision systems.

Categories and Subject Descriptors: I.2.9 [Artificial Intelligence]: Robotics—*sensors*; I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—*modeling and recovery of physical attributes*; I.4.6 [Image Processing]: Segmentation; I.4.7 [Image Processing]: Feature Measurement—*invariants; size and shape; texture*; I.4.8 [Image Processing]: Scene Analysis; I.5.4 [Pattern Recognition]: Applications—*computer vision*.

General Terms: Algorithms

Additional Key Words and Phrases: Bin picking, computer vision, 2-D, $2\frac{1}{2}$ -D, and 3-D representations, feature extraction, industrial part recognition, matching, model-based image understanding, modeling, robot vision

INTRODUCTION

Research and development in computer vision has increased dramatically over the last thirty years. Application areas that have been extensively studied include character recognition, medical diagnosis, target detection, and remote sensing. Recently, machine vision for automating the manufacturing process has received considerable attention with the growing interest in robotics. Although some commercial vision systems for robotics and industrial auto-

mation do exist, their capabilities are still very primitive. One reason for this slow progress is that many manufacturing tasks require sophisticated visual interpretation, yet demand low cost and high speed, accuracy, and flexibility. The following delineates some of these requirements:

- **Speed.** The processing speed of acquiring and analyzing an image must be comparable to the speed of execution of the specific task. Often, this "real-time" rate is less than fractions of a second per part.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1986 ACM 0360-0300/86/0300-0067 \$00.75

CONTENTS

INTRODUCTION

1. MODEL-BASED OBJECT RECOGNITION
2. MODELS, FEATURES, AND MATCHING
3. 2-D IMAGE REPRESENTATIONS
 - 3.1 Examples of Global Feature Methods
 - 3.2 Examples of Structural Feature Methods
 - 3.3 Examples of Relational Graph Methods
 - 3.4 Comparison of the Three Methods for 2-D Object Representation
4. 2½-D SURFACE REPRESENTATIONS
 - 4.1 Example 1: A Relational Surface Patch Model
 - 4.2 Example 2: A Relational Surface Boundary Model
 - 4.3 Other Studies
5. 3-D OBJECT REPRESENTATIONS
 - 5.1 Example 1: A Surface Patch Graph Model
 - 5.2 Example 2: Hierarchical Generalized Cylinders
 - 5.3 Example 3: Multiview Feature Vectors
 - 5.4 Example 4: Multiview Surface Orientation Features
 - 5.5 Other Studies
6. RELATED SURVEYS
7. SUMMARY

ACKNOWLEDGMENTS

REFERENCES

• **Accuracy.** The recognition rate of objects in the scene and the accuracy in determining parts' locations and orientations must be high. Although there are instances where engineering solutions can be applied to improve accuracy (e.g., by controlling lighting and positional uncertainty), these solutions may not be realistic in terms of the actual environment in which these tasks must be performed.

• **Flexibility.** The vision system must be flexible enough to accommodate variations in the physical dimensions of multiple copies of a given part, as well as uncertainties in part placement due to individual workstation configurations. Furthermore, many robot vision tasks are distinguished by their performance in dirty and uncontrolled environments.

To be fully effective, future robot vision systems must be able to handle complex industrial parts. This includes recognizing various types of parts and determining their position and orientation in industrial

environments. In addition, vision systems must be able to extract and locate salient features of parts in order to establish spatial references for assembly and handling operations and be able to verify the success of these operations.

The performance requirements indicated above are not the only factors distinguishing robot vision from other application areas and general computer vision research. The nature of the domain of objects must also be recognized. Most industrial parts-recognition systems are *model-based systems* in which recognition involves matching the input image with a set of predefined models of parts. The goal of such systems is to precompile a description of each of a known set of industrial parts, then to use these object models to recognize in an image each instance of an object and to specify its position and orientation relative to the viewer. In an industrial environment the following types of constraints and properties that distinguish this problem domain are usually found:

- The number of parts in a given domain is usually small (1–50).
- Parts may be exactly specified, with known tolerances on particular dimensions and features.
- Parts often have distinctive features (e.g., holes and corners), which are commonly found on many different types of parts.
- In scenes containing multiple parts, there are a number of possible configurations (e.g., touching parts, overlapping parts, and parts at arbitrary orientations with respect to one another and the camera).

A growing number of studies have been conducted investigating various approaches to machine recognition of industrial parts. The body of literature generated from this developing field is both vast and scattered. Numerous journal publications have discussed issues involved in industrial vision system design and requirements. A significant number of research activities have been reported on the development of prototype systems for certain specific industrial applications. These studies are concerned with providing pragmatic solutions to current problems in industrial

vision. Some of them show the adequacy of image-processing techniques and the availability of technology needs for practical automation systems. Others are concerned with the development of more general parts-recognition algorithms that work in less controlled environments.

While several related survey papers have been published on the topic of robot vision and industrial-parts recognition (see Section 6), this paper presents a broader and more comprehensive approach to this subject. We concentrate on a comparative survey of techniques for model-based recognition of industrial parts. Related topics that are largely or entirely omitted from this paper are (a) industrial visual inspection applications, methodologies, and systems; (b) machine vision applications and research activities in private industry that have not been published; (c) the role of software and hardware implementation and the use of special-purpose optical and digital imaging devices; (d) the use of other sensory data (e.g., tactile data) as additional aids for recognition; and (e) the examination of the economic, social, and strategic issues that justify the use of robot vision.

1. MODEL-BASED OBJECT RECOGNITION

A number of factors limit the competence of current recognition systems for complex industrial parts. One of the major limitations is the low dimensionality in spatial representation and description of parts. Simple objects presented against a high-contrast background with no occlusion are recognized by extracting simple 2-D features, which are matched against 2-D object models. The lack of higher dimensional spatial descriptions (e.g., 3-D volumetric representations) and their associated matching and feature extraction algorithms restrict the system's capabilities to a limited class of objects observed from a few fixed viewpoints. The ability to recognize a wide variety of rigid parts independent of viewpoint demands the ability to extract view-invariant 3-D features and match them with features of 3-D object models. Another problem is the lack of descriptions of surface characteristics of industrial

parts. Without using properties of the surface, many recognition tasks cannot be accomplished by machine vision. It can be concluded that the dimensionality of spatial description and representation is highly dependent on both the particular application and its intended level of accomplishment. Many levels of spatial description (2-D, 3-D, and intermediate levels that fill the gap that exists between images and physical objects) are needed to fulfill various tasks. See, for example, Binford [1982], Brady [1982b], and Tenenbaum et al. [1979] for more discussion of the limitations of current robot vision systems.

Three central issues arise as a consequence of the problems mentioned above: (1) What *features* should be extracted from an image in order to describe physical properties and their spatial relations in a scene adequately? (2) What constitutes an adequate representation of these features and their relationships for characterizing a semantically meaningful class of objects; that is, in what form should features be combined into object *models* such that this description is appropriate for recognizing all objects in the given class? (3) How should the correspondence or *matching* be done between image features and object models in order to recognize the parts in a complex scene?

In this paper we discuss a variety of solutions to these issues. It is convenient to categorize all industrial parts-recognition systems into several classes before focusing on their problems, requirements, limitations, and achievements. The selected cases fall into three categories on the basis of their dimensionality of spatial description. To be more specific, we have grouped the reported studies into three classes: 2-D, 2½-D, and 3-D representations, presented in Sections 3, 4, and 5, respectively. It is natural to organize the studies in this fashion since systems within each class usually make similar assumptions. The grouping is also intended to provide the readers with an easy understanding of the state-of-the-art technology related to industrial parts recognition. Associated with each category are issues related to feature extraction, modeling, and matching, and these are discussed in detail.

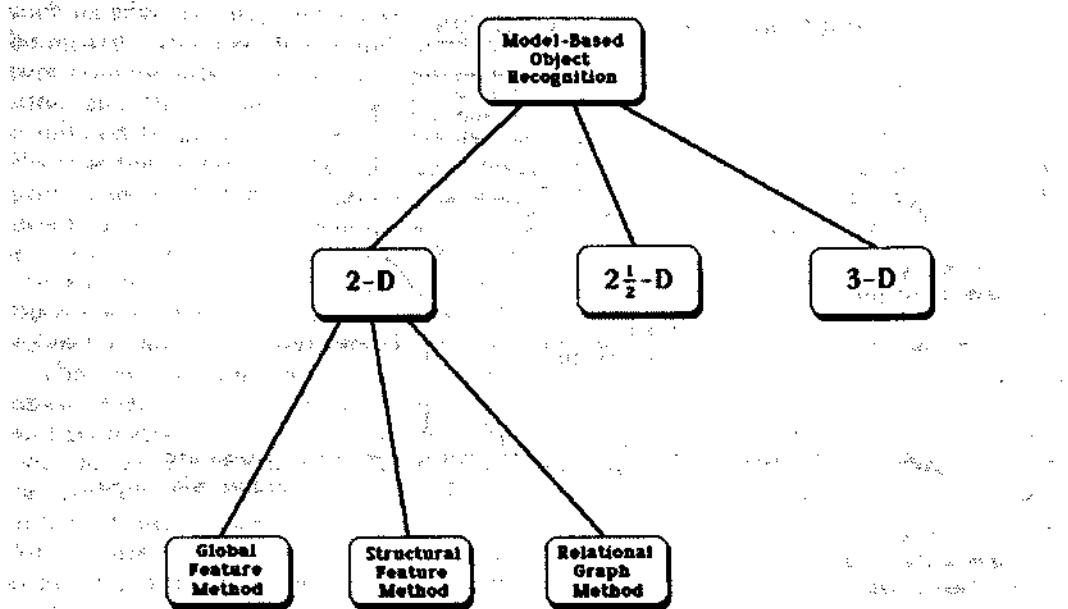


Figure 1. Organization of the survey.

Section 2 discusses the goals of each of these three components. Figure 1 provides a graphical summary of our organization.

3-D spatial descriptions define exact representations in "object space" using an object-centered coordinate system. 3-D representations are viewpoint-independent, volumetric representations that permit computations at an arbitrary viewpoint and to an arbitrary precision of detail.

2-D spatial descriptions are viewer-centered representations in "image space." Each distinct view is represented using, for the most part, shape features derived from a gray-scale or binary image of a prototype object. This class of representation is appropriate when the viewpoint is fixed and only a small number of stable object positions are possible. The 2-D representations are further subdivided into three classes according to their method of object modeling. They are (a) the global feature method, (b) the structural feature method, and (c) the relational graph method. This categorization is discussed in more detail in Section 3.

2½-D representations have attributes of both 2-D and 3-D representations, using features defined in "surface space." These spatial descriptions are viewer-centered

representations, but depend on local surface properties of the object in each view, for example, range (i.e., depth) and surface orientation.

Many reported studies using the above image representations are worth mentioning, but it is impossible to discuss all of them in detail. These studies are included in the sections under "Other Studies." They are included to provide a more complete annotated bibliography on industrial parts-recognition algorithms.

2. MODELS, FEATURES, AND MATCHING

A parts-recognition system can be broken down into a training phase and a classification phase, as illustrated in Figure 2. The three major components of the system are *feature extraction*, *object modeling*, and *matching*. The sensor and feature extraction components in training are not necessarily the same as those in classification. In this section we specify the general goals of each of these three segments.

Models: The use of models for image understanding has been studied extensively (e.g., see Binford [1982] and Rosenfeld and Davis [1979]). However, most of the models

INDUSTRIAL PART RECOGNITION

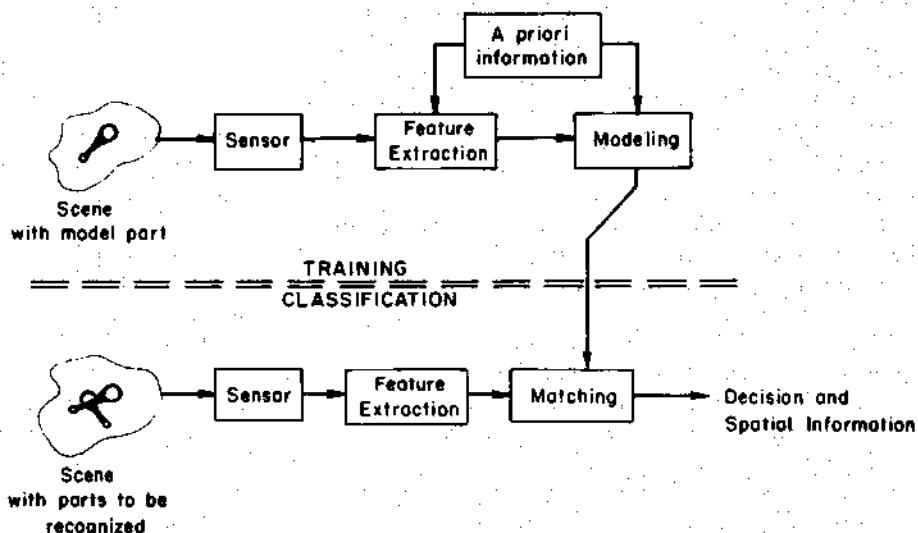


Figure 2. Components of a model-based recognition system.

that have been investigated are relatively simple and do not provide adequate descriptions for recognizing industrial parts in complex scenes. Although many models of regions and images have been developed on the basis of the homogeneity of gray-level properties (e.g., texture and color), they have not been widely used for industrial applications. For this reason, this type of model is not discussed further here.

Models based on geometric properties of an object's visible surfaces or silhouette are commonly used because they describe objects in terms of their constituent shape features. Throughout this paper we focus on alternative methods for representing object models using 2-D, $2\frac{1}{2}$ -D, and 3-D shape features.

2-D models have the advantage that they can be automatically constructed from a set of prototype objects, one from each possible viewpoint. (In general, it is nontrivial to automatically construct 3-D representations from a set of 2-D views.) They have the disadvantage that they do not make the full 3-D description of an object explicit—their completeness depends on the complexity of the object and the number and positions of the viewpoints used. In industrial parts-recognition applications, however, it is frequently the case that limited

allowable viewpoints, limited possible stable configurations, and object symmetries substantially reduce the number of distinct views that must be considered.

$2\frac{1}{2}$ -D models use viewer-centered descriptions of surfaces instead of boundaries and therefore have the advantage of more accurately representing the complete object and hence improving chances for reliable recognition. Their disadvantages include the additional step of accurately deriving the surface description and the need, as with 2-D methods, for separate viewpoint-specific representations.

3-D models allow the most general and complete descriptions of objects from an unconstrained viewpoint. They can be derived directly from a CAD-like representation and describe the physical volume filled by objects. Of course, this compact description is not directly comparable with 2-D or $2\frac{1}{2}$ -D features, which are extracted from an image. Therefore the principle disadvantage is that a more sophisticated 2-D to 3-D correspondence procedure must be defined.

Features. The problem of selecting the geometric features that are the components of the model is integrally related to the problem of model definition. Image features

such as edge, corner, line, curve, hole, and boundary curvature define individual feature components of an image. These features and their spatial relations are then combined to generate object descriptions. Because they represent specific higher level primitives that correspond to physically meaningful properties of the scene, features are less sensitive to variations than the original noisy gray-level values. Usually, the decision of what features to use is rather subjective and application specific.

The features important for industrial-image analysis are most often boundaries and geometric measurements derived from boundaries. These features can be roughly categorized into three types: global, local, and relational features. Examples of *global features* are perimeter, centroid, distance of contour points from the centroid, curvature, area, and moments of inertia. Examples of *local features* include line segment, arc segment with constant curvature, and corner, defining pieces of an object's boundary. Examples of *relational features* include a variety of distance and relative orientation measurements interrelating substructures and regions of an object.

Most existing industrial-vision systems and algorithms extract features from industrial objects against a high-contrast background with controlled lighting to eliminate shadows, highlights, and noisy backgrounds. The process of feature extraction usually begins by generating a binary image from the original gray-scale image by choosing an appropriate threshold, or simply by using a sensor that produces binary images. The use of a binary representation reduces the complexity of data that must be handled, but it places a serious limitation on the flexibility and capabilities of the system. After thresholding, 2-D features are extracted from the binary image. Thus, in these systems features are simple functions of a part's silhouette. A tutorial on binary image processing for robot-vision applications is given in Kitchin and Pugh [1983].

Most feature-extraction algorithms used in these binary imaging systems are simple outline-tracing algorithms. They detect boundaries of simple planar objects but

usually fail to detect low-contrast surface boundaries. Another limitation is that they attempt to deal with 3-D physical objects in terms of 2-D features. This simplification might meet the cost requirement of many industrial applications, but it lacks the capability and flexibility required by many other industrial-vision tasks. Finally, current systems seldom have representations of physical surface properties such as surface reflectance and surface orientation (i.e., $2\frac{1}{2}$ -D representations). Such information is lost in reducing the gray-scale image to a binary image or to a piecewise constant image. Without using these properties of the object's surface, many important industrial-vision tasks that are easy for people to perform will remain beyond the competence of computer-vision systems.

A few current vision systems are capable of extracting useful information from images of complex industrial parts with considerable noise caused by dirt and unfavorable lighting conditions. These systems process gray-scale images with reasonable dynamic range. The most important drawback of gray-scale image processing is the slow processing rate in extracting features. Most of these systems employ sophisticated feature-extraction methods, but their matching procedures are still based on 2-D models.

Matching. Given a set of models that describes all aspects of all parts to be recognized, the process of model-based recognition consists of matching features extracted from a given input image with those of the models. The general problem of matching may be regarded as finding a set of features in the given image that approximately matches one model's features. Some methods rely on total image matching using cross-correlation types of measures applied to image intensities or coefficients of some mathematical expansion (e.g., orthogonal expansion). They can be formulated as global optimization problems to achieve great reliability, but are computationally too expensive. Moreover, the image is generally noisy, and parts within the image will be occluded and located at random positions and orientations.

Consequently, matching algorithms of this type has little value in industrial parts-recognition systems.

Matching techniques using 2-D global, local, or relational features, or a combination of these features, provide a way to recognize and locate a part on the basis of a few key features. Matching using features becomes a model-driven process in which model features control the matching process. Several model-driven matching techniques have been developed. Most are invariant to translation and rotation, and are not too sensitive to noise and image distortion. The choice of matching process is highly dependent on the type of model used for object representation. Models using global features are usually associated with statistical pattern-recognition schemes. Models based on local features are usually associated with syntactic matching methods, and models using a combination of local and relational features are usually associated with graph-matching techniques.

Matching using $2\frac{1}{2}$ -D models requires procedures that must compare sets of planar or curved surface patches. This can be done either directly by finding best-fitting regions between the image and models, or indirectly by comparing features derived from these surfaces. Matching with 3-D models requires the most extensive processing in order to make explicit the 2-D projection of the model that best matches the image features.

3. 2-D IMAGE REPRESENTATIONS

In this section we review recognition algorithms that are based on 2-D image representations. Each 3-D part is modeled by a set of one or more distinct views. This set of 2-D views can be determined either by training the system with the part in each of its possible stable positions or by computing these positions directly from a CAD description. Figure 3 shows a set of stable orientations of a part and their corresponding models [Lieberman 1979]. These viewer-centered representations treat each view independently, reducing the problem to 2-D by using 2-D image features and

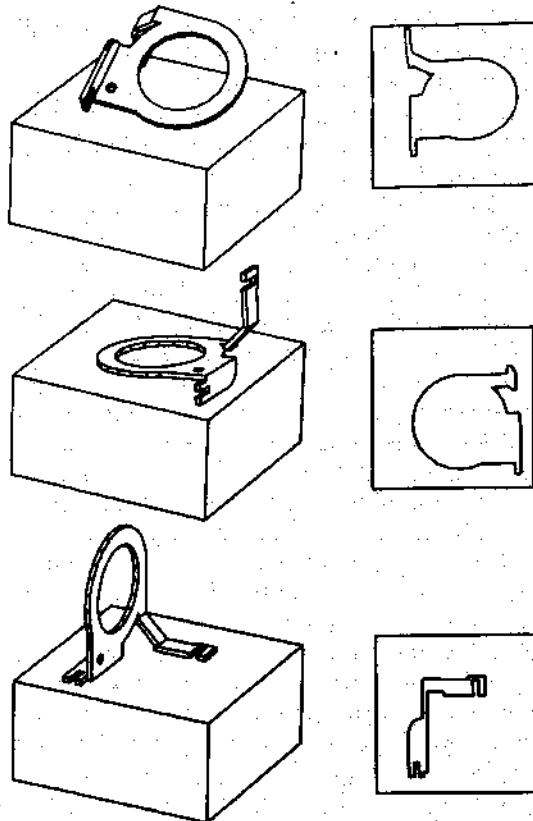


Figure 3. A set of stable orientations for a part and the corresponding silhouettes calculated for each of the orientations when viewed from directly overhead. (From Lieberman [1979].)

their relations as primitives. For each viewpoint, a sufficient set of image-space-derived features and relations are combined for modeling the object.

We classify 2-D object-recognition methods into three classes based on the kinds of models and matching algorithms they employ. The first type of method uses global features of an object's size and shape (e.g., perimeter and area) organized in geometric property lists. This class of method is referred to as the *global feature method*.

The second type of method uses local features that describe more complex properties of the object, usually in terms of line and curve segments defining the object's boundary. Typically, the features are organized as a highly structured and abstracted representation. This class of

Table 1. The Three Methods Based on 2-D Image Representations

Method	Feature	Model	Matching
Global feature	Global scalar	Feature vector (unordered)	Statistical pattern recognition
Structural feature	Local	Ordered string of features or abstract description of feature strings	Syntactical or verification of string descriptions
Relational graph	Local and relational	Relational graph	Graph searching

method is referred to as the *structural feature method*.

The third type uses local and relational features which are organized in a graph. Nodes describe local features and arcs have associated properties that describe the relationship between the pairs of features that they connect. This type is referred to as the *relational graph method*.

The three types of object-recognition methods are summarized in Table 1. All three components (feature, models, and matching) of each type have distinctly different characteristics. Their strengths and weaknesses are discussed in detail in the following.

Global Feature Method. Global features such as area and perimeter are relatively easy to extract from a single image. In some systems the feature set also includes position and orientation descriptors, such as center of gravity, and moments of inertia, which provide useful information for part manipulation. Models using global features are feature lists, and the order of features in the feature list is unimportant. This type of method is usually associated with the classical feature-space classification scheme. The features of a part may be thought of as points in n -dimensional feature space, where n is the number of global feature measurements. The recognition of an unknown part with an n -dimensional feature vector involves statistical pattern-recognition methods where the feature vector is compared with each of the model feature vectors. Both parallel (e.g., the Bayes classifier) and hierarchical/sequential decision rules (e.g., the decision-tree classifier) can be used. The computational expense associated with the parallel classification increases steeply with dimension, but optimal results are achievable. There are numerous advantages to hierarchical

classification. Most important, the decision procedure can be designed to be both inexpensive and effective; but the overall accuracy is not so great as with the parallel decision rules.

Structural Feature Method. Models can be constructed using abstracted and precise geometric representations such as arcs, lines, and corners. These features are local in nature, each describing a portion of the object. They are organized in a highly structured manner, such as an ordered list or a sequence of equations. The ordering of features in this type of method is usually related to the object's boundary in such a way that following the entire feature list sequentially is equivalent to tracing the boundary of the object. Recognition (matching) uses a hypothesis-verification procedure. The structured local features of the model are used to predict where objects are located in the scene. Then, features of the hypothesized object are measured, on the basis of the prediction hypothesized by the model, in order to verify and fine-tune the match. In addition, this type of method allows the use of syntactic pattern-recognition approaches, in which local features are transformed into primitives which are organized into strings (sentences) by some highly structured grammatical rules. Matching is performed by parsing.

Relational Graph Method. Objects can be represented structurally by graphs. In this method, geometrical relations between local features (e.g., corner and line) are of particular interest. The relational structure can be represented by a graph in which each node represents a local feature and is labeled with a list of properties (e.g., size) for that feature. Arcs represent relational features linking pairs of nodes and are

labeled with lists of relation values (e.g., distance and adjacency). Recognition of the object becomes a graph-matching process. This type of method can be used to handle overlapping parts where a partially visible part corresponds to a subgraph. The matching reduces to one of finding the subgraph.

The remainder of this section covers examples of each of these three methods in detail.

3.1 Examples of Global Feature Methods

The predominant method to date, especially in commercial systems, uses a set of 2-D, global shape features describing each possible stable object view. Recognition is achieved by directly comparing features of an object with those of the model. This type of model is compact and facilitates fast matching operations because of the limited number and size of the feature vectors extracted from a given image.

The major limitations of this type of model are (1) each possible 2-D view of an object must be described by a separate model; (2) all objects in an image must be extracted by a single predefined threshold (hence lighting, shadows, and highlights must be controlled); and (3) objects may not touch or overlap one another, nor may objects have significant defects. (A defective object that is not sufficiently similar to any model can be recognized as a reject, but this may not be adequate in many applications.)

3.1.1 Example 1: A System Based on Connected Components

Model. The SRI Vision Module [Gleason and Agin 1979] is the prototypical system of the global feature method. The user interactively selects a set of global features which are used to construct an object model as a feature vector. This process is an example of the "training by showing" method of modeling. For each distinct viewpoint of each object modeled, a sample prototype is used to compute the values of each feature selected. The selection of which features are sufficient to discriminate adequately among objects is determined by trial and error. Thus, if a new object is introduced

later into the system, the complete process of feature selection must be repeated in order to discriminate properly among all of the possible objects in a scene.

Features. Each connected component in the input binary image is extracted so that each of these regions can be analyzed independently. For each connected component a number of gross scalar shape descriptors are computed (such as number of holes, area, perimeter, boundary chain code, compactness, number of corners, and moments of inertia). All of these features can be computed in a single pass through the image, either from the binary image representation or from the image's run-length encoded representation (which is more compact).

Matching. Matching uses a decision-tree method based on the list of global features associated with each model [Agin and Duda 1975]. The tree is automatically constructed from the models as follows. (1) The feature values with the largest separation for a given feature and pair of object models are found, and this feature is used to define the root node of the tree. That is, a threshold is selected for this feature that distinguishes between these two models. (2) Two children of the root node are constructed such that all models that have a feature value less than or equal to the threshold are associated with the left child; the right child is assigned all models with a feature value greater than the threshold. (3) This procedure is repeated recursively, dividing a set of model candidates associated with a node into two disjoint subsets associated with its two children. A terminal node in the tree is one that contains a single model. Figure 4 illustrates such a decision tree.

The decision-tree method has the primary advantage of speed, but it also has the disadvantage of not allowing similar models to be explicitly compared with a given list of image features.

Alternatively, the best matching model to a given list of global features extracted from an object in an image is computed using statistical pattern-recognition schemes (e.g., a nearest-neighbor classifier)

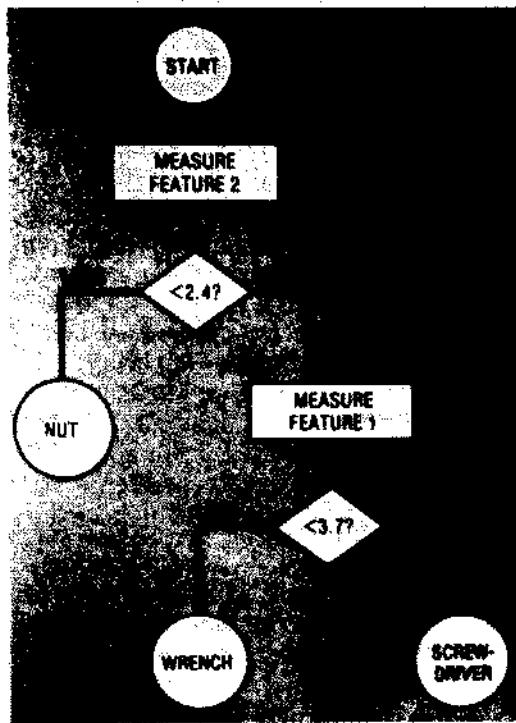


Figure 4. A decision tree classifier. (From Agin [1980]; © IEEE 1980.)

in feature space, as illustrated in Figure 5. That is, if n features are used to describe all models, then each model is represented by a point in n -dimensional feature space. Given a new feature list extracted from an image, the component is recognized as being an instance of the model that is closest in feature space.

3.1.2 Example 2: Using Global Features to Identify Grasp Points

Kelley et al. [1982] have developed a simple system for rapidly determining grasp points for a robot arm which must remove randomly oriented cylindrical workpieces piled in a bin. Thus all parts are known to be of the same type, and only their positions and orientations are unknown. In this case several simplifications of the SRI Vision Module approach are possible. First, a shrinking operator is applied to reduce the regions in the original binary image into small, connected components of pixels. These components are then sorted in order by size

(i.e., number of pixels), and the largest component is selected as the grasp-point location for the gripper. Position and orientation features of the selected region (e.g., the centroid and axes of minimum and maximum moments of inertia through the centroid) are computed to determine the location and orientation of the gripper relative to the image. An additional feature, the ratio of eigenvalues of axes of minimum and maximum moments of inertia, is also computed to determine the inclination of the cylinder with respect to the image plane, so as to determine the appropriate opening of the gripper's fingers. The "i-bot" system is based on this technique [Zuech and Ray 1983]. This system computes the locations and orientations of a maximum of three workpieces in a bin in 2 seconds.

3.1.3 Other Studies

Several systems based on the SRI Vision Module have been developed commercially, including Machine Intelligence Corporation's VS-100 system, Automatix's Autovision system, Unimation's Univision I, Control Automation's V-1000, Intelleddex V-100 Robot Vision System, and Octek's Robot Vision Module. The VS-100 system (and the related system for Puma robots, Univision I) accepts images up to 256×256 and thresholds them at a user-specified gray level. Up to 12 objects can be in an image and up to 13 features can be used to model each part. Recognition times of from 250 milliseconds (ms) (1 feature) to 850 ms (11 features) per object are typical [Rosen and Gleason 1981]. The Autovision 4 system processes images up to 512×256 , and recognition performance is listed at over 10 parts per second for simple parts [Villers 1983].

Birk et al. [1981] model objects by a set of coarse shape features for each possible viewpoint. For a given viewing position, the thresholded object is overlaid with a 3×3 grid (each grid square's size is selected by the user), centered at the object's centroid and oriented with respect to the minimum moment of inertia. A count of the number of above threshold pixels in each grid square is used to describe the object.

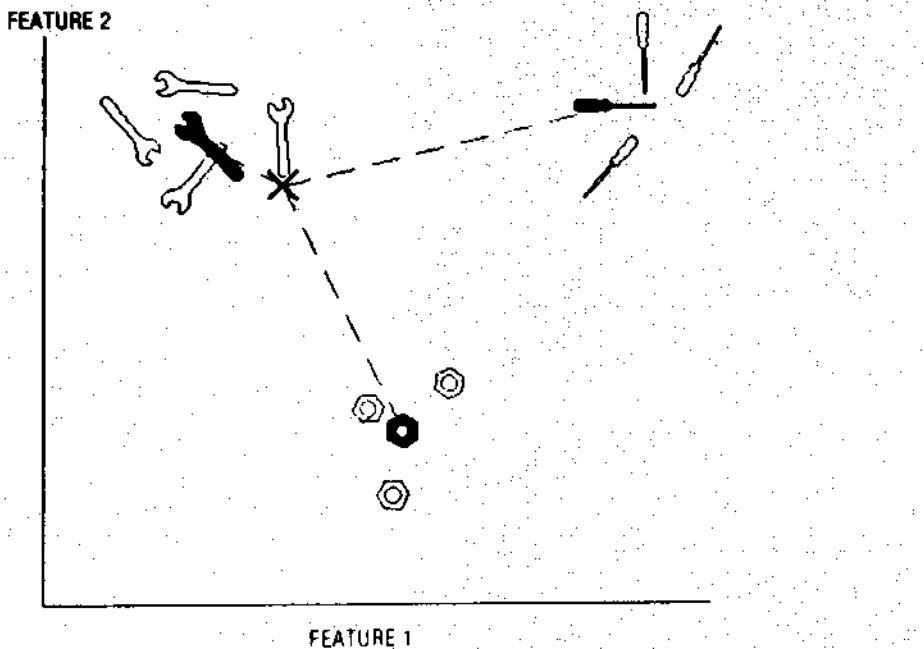


Figure 5. A nearest neighbor classifier. (From Agin [1980]; © IEEE 1980.)

CONSIGHT-I [Holland et al. 1979] avoids the problem of threshold selection for object detection by employing a pair of line light sources and a line camera focused at the same place across a moving conveyor belt of parts. Part boundaries are detected as discontinuities in the light line, as shown in Figure 6. Features, such as centroid and area, are computed for each object as it passes through the line of light.

Fourier descriptors [Zahn and Roskiew 1972] have been suggested as shape descriptors for industrial-parts recognition by Persoon and Fu [1977]. A finite number of harmonics of the Fourier descriptors are computed from the part boundary and compared with a set of reference Fourier descriptors. A minimum-distance classification rule is used for the recognition of various classes of parts.

In gray-scale image processing the first step is usually to segment the image to find regions of fairly uniform intensity. This greatly increases the degree of organization for generating higher level descriptions such as shape and size. Perkins [1980] has developed a region-segmentation method for industrial parts using edges. This

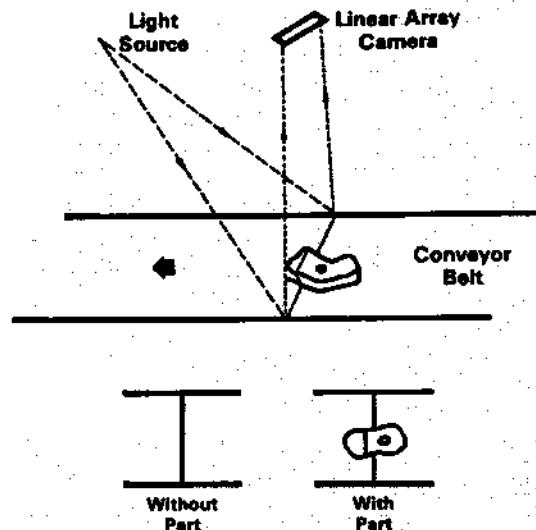


Figure 6. Basic lighting principle of the CONSIGHT-1 system and the computer's view of a part. (From Holland et al. [1979].)

method uses an expansion-contraction technique in which the edge regions are first expanded (to close gaps) and then contracted after the separate uniform regions have been identified. The process

is performed iteratively to preserve small segments.

An industrial-vision system, S.A.M., has been developed by Tropf et al. [1982], using binary image processing to extract global scalar features for inspection and parts recognition. The system is now commercially available for flexible manufacturing assembly systems [Brune and Bitter 1983]. A development system for machine vision based on the Machine Intelligence Corp. VS-100 has been developed and marketed [Chen and Milgram 1982]. The performance of the above vision system has also been evaluated by Rosen and Gleason [1981].

An experimental system has been developed by Page and Pugh [1981] to manipulate engineering parts from random orientation. Simple global scalar features are used to identify gripper locations. Typical recognition times are in the range 0.5–3 seconds.

3.2 Examples of Structural Feature Methods

The system described in the previous section included global shape and size features, which consisted, for the most part, of simple integral or real-valued descriptors. In this section we describe methods that use more complex features, for the most part structural descriptions of object boundaries.

3.2.1 Example 1: Line and Arc Boundary Segment Descriptions

Model. Perkins [1978] constructs 2-D models from boundary segments, called concurves, constructed from line segments and arcs that are extracted from training images of each stable view of each part. The list of concurves comprises a structural approach to describing objects that is not as sensitive to noise as most global features.

The model uses an object-centered coordinate system in which the origin is defined by either (a) the center of area of the largest closed concurve or (b) the center of a small closed concurve if it is sufficiently close to the center of the largest closed concurve. The axes are defined in terms of the direc-

tion of the least moment of inertia of the largest concurve.

For each concurve in the model a property list is computed, including type (circle, arc, line segment, complex curve, etc.), total length or radius of arcs, magnitude of total angular change, number of straight lines, number of arcs, bending energy, and compactness. In addition, rotational symmetries of the concurve and the complete object are computed as additional descriptors. Rotational symmetry is computed using a correlation-like technique which determines whether a sufficient percentage of concurve "multisectors" intersects the rotated concurve. Multisectors are short line segments that are placed at equal intervals along the concurve and at orientations perpendicular to the tangent directions at these points.

Features. Concurve features are used in order to represent the 2-D shape of a part as a line drawing of its boundary. This representation is compact and, because the boundary is smoothed before the concurves are computed, relatively insensitive to noise in the imaging system and environment. First, a gray-scale image is transformed into an edge map using the Hueckel edge operator [Hueckel 1971]. Next, edge points are thinned and connected together into long chains by using knowledge of proximity, directional continuity, and gray-scale continuity.

Finally, the chains are transformed into a group of concurves. A concurve is defined as an ordered list of shape descriptions which are generated by fitting a segment of the chain data to straight lines, or circular arcs, or a combination of both. This curve-fitting step is quite similar to the one used by Shirai [1975] in his feature-extraction algorithm. The fitting procedure first examines the curvature of the chain data (i.e., connected edge points). Next, it looks for abrupt changes in curvature and picks out end points (critical points) to set the bounds of each grouping. The chain of edge points in each group is fitted with a circular arc or straight line using Newton's method. An additional step that verifies and corrects for a poor fit is included. Figure 7 shows

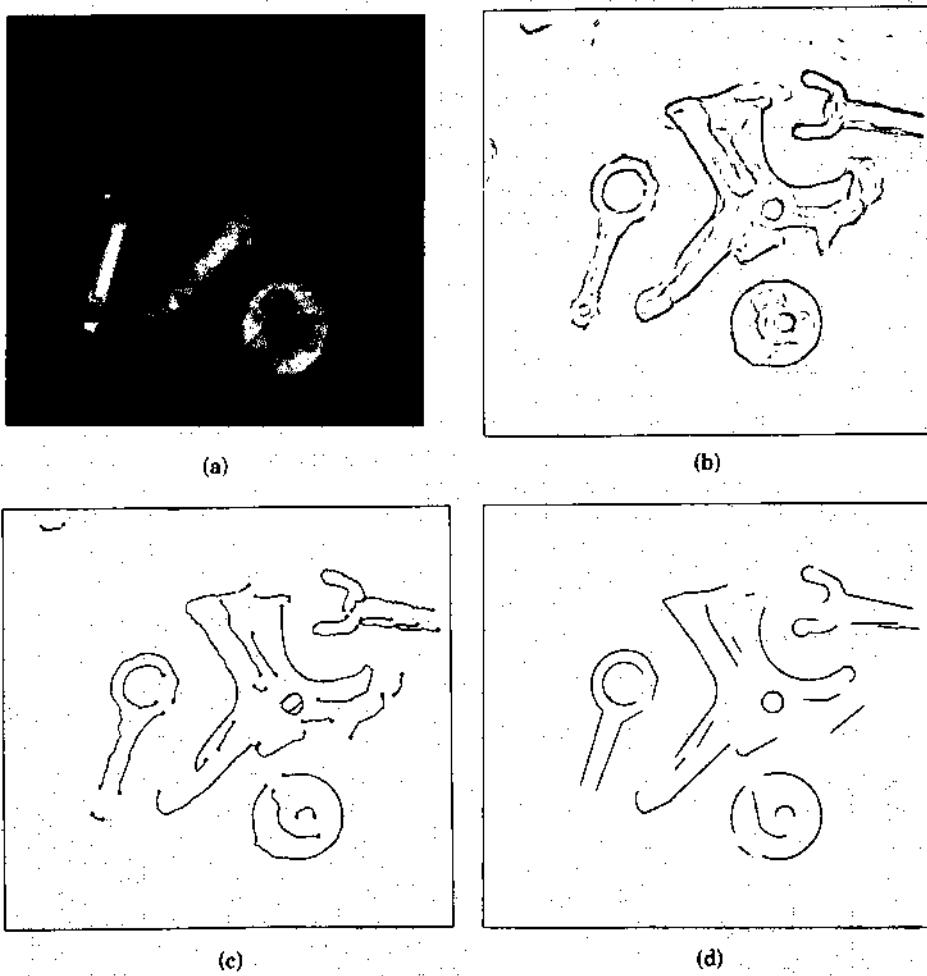


Figure 7. Concave representation. (a) Digitized image. (b) Edge points. (c) Chains with critical points at the ends of open chains. (d) Concavities. (From Perkins [1978]; © IEEE 1978.)

the various stages of extracting concavities from a sample image.

Matching. The matching process is performed in three steps. First, scalar measurements (length, area, etc.) extracted from the model and image concavities are compared. The comparison is an exhaustive matching procedure applied to all possible pairings between the model concavities and the image concavities, and the results, given in terms of likelihood measures, are arranged in an ordered list. Second, one model concavity is matched against one image concavity to determine a tentative transformation (x, y, θ) from model to im-

age coordinates. The pair with the highest likelihood is used first; successive pairs are compared until a tentative transformation is found. In cases in which the model concavity is symmetric, two matching pairs are required to determine the transformation. Third, a global check of the tentative transformation is performed by matching the complete model with the image. In this step a set of model multisectors is first transformed using the tentative transformation determined in the previous step. The transformed multisectors of the model are then superimposed on the image for a final comparison by intersecting each multisector with the image concavities. This matching

process is shown to be successful with closed concavities and has been tested with images containing partially overlapping parts.

3.2.2 Example 2: Hierarchical Boundary Segment Models

Model. In the system developed by Shirai [1978], object models are organized as a hierarchy of features consisting of main and secondary features. These features are edges represented by a description of their curvature in terms of an equation and endpoints. The main feature is the most obvious one found in an object, and it is used to imply the presence of the object during the initial stage of the search. Successful detection of the main feature generates clues for verifying the recognition. Secondary features are details of the object. These are chosen on the basis of the ease with which they may be found in a scene. For recognizing a cup, for example, the main feature can be a pair of vertical edges corresponding to the sides of a cup, and the secondary features can be other detail contours connected to the sides.

Features. The features used by Shirai are similar to those used by Perkins, consisting of long, connected edge segments that describe pieces of an object's boundary. The system first extracts edges using a conventional gradient operator. The extracted edges are classified into three types according to their intensity profiles. Next, an edge kernel is located by searching for a set of edge points of the same type which have similar gradient directions. A tracking algorithm is applied in both directions of the kernel to find a smoothly curved edge and its endpoints. Several passes are applied to locate all sets of smoothly connected edges in the scene. Finally, straight lines and elliptic curves are fit to each segment, and segments are merged together, if possible.

Matching. Recognition involves three steps. First, the main feature is located to get clues for the object. Next, a secondary feature is searched for to verify the main feature and to determine the region occu-

pied by the object. Finally, the other lines of the object are located to confirm the recognition.

3.2.3 Example 3: Accumulating Local Evidence by Clustering Pairs of Image and Model Boundary Segments

Model. Stockman et al. [1982] have proposed a method in which models of 2-D objects are defined by organizing (1) real vectors describing boundary segments and (2) abstract vectors linking primitive features (e.g., a vector connecting two holes) into a set. The set is in an object-centered coordinate system and is defined by modeling rules (e.g., size of the object, known a priori) to permit only certain combinations of features to be linked. The resulting model is a line-drawing version of the object, plus additional abstract vectors to allow increased precision and control over the matching process.

Features. Directed edge elements (vectors) are used as one type of primary feature containing directional, positional, and size information. First, point features (i.e., the tip and tail of a vector) are extracted, and then vectors are formed from suitable point pairs. Straight edge detectors, curved edge detectors, circle detectors, and intersection detectors are employed to define vectors between point pairs. Holes are detected by a set of circular masks, and curves and intersections are detected by linking edges together. Details of the feature-extraction procedure are presented in Stockman [1980].

Matching. Matching is done using a clustering procedure. The procedure matches all possible pairs of image and model features on the basis of local evidence. The matching in cluster space consists of points, each representing a match of an image feature to a model feature. A cluster of match points in this space is a good indication that many image features are matched to corresponding model features. In order to handle randomly placed objects, a rotation, scaling, and translation transformation is derived to extract parameters from all possible pairs of features.

Clustering is then performed in the space of all possible transformation parameter sets. This method is believed to be more robust because the clustering procedure integrates all local information before any recognition decision is made. A set of simulated carburetor covers and T-hinges are used to demonstrate the method. The reported results indicate that this method works well with isolated objects, but the success rate for recognizing overlapping parts is low.

3.2.4 Other Studies

Hattich [1982] uses contour elements, described in terms of straight-line segments, as the global structural features. Matching is done by iteratively constructing the model contour from image data.

Experiments on occluded part recognition have been performed by Turney et al. [1985] using edges as the features. Recognition is based on template matching between the model edge template and the edge image in the generalized Hough transform space [Ballard 1981a]. This algorithm is shown to be more efficient than direct template matching. Dessimoz [1978a, 1978b] recognizes overlapping parts by first mapping the objects' boundaries into a set of curves and then matching the curves with those in the model. Tropf [1980, 1981] has developed a recognition system for overlapping workpieces using corner and line primitives and semantic labeling. Structural knowledge of workpieces is used to construct models. Recognition uses heuristic search to find the best match based on a similarity measure. Ayache [1983] uses binary images and polygonal approximations to each connected component. Models are automatically constructed by analyzing a prototype part in its different stable positions. The matching is done first by generating a hypothesis of the object location and then by matching model segments to scene segments. The model location is sequentially adjusted by evaluating each match until the best match is found.

Bhanu has developed a hierarchical relaxation labeling technique for shape matching and has performed experiments

using 2-D occluded industrial parts [Bhanu 1983; Bhanu and Faugeras 1984]. Two-dimensional shapes are used as the global structural features, and they are represented by a polygonal approximation. The technique involves the maximization of an evaluation function which is based on the ambiguity and inconsistency of classification. Umetani and Taguchi [1979] use "general shapes," defined as artificial and nonartificial shapes, to study the properties and procedures for complex shape discrimination. Feature properties based on vertices, symmetry, complexity, compactness, and concavity have been investigated. These features are chosen on the basis of some psychological experiments, and a procedure to discriminate random shapes has been proposed [Umetani and Taguchi 1982].

Vamos [1977] has proposed the use of syntactic pattern recognition for modeling machine parts from picture primitives: namely, straight line, arc, node, and undefined. A set of syntax rules is used to characterize the structural relationships of these strings of primitives describing the part. The matching process is a syntax analysis or parsing procedure involving the use of similarity measures between two grammar strings or two graphs. Jakubowski has conducted a similar study using straight lines or curves as primitives to model machine part shapes and to generate part contours [Jakubowski 1982; Jakubowski and Kasprzak 1977].

Takeyasu et al. [1977] and Kashioka et al. [1977] have developed an assembly system for vacuum cleaners using integrated visual and tactile sensory feedback. First, global scalar features of various parts of the vacuum cleaner are used to locate the cleaner. Then, structural features, such as circles and arcs, are used in a template-matching step for the assembly operation.

Foith et al. [1981] describe an object boundary with respect to the centroid of the "dominant blob" defining the 2-D binary object. Circles of prespecified radii are centered on the centroid, their intersections with the object boundary are marked, and line segments are then drawn between these intersections and the centroid. The

sequence of angles between successive line segments is used as a rotation-invariant model of the object boundary.

3.3 Examples of Relational Graph Methods

This class of methods is based on a graph representation of a part. The graph is constructed in terms of locally detectable primitive features and the geometric relations between pairs of these features. This class of method is thus based on local rather than global features and has the following advantages: (a) local features may be cheaper to compute because they are simpler and can be selectively (sequentially) detected; (b) models are less sensitive to minor differences in instances of a given object type; (c) if a few local features are missing (owing to noise or occlusion), it may still be possible to recognize the object on the basis of the remaining features associated with the model; and (d) since a few types of local features are often sufficient to describe a large number of complex objects, it is possible to specify only a few types of local feature detectors which are applied to the image.

A disadvantage with this type of method is the fact that a large number of features must be detected and grouped together to recognize an object. Thus the matching algorithm used with these models must be more complex and may be somewhat slower than the matching algorithms used with the previous methods.

3.3.1 Example 1: A Two-Level Model of Coarse and Fine Features

Model. Yachida and Tsuji [1977] use a simple kind of feature graph representation plus a two-level model (for coarse-to-fine processing) to speed the search process. Each object is described by a set of models, one for each possible viewpoint. Each model contains a coarse representation of the object using global features, such as area and elongatedness, plus a description of the outer boundary (in polar coordinates). Each component extracted from an image is compared with each coarse model to determine whether it is sufficiently similar to warrant further comparison. Object

boundaries are compared by using cross-correlation as the measure of shape match.

The fine level of representation of each model is based on a higher resolution image and consists of a list of features such as outer boundary, holes, edges, and texture. Associated with each feature is an attribute list, location (relative to the object's centroid), and the expected likelihood that the feature can be extracted reliably. Features are ordered in the model by their reliability value.

Features. The feature-extraction process in this system is divided into several stages by using the idea of "planning"; that is, knowledge of the structure of an object guides the feature-extraction module in a top-down manner. Simple features are detected first in a coarse resolution image, and then more complex features are sought on the basis of the locations of the coarse features. Industrial parts used for demonstration are parts of a gasoline engine. In the preprocessing stage, a low-resolution version of the image is analyzed and outlines of objects are detected by thresholding. Each outline is then analyzed separately, using a high-resolution image of the region of interest to extract a finer outline of the object. By employing the method in Chow and Kaneko [1972], local histogramming and dynamic thresholding based on 11×11 windows are used in this step. Next, the object's gross properties, such as size, thinness ratio, and shape, are computed. This coarse description of the object is used to select candidate models for matching and to guide the extraction of finer features for final recognition. There are four features extracted in the fine-resolution processing stage, and they include circle, line, texture, and small hole. Each feature is extracted from a search region around the expected location in the gray-scale image. The circle detector uses thresholding as in the preprocessing step; the line finder, using dynamic programming, searches for the optimum sequence of edge points in the region that maximizes a measure of goodness; the texture detector measures edge strength per unit area and average edge direction; the small-hole

detector uses neighbor merging to locate circular objects.

Matching. The matching process examines the current information obtained from the scene and the model graphs of objects to propose the next matching step. The model relates features at a coarse resolution with more detailed features at a fine resolution, enabling the matching to be performed using simple features as cues. Given a tentative match between an image component and an object model based on the coarse model features, the fine model features are then successively compared. The object boundary matched at the coarse level determines a tentative match angle of rotation. For a given feature extracted from the image, a measure of the dissimilarity between it and each of the model features is computed. A cumulative dissimilarity measure is kept for each active model. When a model's dissimilarity exceeds a threshold, the model is rejected as a possible match. After the current feature has been compared with each of the remaining candidate models, a next-feature proposer analyzes the features described in these candidate models and proposes the most promising feature among them as the one to be examined next for recognizing the input object.

3.3.2 Example 2: Corner and Hole Relational Models

Model. Chen et al. [1980] estimate the position and orientation of workpieces using the 3-D locations of at least three noncollinear feature points. The location of features is computed using trigonometric relations between corresponding features from two stereo views of the workpiece. The model is in the form of a local feature graph. Each node is a (feature-type, position) pair, and arcs connect pairs of nodes when an edge connects the pair of features on the part. Feature types are corners and small holes. Feature position is specified using an object-centered coordinate system.

Features. Local image features include small holes and corners. Corner and small hole detection is based on diameter-limited

gradient direction histograms [Birk et al. 1979] in which intensity variations in several directions and various heuristic thresholds are examined. Detected features from the image are evaluated to eliminate redundant features. The resultant corner points are fine-tuned for accuracy by fitting a pair of lines in an 11×11 window. The intersection of the two lines yields the final corner location. Finally, the interfeature distances between every pair of features are computed. Workpiece examples used in the experiments include simple planar industrial parts and 3-D block objects.

Matching. The matching is carried out by a sequential pairwise comparison algorithm in which a feature point is matched in turn to all model feature points of the same type. The matching process starts with the selection of the feature point that has the highest confidence. The remaining feature points are then matched with all model points. In this step feature type, interfeature distance, and edge information are used as matching criteria, and redundant matched points are deleted. If enough feature points are successfully matched with the model points, and a transformation test, used to eliminate problems due to symmetry, is passed, a match is considered to be found. Finally, the position and orientation of the workpiece are computed from the correspondence between workpiece and model features.

3.3.3 Example 3: Combining Model Graphs Based on Distinctive Focus Features

Model. Bolles and Cain [1982] have developed a sophisticated modeling system for 2-D objects called the local-feature-focus method. Two types of local features are used: corners and regions. An object model consists of three parts. The first is a polygonal approximation of the object's borders. The second is a list of local features, where each is specified by a unique name, its type, position, and orientation relative to the object's centroid, and rotational symmetries about the centroid. Position and orientation values also have associated allowable tolerances. Third, for each distinct feature type, an unambiguous

detector uses neighbor merging to locate circular objects.

Matching. The matching process examines the current information obtained from the scene and the model graphs of objects to propose the next matching step. The model relates features at a coarse resolution with more detailed features at a fine resolution, enabling the matching to be performed using simple features as cues. Given a tentative match between an image component and an object model based on the coarse model features, the fine model features are then successively compared. The object boundary matched at the coarse level determines a tentative match angle of rotation. For a given feature extracted from the image, a measure of the dissimilarity between it and each of the model features is computed. A cumulative dissimilarity measure is kept for each active model. When a model's dissimilarity exceeds a threshold, the model is rejected as a possible match. After the current feature has been compared with each of the remaining candidate models, a next-feature proposer analyzes the features described in these candidate models and proposes the most promising feature among them as the one to be examined next for recognizing the input object.

3.3.2 Example 2: Corner and Hole Relational Models

Model. Chen et al. [1980] estimate the position and orientation of workpieces using the 3-D locations of at least three noncollinear feature points. The location of features is computed using trigonometric relations between corresponding features from two stereo views of the workpiece. The model is in the form of a local feature graph. Each node is a (feature-type, position) pair, and arcs connect pairs of nodes when an edge connects the pair of features on the part. Feature types are corners and small holes. Feature position is specified using an object-centered coordinate system.

Features. Local image features include small holes and corners. Corner and small hole detection is based on diameter-limited

gradient direction histograms [Birk et al. 1979] in which intensity variations in several directions and various heuristic thresholds are examined. Detected features from the image are evaluated to eliminate redundant features. The resultant corner points are fine-tuned for accuracy by fitting a pair of lines in an 11×11 window. The intersection of the two lines yields the final corner location. Finally, the interfeature distances between every pair of features are computed. Workpiece examples used in the experiments include simple planar industrial parts and 3-D block objects.

Matching. The matching is carried out by a sequential pairwise comparison algorithm in which a feature point is matched in turn to all model feature points of the same type. The matching process starts with the selection of the feature point that has the highest confidence. The remaining feature points are then matched with all model points. In this step feature type, interfeature distance, and edge information are used as matching criteria, and redundant matched points are deleted. If enough feature points are successfully matched with the model points, and a transformation test, used to eliminate problems due to symmetry, is passed, a match is considered to be found. Finally, the position and orientation of the workpiece are computed from the correspondence between workpiece and model features.

3.3.3 Example 3: Combining Model Graphs Based on Distinctive Focus Features

Model. Bolles and Cain [1982] have developed a sophisticated modeling system for 2-D objects called the local-feature-focus method. Two types of local features are used: corners and regions. An object model consists of three parts. The first is a polygonal approximation of the object's borders. The second is a list of local features, where each is specified by a unique name, its type, position, and orientation relative to the object's centroid, and rotational symmetries about the centroid. Position and orientation values also have associated allowable tolerances. Third, for each distinct feature type, an unambiguous

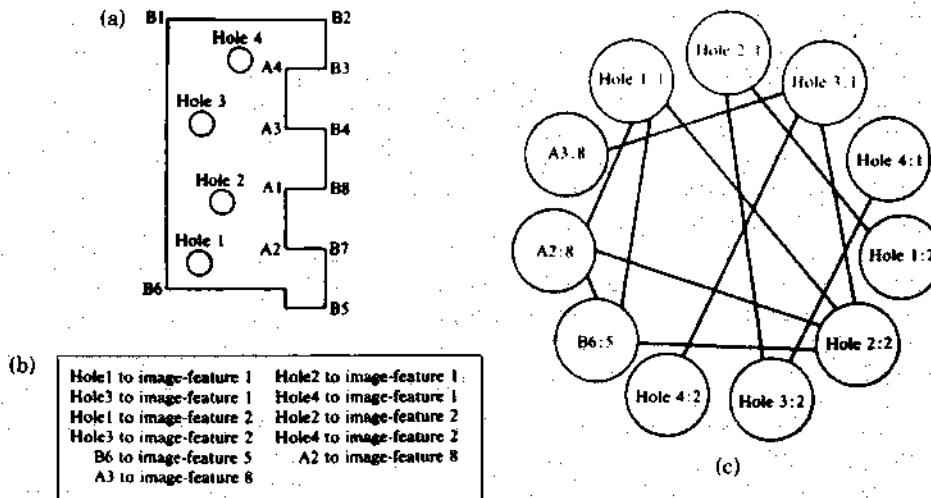


Figure 9. (a) Definitions of the model features of the hinge. (b) List of model-feature-to-image-feature assignments. (c) Graph of pairwise-consistent assignments. Each node represents a possible assignment of a model feature to an image feature. Two nodes are connected if the two assignments they represent are mutually consistent. (From Bolles and Cain [1982].)

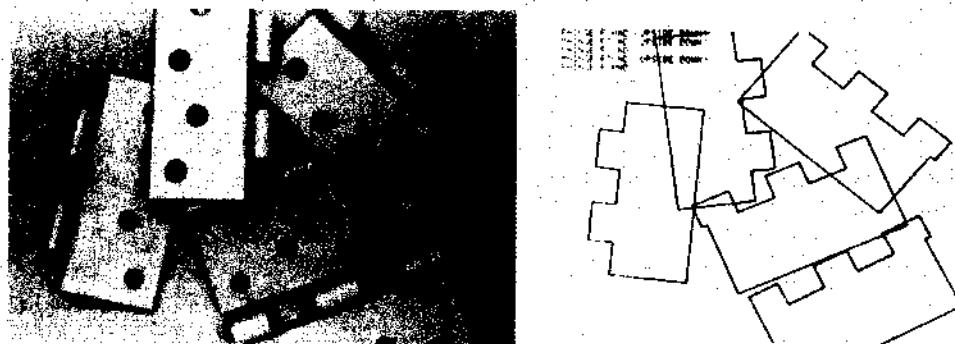


Figure 10. Image of five hinges and the recognition result. (From Bolles and Cain [1982].)

forms clusters of them to hypothesize part occurrences, and finally performs template matches to verify these hypotheses.

After locating all the features found in the image, the system selects one feature (the focus feature) around which it tries to find a cluster of consistent secondary features. If this attempt fails to lead to a hypothesis, the system seeks another potential focus feature for a new attempt. As it finds matching features, it builds a list of possible model-feature-to-image-feature assignments. This list is transformed into a graph by creating a node for each assignment pair and adding an arc between pairs

of nodes referencing the same model; Figure 9 shows the possible assignments and the resulting graph. The result from the first stage of the matching algorithm is used to hypothesize an object. At the final stage, two tests are used to verify the hypotheses by looking at other object features and checking the boundary of the hypothesized object. Figure 10 shows an example.

3.3.4 Example 4: Template Feature Relations

Models. An automatic system for transistor wire bonding has been implemented by Kashioka et al. [1976]. The model



Figure 11. Nine corner templates and the recognition of the circuit position by evaluating relations between pairs of matched templates. (From Kashioka et al. [1976]; © IEEE 1976.)

consists of three sets of three 12×12 binary templates, which are selected by the user from three different orientations of a given prototype chip. For each triple of patterns in a set, an associated distance and direction (relative to the image's x axis) pair is computed from the same binary image of the chip used to define the templates. Chips are assumed to be of a fixed size (camera position above the table is fixed); orientation of a chip is fixed with a tolerance of up to 15 degrees in either direction. It was empirically determined that a triple of templates is a reasonable model for rotations of up to 7 degrees from the normal orientation. Therefore, in order to meet system-orientation specifications, three sets of templates are selected by the user with the prototype chip positioned at orientations -10, 0, and 10 degrees from the normal orientation.

Features. In most of the recognition systems for IC alignment and bonding, multiple template-matching procedures are used. Features used for template matching are distinct patterns such as corners and bonding pads. Relational features, such as the distance and angle between pairs of successfully matched templates, are also used. In most cases these features are extracted by thresholding. The Hitachi transistor wire-bonding system is a typical example of such systems.

Matching. In the multiple template matching of Kashioka et al. [1976], a set of characteristic 12×12 binary templates is

used. The process searches a 160×120 image for the local region which best matches the first template. It then searches for the best match to a second template. From these positions, a distance and a direction angle are computed and compared with the values predetermined from the geometry of the chip. If the measurements are not close to the predefined values, a third local template is used, and measurements are again computed. Locations of bonding pads are computed using the measurements obtained from the multiple local template-matching. Figure 11 shows a set of templates and the matching process.

3.3.5 Other Studies

The SIGHT-I system locates integrated circuit chips by using a set of local templates [Baird 1978]. This model consists of the specification of the possible relative positions of the four corners of a chip. A set of four 4×4 templates is used to evaluate the probability that a corner is present at a given position. A coarse processing stage is applied to the gray-scale image before the relational template-matching step. In this step the approximate orientation of the chip is determined by analyzing the edge orientation histogram to find the most prominent edge orientation. This enables the matching stage to search for corners in known orientations.

Cheng and Huang [1982] have developed a method for recognizing curvilinear objects by matching relational structures. The

boundary of an object is segmented into curve segments and then into chords. Attributes (parallel, symmetric, adjacent, etc.) associated with the chords are used as the nodes in the relational structure representation of the object. Matching is based on a star structure representation of the object [Cheng and Huang 1981]. The recognition of overlapping tools has been shown.

Segen [1983] has developed a method for recognizing partially visible parts by using local features computed from an object boundary. The local features used are defined at points of local maximum and minimum of contour curvature. A local feature from the image is matched with a feature from the model, and they determine a transformation (rotation and translation). All features are used in the matching, and a set of transformations is generated. The algorithm then clusters together features that imply similar transformations. The center of each cluster is used to define a candidate transformation that may possibly give a partial match. Finally, these candidate transformations are tested with a point-by-point matching of the image contour and the transformed model contour.

Westinghouse's gray-level robot vision system uses a simple form of the relational feature graph approach. In one of the reported studies [Schachter 1983], edges are used to form corners where a corner is defined as two intersecting edge vectors. The matching algorithm searches for four edge vectors forming two opposing corners such that the center of the line segment joining the corner pair coincides with the part center. The assumption that the object center and the two opposing corners are collinear restricts the applicability of the algorithm to limited types of industrial parts.

In semiconductor chip manufacturing, each die is visually inspected for the bonding of the die onto the package substrate and the bonding of wires from the die pads to the physically larger package leads. The process involves the recognition of the chip boundary, the determination of the chip position and orientation, and the recognition of bonding pads. Conventionally, human operators have to perform all of these functions. Recently, a number of automatic

die-bonding and wire-bonding systems have been developed for the manufacturing of chips. Most of these systems are based on relational features and associated matching algorithms. Some other IC recognition systems include those of Horn [1975a], Hsieh and Fu [1979], Igarashi et al. [1979], and Mese et al. [1977].

3.4 Comparison of the Three Methods for 2-D Object Representation

On the basis of the above descriptions of 2-D object-recognition algorithms, the following general conclusions can be made about global feature, structural feature, and relational graph methods. A summary of this comparison is shown in Table 2.

Features used in the global feature method are easy to compute from binary images, and their ordering in the model is unimportant. This makes the training process a relatively simple task. Features can be computed in real-time from, for example, a run-length encoding of the image. This method also has the advantage that the features can often be simply defined to be shift and rotation invariant. That is, objects may be placed at any position and orientation, and the camera geometry does not have to be fixed. In addition, optimal matching accuracy can be achieved by using standard statistical pattern-recognition techniques. The main disadvantage of global feature methods is the assumption that almost all of the objects must be visible in order to measure these features accurately. Thus, objects are not allowed to touch or overlap one another or contain defects. Unless the environment can be sufficiently controlled to eliminate these conditions, we are not likely to find global features because they are so large (e.g., due to occlusion).

The structural feature method is an improvement over the global feature method in terms of capability and robustness, but its complexity requires more sophisticated training and matching processes. This makes it computationally more expensive. Local and extended boundary features are used to represent smoothed, intermediate-level symbolic descriptions. Since gray-level images are generally used, the

Table 2. Comparison of the (a) Global Feature, (b) Structural Feature, and (c) Relational Graph Methods

Global feature method	Structural feature method	Relational graph method
Features are functions of the entire silhouette; controlled environment for binary image processing is required	Features describe local properties; binary image processing is not required	Features describe local properties; binary image processing is not required
Unable to handle noisy images	Able to handle noisy images by gray-level processing	Able to handle noisy images by gray-level processing
Global features are relatively easy and inexpensive to extract	Feature extraction is expensive when compared with the other two methods; it involves the transformation of local features into abstracted representations	Feature extraction is less expensive than in the structural feature method; local features are used directly in the model
Extracted features are invariant to rotation, shift, and size	Extracted features do not have the invariance properties	Local and relational features are not invariant to rotation, shift, or size
The training (modeling) process is simple, involving the generation of an unordered feature list	Modeling involves the generation of a structured feature list which describes the object's outline; it is relatively straightforward	Modeling involves the generation of a graph which relates all chosen local features; it requires carefully thought-out strategies
Matching involves statistical pattern-recognition schemes; optimal matching accuracy is achievable	Matching involves trial-and-error (hypothesize-verification) procedures	Matching involves graph-searching procedures
Matching is fast if a small number of features is used	Matching is a sequential process; slow if a large number of hypotheses is needed	Matching is slow if the model graph is complex
Unable to handle occlusion	Able to handle occlusion if a significant portion of the outline is apparent	Able to handle occlusion if key features of the object are apparent

resulting features are more reliable than the features extracted from a thresholded image. Methods using image enhancement and feature smoothing and abstraction (e.g., using the best fitting line or curve to represent a boundary segment) lead to systems that are much more flexible and less sensitive to noise than the methods using simple global image features. Of course, because many local features are used to model an object, the search procedure used for matching image features with model features must avoid testing all possible correspondences. Another difficulty is that boundary features are not usually invariant under translation and rotation. Consequently, matching usually consists of a sequential procedure that tentatively locates a few local features and then uses them to constrain the search for other features. This hypothesis-verification procedure will become very time consuming if the model is not appropriately designed. Unlike the global feature method, partial occlusion of

objects can be handled. However, a significant portion of an object's boundary has to be unobscured for successful recognition because most features are derived from the boundary.

The relational graph method further relaxes the requirements of how an object has to be presented for successful recognition. Since the model contains both local and relational features in the form of a graph, matching does not depend only on the presence of certain boundary features, but also on other features and properties of their interrelations (e.g., distance). Each local feature provides a local cue for the recognition of objects that overlap each other. The only requirement for successful recognition is that a sufficient set of key local features has to be visible and in the correct relative positions. When this method is compared with the global feature and structural feature methods, the design of the model and the matching procedure are more complex and of increasing impor-

tance. The matching procedure involves graph-searching techniques that are computationally intensive and too slow without special-purpose hardware for many industrial applications. Hierarchical graph-searching techniques (e.g., Barrow and Tenenbaum [1981]) can reduce the time complexity of the matching process by decomposing the model into independent components.

4. $2\frac{1}{2}$ -D SURFACE REPRESENTATIONS

The previous section presented methods based on image intensities, deriving features from gray-level or binary images to represent the projection of an object in two dimensions. In this section we present another class of methods, which is also viewer centered, but which is based on physical-scene characteristics of a single view of an object. This representation maintains information in register with the original gray-scale image and includes *intrinsic images* [Barrow and Tenenbaum 1978], *$2\frac{1}{2}$ -D sketch* [Marr 1978], *needle map* [Horn 1979], *parameter images* [Ballard 1981b], and *surface-orientation map* [Brady 1982a]. Intrinsic scene parameters include surface range, orientation, discontinuities, reflectance, illumination, color, and velocity. Since this local information is obtained over a whole region within some boundaries, it is more robust than the edge-based techniques used with many of the 2-D representations discussed in the previous section.

All of the methods in this section use scene surface properties derived from a single viewpoint to define features and construct models. If multiple views of an object are required, each is modeled independently. We have included range maps as part of this class of representation despite the fact that 3-D data are used. This is because the models that use these data are viewer centered and emphasize the description of observable surface features from a single viewpoint. Models that describe a complete (viewpoint-insensitive) 3-D object are included in the next section as 3-D representations.

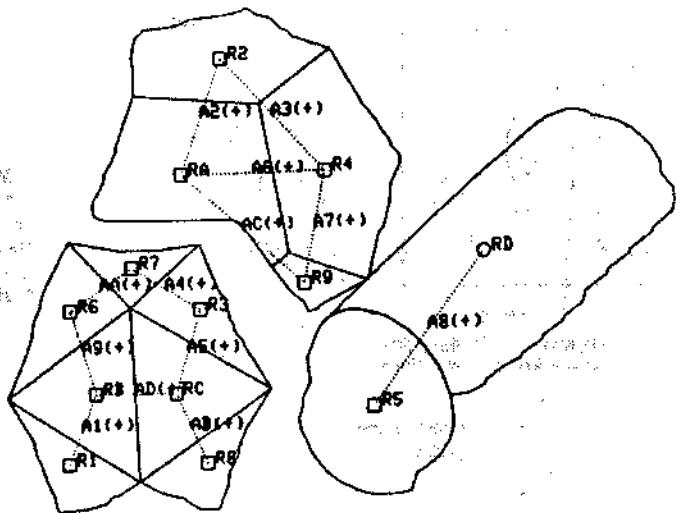
Most current research is focusing on the problem of how to compute these intrinsic surface maps. See, for example, Ballard and Brown [1982], Barrow and Tenenbaum [1978], Brady [1982a], Jarvis [1983a], and Marr [1982], for surveys of many applicable techniques. The present survey does not consider this "measurement" stage.

Of particular interest for applications in industrial-parts recognition is the computation and use of range maps and local surface-orientation (needle) maps. Jarvis [1983a] and Poje and Delp [1982] give recent overviews of range-finding techniques using both active and passive methods. Active methods include ultrasonic and light time-of-flight measurement, and structured light projection using a plane or grid of light. Although early methods of these types have been slow, expensive, and of low accuracy, many recent improvements have been made [Agin and Highnam 1982; Altschuler et al. 1981; Jarvis 1983b; Kanade and Asada 1981; Pipitone and Marshall 1983; Popplestone et al. 1975].

Instead of extracting a range map, other researchers are focusing on obtaining local surface orientation as a descriptor of surface shape. This includes such direct methods as shape from shading [Horn 1975b], shape from texture [Bajcsy 1973; Bajcsy and Lieberman 1976; Kender 1980; Stevens 1981; Witkin 1981], and shape from photometric stereo [Woodham 1978].

One method of computing surface orientation that shows considerable promise for industrial-parts recognition is called photometric stereo [Woodham 1978]. Local surface orientation is computed using a reflectance map for each of three different incident illumination directions, but from a single viewing direction. Since an object point corresponds to the same pixel in each of these three gray-level images, the surface orientation at this point can be obtained from the intersection of isobrightness contours in the reflectance maps associated with each light source. The method has been implemented very efficiently by inverting the reflectance maps into a lookup table which gives surface orientation from a triple of gray levels [Silver 1980]. So far

Figure 12. Relational graph description of planar and curve surfaces derived from a range map. (From Oshima and Shirai [1983]; © IEEE 1983.)



the technique has been defined for objects containing Lambertian and specular surfaces [Ikeuchi 1981a; Woodham 1978], and error analysis has been performed [Ray et al. 1983].

To date, researchers have developed only a few model-based recognition systems predicated on features derived from one or more surface maps. Hence application to industrial parts-recognition has yet to be extensively investigated. In the remainder of this section we present some of the techniques that have been studied. All of these methods are based on features derived from either a range map or a needle map. We expect that considerable future work will be devoted to expanding this class of techniques.

4.1 Example 1: A Relational Surface Patch Model

Model. Oshima and Shirai [1983] construct a relational-feature graph in which nodes represent planar or smoothly curved surfaces extracted from a range map, and arcs represent relations between adjacent surfaces. Surface types include planar, ellipsoid, hyperboloid, cone, paraboloid, cylinder, and others. For each pair of adjacent regions, the type of intersection (convex, concave, mixed, or no intersection), angle between the regions, and relative positions of the centroids are stored. Figure 12 illustrates this relational-graph description for

a scene containing three objects. If objects may be viewed from multiple viewing positions, then a separate relational graph must be constructed for each view, and these models must be treated independently by the matcher. Partial occlusion of certain secondary planar surfaces is allowed, although the extent is dependent on the pre-defined thresholds used by the matcher. Currently, curved surfaces may not be occluded in the scene.

Features. A range map is used as the basis for segmenting an image into regions. First, connected points with similar range values are grouped into small surface elements. Next, the equation of the best plane surface through each of these elements is computed, and then these surface elements are merged into maximal planar and curved regions. For each region, a set of global features is computed, including surface type (planar, ellipsoid, cone, cylinder, etc.), number of adjacent regions, area, perimeter, compactness, occlusion, minimum and maximum extent, and mean and standard deviation of radius.

Matching. Matching is performed by comparing an observed relational graph of surface descriptions with a set of graphs for each viewpoint of each object modeled. First, regions corresponding to maximal smooth surfaces are extracted from the range map of a given scene. A *kernel* con-

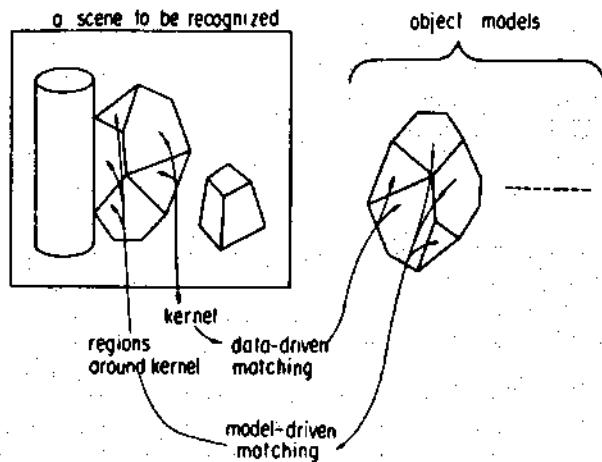


Figure 13. Matching kernel surfaces in a scene with model surfaces is used to select candidate models. Neighboring surfaces are then matched in order to verify a candidate model. (From Oshima and Shirai [1983]; © IEEE 1983.)

sisting of either a single region or a pair of adjacent regions is then selected from the surface descriptors of the given scene in order to guide the search for matching model graphs. The kernel represents regions with high confidence of being found; criteria include no occlusion, planar surfaces, and large region area. Next, an exhaustive search of all model graphs is performed, selecting as candidate models all those that contain regions which match the kernel. Finally, the system performs a depth-first search which attempts to determine the correspondence between each remaining region in the current candidate model and the regions extracted from the scene. A model region and a scene region match if their region properties are similar, all adjacencies to previously matched regions are consistent, and the properties of all new relations between regions are similar. This process is repeated for other kernel regions in the scene until a globally consistent interpretation, in which each scene region is found to correspond to exactly one model region, is achieved. If multiple consistent interpretations are possible, then the system returns each one. Figure 13 illustrates this matching process.

4.2 Example 2: A Relational Surface Boundary Model

Model. Nevatia and Binford [1977] construct a relational graph description of each view of an object using a set of *generalized*

cones corresponding to elongated subparts of the object. In particular, cones are represented as *ribbonlike* descriptors containing 2-D cross-sections of range discontinuity points [Brooks 1979]. Given a set of these ribbons defining an object, a set of *joints* is constructed indicating which ribbons are adjacent to each other. A joint description includes an ordered list of ribbons connected to it and a designated dominant ribbon having the largest width. A relational graph is constructed; in this graph joints are represented by nodes and ribbons by arcs. Figure 14 shows the spines of the ribbons detected in a scene containing a reclining doll and the resulting relational graph description. In addition, a set of coarse descriptors is associated with this object graph, including number of ribbons, number of elongated ribbons, number of joints, bilateral symmetries, and a set of *distinguished* ribbons having the largest widths. For each distinguished piece of an object a three-bit description code is used to permit efficient organization and search of the set of models. The three descriptors encoded are the part's connectivity, type (long or wide part), and conicity, that is, whether or not it is conical. Models are sorted by their description code and secondarily by the maximum number of parts attached at either end of the distinguished piece.

Features. As an alternative to extracting planar and curved surfaces from range maps, some researchers have developed

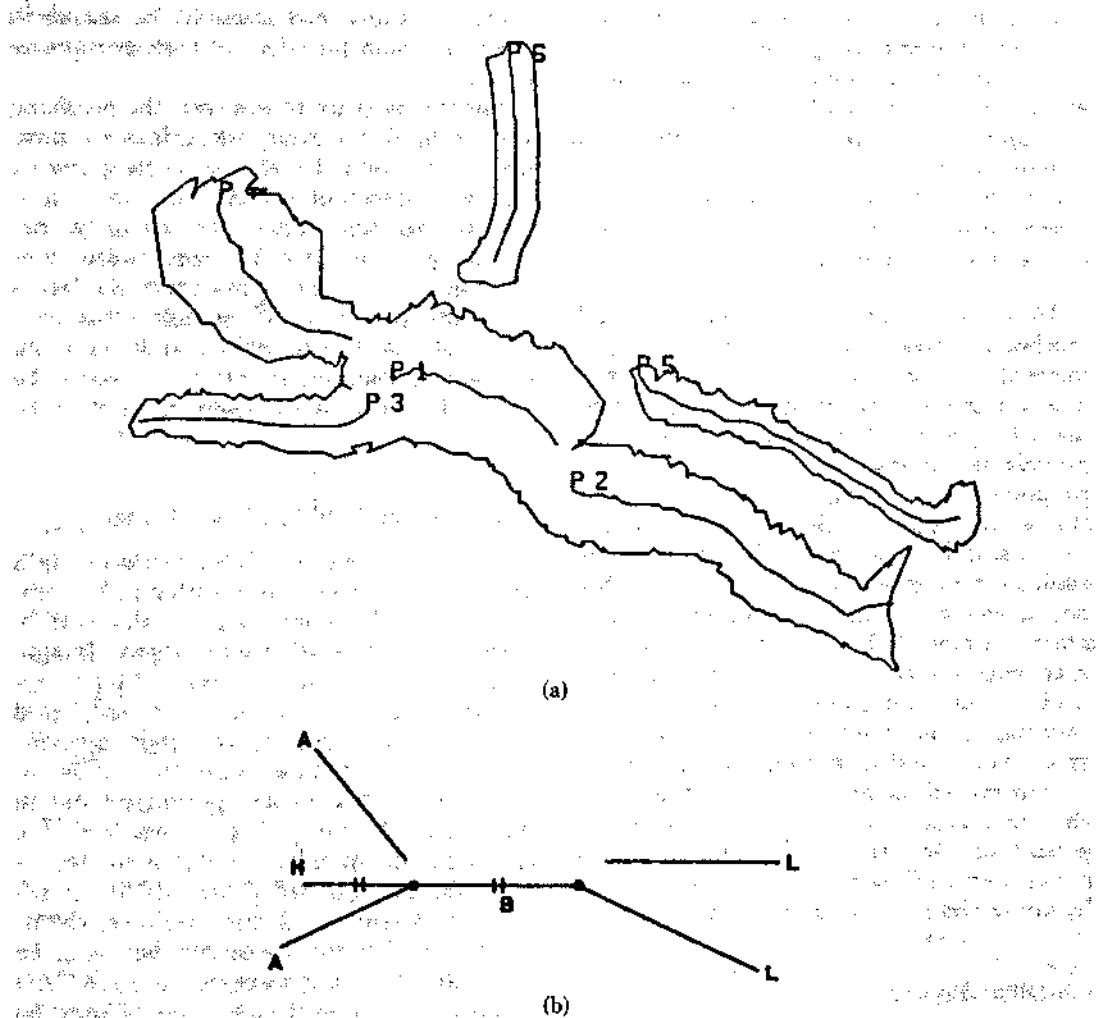


Figure 14. (a) Spines of ribbons detected in a range map for a scene containing a reclining doll. Range discontinuities are used to define the boundary of the object. (b) The relational graph constructed from (a). (From Nevatia and Binford [1977].)

techniques for detecting surface boundaries by detecting and linking points at which range discontinuities occur. Nevatia and Binford use a range map to derive a boundary description of a given view of an object. Rather than directly use this global structural feature to describe an object, they immediately construct a relational graph using ribbonlike primitives to describe subparts in terms of the 2-D projections of the boundaries [Brooks 1979]. A ribbon is the 2-D specialization of a 3-D generalized cylinder [Binford 1971]. Ribbons are specified by three components: spine, cross-section, and sweeping rule. By sweeping a planar

cross-section at a constant angle along a spine according to the sweeping rule, a planar shape is generated. Nevatia and Binford first construct a set of local ribbons restricted to having straight axes in eight predefined directions and smoothly varying cross-sections. This is done by linking midpoints of cross-sections (runs of object points perpendicular to the axis direction) that are adjacent and have similar cross-sections. These local ribbons are then extended by extrapolating the axes of the local ribbons and constructing new cross-sections. This process allows the resulting axis to curve smoothly. In general, a single

part of an object may be described by (part of) several overlapping ribbons. To reduce this redundancy, ribbons that are not so elongated or rectangular as other ribbons overlapping them are deleted. Each ribbon is associated with a crude description of its shape given by its axis length, average cross-section width, elongatedness, and type (conical or cylindrical).

Matching. As in the 2-D relational graph methods, matching involves comparing parts of the relational graph extracted from a given scene with each relational graph describing a model. First, a set of candidate models is determined by comparing the properties of the distinguished ribbons in the scene with those distinguished ribbons associated with each of the models. For each such candidate model a finer match is performed by comparing other ribbons, pairing a model ribbon with a scene ribbon if their properties are similar and all connectivity relations are consistent in the current pair of matched subgraphs. The scene graph is allowed to match a model graph, even if not all model ribbons are present in the scene graph (hence partial occlusion is permitted), but the scene graph may not contain extra ribbons that are not matched by any ribbon in the model graph.

4.3 Other Studies

Many researchers have investigated using range maps as the basis for segmenting an image into regions by grouping (merging) points into planar surfaces, cylindrical surfaces, surfaces on generalized cylinders, and other smoothly curved surface patches [Agin and Binford 1976; Bolles 1981; Bolles and Fischler 1981; Duda et al. 1979; Henderson 1982; Henderson and Bhanu 1982; Milgram and Bjorklund 1980; Oshima and Shirai 1979; Popplestone et al. 1975; Shirai 1972]. Alternatively, range maps can be segmented by locating discontinuities in depth. For example, Sugihara [1979] segments a range map by finding such edges. To aid this process, a *junction dictionary* is precomputed listing all possible ways junctions can occur in range maps for scenes containing only trihedral objects. The dic-

tionary is then used to guide the search in the range map for edges of the appropriate types.

For the most part, however, the resulting surface and boundary descriptions have not been used to define corresponding viewer-centered object models and matching techniques. This is primarily because object-centered 3-D models are more concise and natural representations than a set of independent $2\frac{1}{2}$ -D models. Of course, many of the 3-D modeling and matching methods presented in Section 5 could be adapted and used for each distinct viewpoint.

5. 3-D OBJECT REPRESENTATIONS

If we assume that an object can occur in a scene at an arbitrary orientation in 3-space, then the model must contain a description of the object from all viewing angles. Image-space (2-D) and surface-space ($2\frac{1}{2}$ -D) representations are viewer centered, and each distinct view is represented independently. Thus, when multiple views of complicated objects are permitted (as in the general bin-picking problem), a viewpoint-independent, volumetric representation is preferred [Marr 1978]. In addition, in an industrial automation environment in which the vision system must be integrated with objects represented in CAD databases, an object-space model may be convenient because of its compatibility. In contrast to the previous representations, a single model is used to represent an object, implicitly describing all possible views of the object.

Researchers have investigated two main types of 3-D representations. These are (1) *exact representations* using surface, sweep, and volume descriptions; (2) *multi-view feature representations* in which a set of 2-D or $2\frac{1}{2}$ -D descriptions are combined into a single composite model. This includes the specification of a set of topologically distinct views or a uniformly sampled set of 2-D viewpoints around an object. The first representation method completely describes an object's spatial occupancy properties, whereas the second only represents selected visible 2-D or $2\frac{1}{2}$ -D surface

features (and sometimes their 3-D spatial relationships).

Exact representations include the class of complete, volumetric methods based on the exact specification of a 3-D object using either surface patches, spines and sweeping rules, or volume primitives. Object-centered coordinate systems are used in each case. See, for example, Badler and Bajcsy [1978], Ballard and Brown [1982], and Requicha [1980] for a general introduction to this class of representations. Surface model descriptions specify an object by its boundaries or enclosing surfaces using primitives such as edge and face. Baumgart's [1972] "winged edge" representation for planar polyhedral objects is an elegant example of this type of model. Volume representations describe an object in terms of solids such as generalized cylinders, cubes, spheres, and rectangular blocks. The main advantage of this class of representations is that it provides an exact description that is object centered. The main disadvantage is that it is difficult to use in a real-time object-recognition system since the processing necessary to perform either 2-D to 3-D or 3-D to 2-D projections (for matching 2-D observed image features with a 3-D model) is very costly. For example, in the ACRONYM system [Brooks and Binford 1981] camera constraints are built in so as to limit the number of 3-D to 2-D projections that must be hypothesized and computed at run time.

Multiview feature representation can include the work on storing 2-D descriptions for each stable configuration of an object. We restrict our discussion here to coordinated representations of multiple views that permit the specification of efficient matching procedures that take advantage of intraview and interview feature similarities.

One class of multiview representations is based on the description of the *characteristic views* of an object. This requires the specification of all topologically distinct views. Koenderink and vanDoorn [1976a, 1976b, 1979] are studying the properties of the set of viewing positions around an object, and the qualitative nature of the stability of most viewing positions. That is,

small changes in viewing position do not affect the topological structure of the set of visible object features (i.e., point and line singularities). On the basis of the topological equivalence of neighboring viewpoints, they define an "aspect graph" of feature-distinct viewpoints (see Figure 15).

Fuchs et al. [1980] have also used this idea to perform a recursive partitioning of a 3-D scene using the polygons that describe the surfaces of the constituent 3-D objects. That is, a *binary space-partitioning tree*, in which each node contains a single polygon, is constructed. Polygons associated with a node's left subtree are those contained in one half-space defined by the plane in which the current polygon lies; the polygons in the right subtree are the ones in the other half-space. Using this structure they perform hidden surface elimination from a given viewpoint by a simple in-order tree traversal, in which subtrees are ordered by their "visibility" from the given viewpoint. In this representation each leaf defines a characteristic view volume; hence the set of leaf nodes defines a partition of 3-space into distinct viewing positions.

Another type of multiview representation, the *discrete view-sphere representation*, is the "viewing sphere" of all possible viewpoints (at a fixed distance) around an object, storing a viewer-centered description for each sample viewpoint. This can be precomputed from a complete 3-D volumetric description and provide a description that is compatible with the features extracted from a test image at run time. Thus it is a more convenient representation, and yet it provides sufficient accuracy of description, except at pathological viewing positions.

5.1 Example 1: A Surface Patch Graph Model

Model. Shneier [1979, 1981] constructs 3-D surface models from a set of light-stripe images of the object to be modeled. Each distinctly different plane surface that is extracted is represented by a unique node in a *graph of models*, which describes all models to be recognized. Associated with each node is a set of properties that describes the surface's shape and a set of

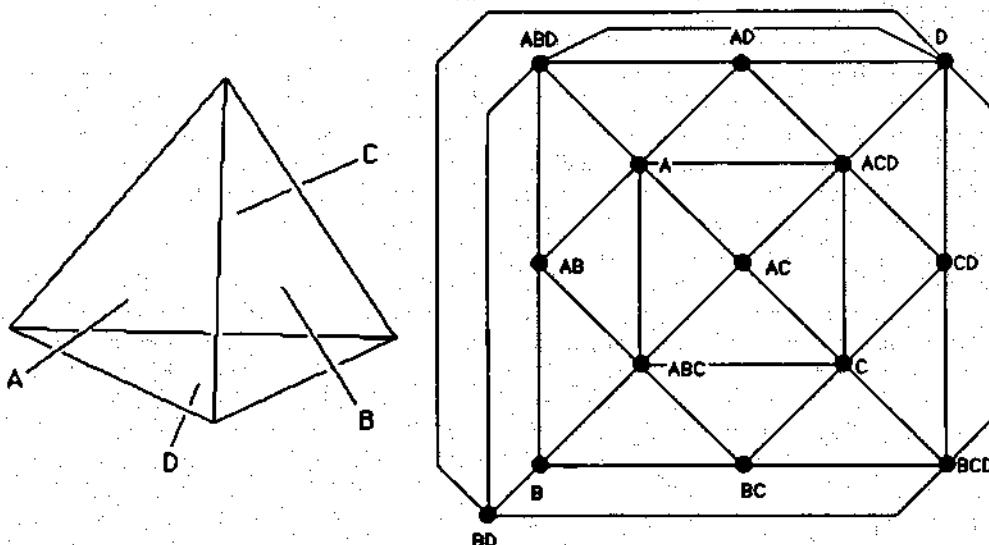


Figure 15. The aspect graph for a tetrahedron. Nodes are of three types representing whether one, two, or three faces are visible. Arcs connect a pair of nodes when some arbitrarily small change in viewing direction suffices to change the set of visible faces from the faces visible at one node to those visible at the other node without going through any intermediate set of visible faces.

pointers to the names of the models of which this primitive shape is a part. Thus, if two surface shape descriptions are similar with regard to the same or different objects, they are represented by a single node in the graph. Arcs connect pairs of nodes using a set of predefined relation schemata (e.g., the "is adjacent to" relation). Arguments to relation schemata are surface descriptions, not actual surfaces. Relation schemata also index the models in which they occur and the primitives that form their arguments. Thus nodes and arcs in the graph of models may be shared within models and across models. This integration of multiple object models into a single graph has the advantages of being very compact and enabling a rapid indexing scheme to be used.

Features. Planar surfaces are determined from a set of light-stripe images using techniques described in Section 4.

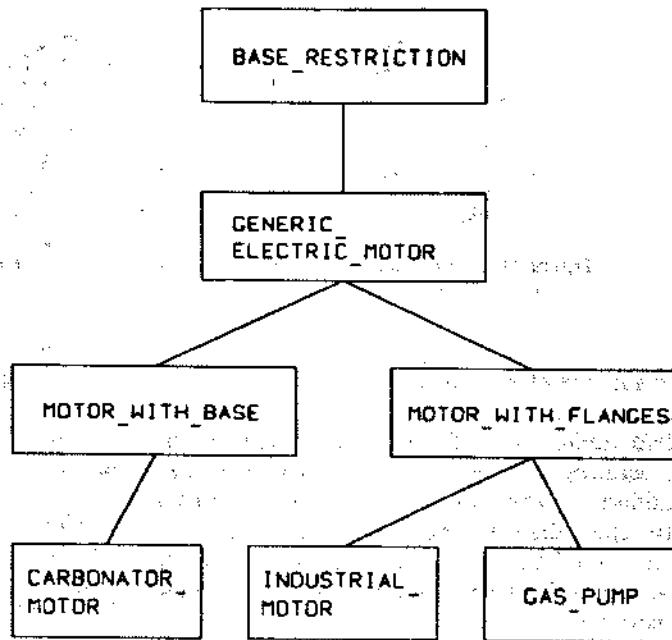
Matching. The set of observed planar surfaces extracted from a given scene are matched with the graph of models for all possible objects using a two-step procedure. First, for each observed surface that is sufficiently similar to a node in the graph of models, a node is created in the *scene graph*

indicating this match. Since each node in the graph of models corresponds to one or more surfaces in one or more objects, each possibility is tested using a predefined set of procedures. These procedures decide whether an interpretation is possible for the observed surface and assign confidences to these interpretations. A subgraph of the scene graph is created for each possible interpretation, and each surface/model-node pair is assigned to one or more such subgraphs. Next, the scene graph is traversed, deleting surfaces that are insufficiently substantiated and propagating constraints in order to remove multiple interpretations for a single surface.

5.2 Example 2: Hierarchical Generalized Cylinders

Model. Brooks' ACRONYM system constructs *sweep* models using part/whole hierarchical graphs of primitive volume elements described by generalized cylinders [Binford 1971; Brooks 1983a, 1983b; Brooks and Binford 1981]. A *generalized cylinder* (GC) describes a 3-D volume by sweeping a planar cross-section along a space-curve spine; the cross-section is held

Figura 16. The restriction graph for the classes of electric motors used in ACRONYM. (From Brooks [1983a].)



at a constant angle to the spine, and its shape is transformed according to some sweeping rule. The user constructs a tree for each object, where nodes include GC descriptions and arcs indicate the subpart relation. The tree is designed to provide a hierarchical description of an object, where nodes higher in the tree correspond to more significant parts in the description. For example, the root of an "electric motor" tree describes the cylinder for the large cylindrical body of the motor. Arcs from this node point to nodes describing cylinders for the small flanges and spindle, which are part of a lower priority level of description of the motor.

Each GC has its own local coordinate system, and additional affixment arcs between nodes specify the relations between coordinate systems. If multiple parts of the same type are associated with a single object, they are represented by a single node in the tree with a quantity value and a set of coordinate transformations specifying the location of each part. Furthermore, in order to allow variations in size, structure, and spatial relationships in GC descriptions, any numeric slot in a node's description may be filled by an algebraic expression ranging over numeric constants and variables. Classes of objects are speci-

fied by constraints (i.e., inequalities on algebraic expressions that define the set of values that can be taken by quantifiers).

A scene is modeled by defining objects and affixing them to a world-coordinate system. A camera node is also included, specifying bounds on its position and orientation relative to the world-coordinate system.

To aid the matching of models with image features, the user constructs from the static object graph a model class hierarchy called the restriction graph. That is, the sets of constraints on quantifiers in the object graph are used to build a specialization hierarchy of different classes of models. The root node represents the empty set of constraints for all restriction graphs. A node is added as the child of another node by constructing its constraint list from the union of its parent's constraints and the additional constraints needed to define the new node's more specialized model class. The volumetric structure associated with a node is retrieved indirectly by a pointer from the node to the object graph. An arc in the graph always points from a less restrictive model class (larger satisfying set of constraints) to a more restrictive one (smaller satisfying set). Figure 16 illustrates a restriction

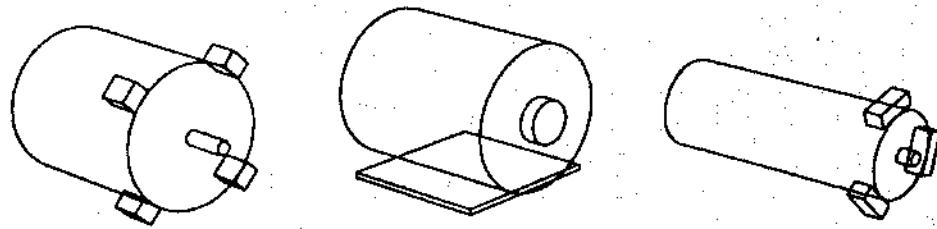


Figure 17. Three instances of the model classes associated with the three leaf nodes in Figure 16 (From Brooks [1983a].)

graph for classes of motors, and Figure 17 shows three instances associated with the leaf nodes' sets of constraints. During the matching process, other nodes are also added to the restriction graph in order to specialize further a given model for case analysis, or to specify an instance of a match of the model to a set of image features.

Features. ACRONYM uses ribbons and ellipses as low-level features describing a given image. A *ribbon* is the 2-D analog of a 3-D generalized cylinder. In particular, Brooks considers the special case in which a ribbon is defined by sweeping a symmetric width line segment normally along another straight-line segment while changing the width of the first segment linearly with distance swept. Ellipses are used to describe the shapes generated by the ends of GCs. For example, ellipses describe ends of a cylinder and ribbons describe the projection of the cylinder body.

The extraction of these features is performed by the descriptive module of the ACRONYM system [Brooks 1979]. First, an edge-linking algorithm creates sets of linked edges (contours) from the image data. Linking edges into a contour is formulated as a tree-searching problem searching for the best edge direction at a given point. A contour is retained only if it satisfies certain global shape criteria. Next, an algorithm fits ribbons and ellipses to the sets of contours by extracting potential boundary points of a ribbon from a histogram of the angles of the edge elements making up the contour. Finally, redundant ribbons in a single area of the image are removed. A graph structure, the *observation graph*, is the output of the descriptive mod-

ule. The nodes of the graph are ribbon and ellipse descriptions, and the arcs linking the nodes together specify spatial relations between ribbons.

Matching. ACRONYM predicts appearances of models in terms of ribbons and ellipses that can be observed in an image. Rather than make exhaustive predictions based on all possible viewing positions, viewpoint-insensitive symbolic constraints are used. These indicate features that are invariant or quasi-invariant over a large range of viewing positions. To generate predictions, a rule-based module is used to identify contours of model faces that may be visible. Case analysis is used to restrict predictions further and produce predicted contours in the viewer's coordinate system.

As a result of this constraint-manipulation process, a *prediction graph* is built. In this graph nodes either represent specific image features or join prediction subgraphs containing lower level features. Arcs of the graph denote image relations between features, relating multiple feature shapes predicted for a single GC. Arcs are labeled either "must be," "should be," or "exclusive." Associated with a prediction graph is a node in the restriction graph that specifies the object class being predicted.

Matching is performed at two levels. First, predicted ribbons must match image ribbons, and second, these "local" matches must be globally consistent. That is, relations between matched ribbons must satisfy the constraints specified in the arcs of the prediction graph, and the accumulated constraints for each maximal subgraph matched in the observation graph must be consistent with the 3-D model constraints in the associated restriction node. Local

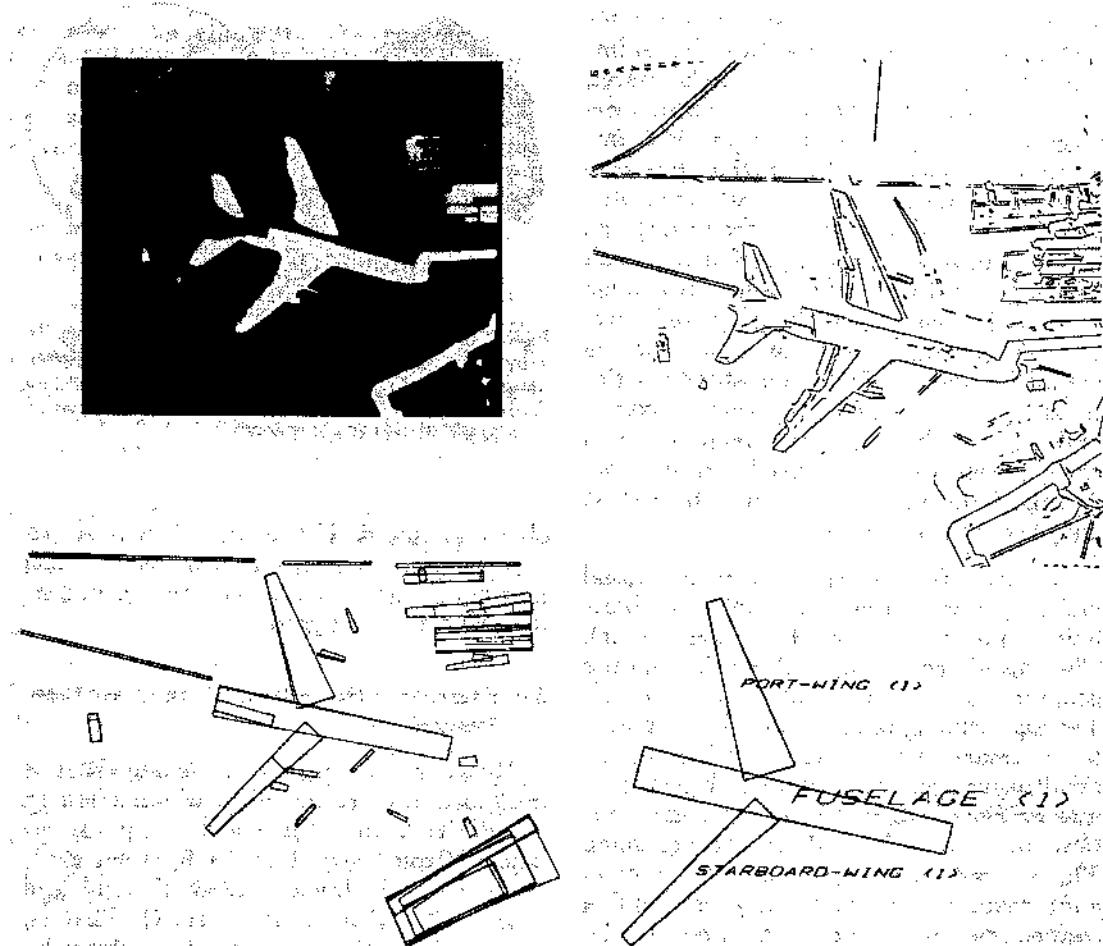


Figure 18. Results of Brooks' matching procedure. The first figure shows the output of the edge detector, the second figure shows the output of the ribbon finder. The final figure is the output of the matcher. (From Brooks [1983b]; © IEEE 1983.)

matches of predicted ribbons with image ribbons also provide additional "back constraints" which are used to further restrict model parameters. Finally, matching is first done for GCs of highest priority in each model's object-graph hierarchy in order to limit the search initially to include only the most important parts. Figure 18 illustrates the results of this method.

5.3 Example 3: Multiview Feature Vectors

Model. Goad [1983] builds a multiview feature model of an object by constructing a list of object features and the conditions under which each is visible. The single object feature used is a straight-line segment representing a portion of the object's sur-

face at which either a surface normal or a reflectivity discontinuity occurs.

The set of possible viewing positions is represented by partitioning the surface of a unit viewing sphere into small, relatively uniform-size, patches. The current implementation uses 218 patches. To represent the set of positions from which a given edge feature is visible, a bit-map representation of the viewing sphere is used to encode whether or not the feature is wholly visible from each patch on the sphere (i.e., a line's projection is longer than a threshold). Thus each feature is stored as a pair of endpoint coordinates plus 218 bits to describe its visibility range.

The matching procedure used with this model requires a sequential enumeration

of model edges which are successively matched with image edges. In order to improve the run-time efficiency of the search for a consistent set of matches (which determines a unique view position), it is important to select an order that presents edges in decreasing order of expected utility. This can be done by preprocessing the list of features in the model using each edge's (a) likelihood of visibility, (b) range of possible positions of the projected edge, and (c) focusing power (i.e., if a match is made, how much information about restrictions on the camera position becomes known). Combining these factors for a given model results in a predetermined ordering of the best edge to match next at any stage of the search.

Features. Goad restricts his model-based vision system to the detection of straight-line segments (straight edges on an object). The edge detection algorithm is based on a program developed by Marimont [1982]. The algorithm applies a Laplacian operator to the image, detects zero crossings in the resulting image, and then links these points into extended edges followed by segmentation into a sequence of smooth contours. Three different types of objects, a universal joint casting, a keyboard key cap, and a connecting rod, have been used in the experiments.

Matching. A sequential matching procedure with backtracking is implemented in Goad's system. The matching involves a search for a match between image and model edges. At any given time in the search, a hypothesis about the position and orientation of the object relative to the camera is used to restrict the search area to some reasonable bounds. The hypothesis is refined sequentially during the matching process.

The procedure starts with predicting the position and orientation of the image projection based on the current hypothesis. Then, a model edge is selected to match with image edges. If a match is found, the measured location and orientation of the new edge are used to update the hypothesis. The algorithm repeats the searching and updating until a satisfactory match of an

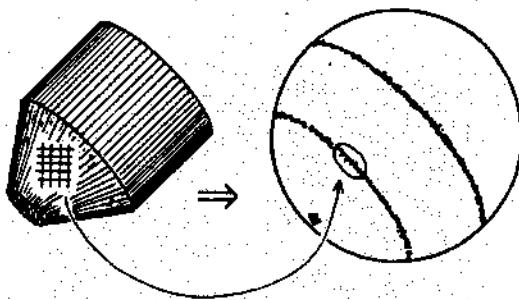


Figure 19. Representation of an object (left) by its extended Gaussian image (right) in which discrete patches of the object are mapped onto points on the Gaussian sphere based on the surface orientation of each patch. (From Horn [1984].)

object is found. If the algorithm fails to locate a predicted edge, it backtracks to use another image edge that has also been predicted as a good match.

5.4 Example 4: Multiview Surface Orientation Features

Model. Horn and his colleagues use a multiview feature model in which features are derived from the needle map for an object [Brou 1984; Horn 1979; Horn 1984; Ikeuchi 1981b; Ikeuchi 1983; Ikeuchi and Shirai 1982; Ikeuchi et al. 1984]. That is, they model each viewpoint of an object by the distribution of its surface-orientation normals on the Gaussian sphere, ignoring positional information by moving all surface normals to the origin. By associating a unit of mass with each point on the unit sphere, we obtain a distribution of mass called the "extended Gaussian image" (EGI) [Horn 1984; Smith 1979]. Segments of developable surfaces (such as planes and cylinders) map into high concentrations of points in known configurations. Figure 19 illustrates this representation for a simple object. A 3-D object is then modeled using a set of (normalized) EGIs, one for each possible viewing direction on a uniformly sampled viewing sphere [Ikeuchi 1981b, 1983; Ikeuchi and Shirai 1982; Ikeuchi et al. 1984]. More specifically, a two-dimensional table is constructed for each possible (viewpoint, mass-distribution) pair. An element in this table stores

the mass (surface area) corresponding to the total surface description for the given viewpoint. If multiple objects are to be recognized, then a table is constructed for each object.

Features. The complete surface-orientation map in the form of the normalized EGI is used as a global-feature descriptor.

Matching. Matching is performed by comparing an observed EGI with each model EGI. To constrain the set of match tests that must be made for each pair, the observed EGI and model EGI mass centers are aligned, constraining the line of sight. Next, the observed and model spheres are rotated about the candidate line of sight so as to align their directions of minimum EGI mass inertia. These two constraints completely specify the alignment of the observed EGI with a model EGI. A match measure for a given pair of normalized EGIs is specified by comparing the similarity in their mass distributions; the model that maximizes this measure is the estimate of the observed line of sight. When multiple objects are present in a single scene, it is first necessary to segment the surface-orientation map into regions corresponding to separate objects.

5.5 Other Studies

The principle features used in most 3-D recognition systems are based on surface properties such as faces, edges, and corners. The references given in Section 4.3 for grouping range data into planar, cylindrical, and other smoothly curved surfaces are also used for 3-D surface description and modeling. Potmesil [1983] constructs 3-D surface models from a series of partially overlapping range images by an iterative merging algorithm which first groups local surface patches into locally smooth surface sheets (using a quadtree representation) and then merges partially overlapping surface representations using a heuristic search procedure.

Bolles et al. [1984] use a surface model as the primary structure for generalizing their local-feature-focus method (see Section 3.3.3) to 3-D using a system called

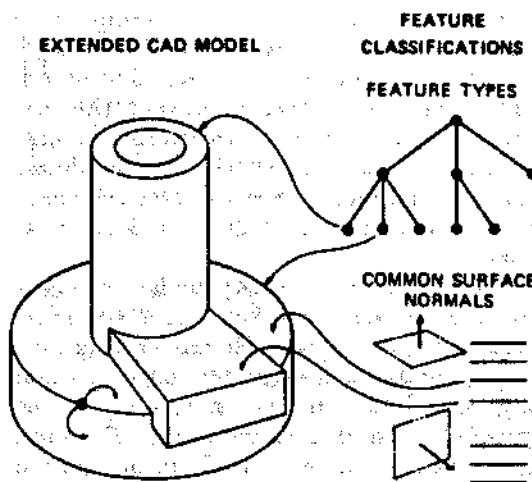


Figure 20. 3DPOs-augmented CAD model and feature classification network. (From Bolles, R. C., Horaud, P., and Hannah, M. J. 1984. 3 DPO: A three-dimensional part orientation system. In *Robotics Research: The 1st International Symposium*, M. Brady and R. Paul, Eds. MIT Press, Cambridge, Mass. © MIT Press 1984.)

3DPO. A model consists of two parts: an augmented CAD model and a set of feature-classification networks. The augmented CAD model is similar to Baumgart's [1972], describing edges, surfaces, and vertices and their relations with one another. The feature-classification network classifies observable features by type and size, for example, surface elements that have the same normal direction and cylinders that have a common axis. Each feature contains a pointer to each instance in all of the augmented CAD models. Figure 20 illustrates this modeling method.

Bolles uses range data to detect surface discontinuities in an image. Two methods are used: detecting discontinuities occurring in 1-D slices of the range finder and finding zero crossings in the output of a second-difference operator applied to the complete range map.

Bolles' matching scheme is similar to that used for the 2-D local-feature-focus method [Bolles and Cain 1982]. First, the system searches for features that match some model's feature (e.g., a cylinder with a given radius). This is accomplished by grouping edges that lie in the same plane,

partitioning each such set of points into line segments and arcs of circles, and associating properties with each line or arc on the basis of relations between the surfaces that meet to form the given segment. Second, objects are hypothesized by determining whether a pair of observed segments are consistent with a given model's features.

Silberberg et al. [1984] model an object using a *multiview* representation to define a Hough space of possible transformations of a set of 3-D line segments (edges), which are observable surface markings on the given object in a given viewpoint. They use a generalized Hough transform to match a set of observed line segments with model lines for each viewpoint. A 3-D Hough space is used to represent a viewpoint (two dimensions for position on the view-sphere, one dimension for orientation at a viewpoint). For each viewpoint and pair of line segments, one from a model and one from the image, the model line is projected onto the image plane, incrementing the corresponding bin in Hough space if the pair of lines match. This procedure is first used with a coarsely quantized Hough space to select a few, approximate, candidate viewpoint regions. Next, each of these viewpoint regions is successively refined to provide a finer resolution estimate of the exact viewing position.

Chakravarty and Freeman [1982] define a *multiview* model using characteristic views for recognizing curved and polyhedral objects. For a given object, they define a finite set of equivalence classes called *characteristic view partitions*, which define a set of *vantage-point domains* on the sphere of possible viewpoints. Each topologically distinct patch is described by a list of the visible lines and junctions in the given object. In order to reduce the number of patches in the partition of the view-sphere, they assume objects will occur in a fixed number of stable positions. The set of possible camera positions is also limited. With these restrictions, two viewpoints are part of the same patch if they contain the same image junctions and lines with the same connectivity relationships, although the lengths of the lines may differ. A linear

transformation describes features within a patch. An object is now modeled as a list of patch descriptors, where each list specifies the number of visible junctions of each of the five possible distinct types for this class of objects [Chakravarty 1979].

Features are combined into a list containing the number of occurrences of each of eight generalized junction types possible for planar and curved-surface objects. Lists are ordered by decreasing significance for recognition and organized into a hierarchical decision tree.

A multistage matching procedure is used for a given observed set of lines and junctions. First, all viewing patches that have similar boundaries to the given observed image boundary are selected; second, patches that do not contain matching junction types are removed; finally, a projection is computed on the basis of the correlated junctions, and this transformation is verified with the original image data.

Faugeras and his colleagues [Faugeras and Hebert 1983; Faugeras et al. 1983, 1984] have developed a system using a surface model computed from a range map. Each object is approximated by a set of planar faces, resulting in a relational graph model in which nodes correspond to faces and arcs connect adjacent faces. Matching is performed using a best-first search for a consistent pairing between observed faces and model faces.

Bhanu [1982] uses a relaxation-labeling technique for identifying which model face is associated with each observed image face. Range data are first merged into planar faces [Henderson and Bhanu 1982]. A two-stage relaxation procedure is then used. In the first stage compatibilities between pairs of adjacent faces are used; in the second stage compatibilities between a face and two of its neighboring faces are used. The compatibility of a face in an unknown view with a face in a model is computed by finding transformations (scale, rotation, translation), applying them, and computing feature value mismatches. The initial probabilities for a face are computed as a function of global features of the face, including area, perimeter, number of vertices, and radius.

Grimson and Lozano-Perez [1984] define a "hypothesize-and-test" search procedure for matching a set of observed surface points (specified by their 3-D position and surface orientation) with a set of polyhedral models (specified by their planar faces). All feasible interpretations of the observed point data are constructed by determining consistent pairings of points to model faces (a point may be mapped into any location on the associated face). Interpretations that are locally inconsistent are rejected. That is, they exploit local constraints on nearby points involving properties such as distance, angle, and direction to rapidly reduce the candidate pairings between a given point and the model faces. For each feasible interpretation of the point data, a final consistency check is made to verify the match.

6. RELATED SURVEYS

There have been published recently a number of survey papers and tutorials that provide selected information on computer vision for industrial automation and robotics. Most of these papers have been organized as summaries of the latest robot vision systems and techniques. In contrast, this paper has attempted to present a more complete listing of results and uses a common descriptive format to clarify similarities and differences in the approaches.

Rosen [1979] examined the desired functions and industrial requirements for machine vision that are applicable to sensor-controlled manipulation. Industrial implementations, as well as selected research problems, are described. Examples are grouped into bin picking, the manipulation of isolated parts on conveyors, the manipulation in manufacturing and assembly, and visual inspection. He also comments that present machine vision techniques are sufficiently advanced to be used in factories in a cost-effective way.

Myers [1980] presents a survey of existing systems, including operational systems in manufacturing and feasibility demonstrations. He describes work done at General Motors Research Laboratories (one of the first to apply computer vision technol-

ogy to a production line), as well as other inspection systems. Yachida and Tsuji [1980] survey industrial machine vision activities in Japan and present a number of successful vision systems that are now operational in Japanese manufacturing. Chin [1982] presents a bibliography on industrial vision for discrete parts.

Kruger and Thompson [1981] present a summary and survey of techniques and applications relevant to the field. They look at generic examples in the areas of inspection, part recognition, and discrete component assembly, and discuss sample systems that exemplify the current state of the art. The authors also make economic projections and give recommendations to guide future investigations. The survey concludes with some comments on the fact that the efficacy of the techniques in any application of machine vision depends on both technical factors and economic considerations.

Foith et al. [1981] discuss selected methods in image processing and analysis related to industrial applications and point out why practical systems perform binary image processing. A brief survey and some specific approaches used in several state-of-the-art systems are presented.

Bolles [1981] reviews some possible applications of image-understanding research to industrial automation, and compares the characteristics of current image-understanding systems with that of industrial automation systems. He points out a few ways in which current image-understanding techniques may be used in the future to enhance the capabilities of industrial systems.

Binford [1982] presents a survey of several general-purpose, model-based image-analysis systems and points out many of the weaknesses of current systems. Besl and Jain [1985] survey general methods for three-dimensional object recognition. Brady [1982a] presents a general survey of image-understanding research.

Kinnucan [1983] briefly looks at the development of machine vision in the United States in the past twenty years and surveys the current activities of several major research laboratories and industries. He also

examines current market activities of existing commercial machine vision systems.

On automated visual inspection, Jarvis [1980] uses three practical examples to illustrate the nature of the techniques and problems. Chin and Harlow [1982] present an extensive survey and discuss the inspection of printed circuit boards, photomasks, and IC chips. Porter and Mundy [1980] provide a comprehensive list of the types of visual-inspection techniques currently in use.

Other related surveys and overviews include Agin [1980], Aleksander et al. [1983], Casler [1983], Fu [1983], Kelly [1983], Kelly et al. [1983], Pot et al. [1983], Pugh [1983], Rossol [1983], Trombly [1982], Tropf et al. [1982], and West [1982].

7. SUMMARY

An extensive review of robot vision techniques for industrial parts recognition has been presented. The major motivation for using industrial machine vision is to increase flexibility and reduce cost of these tasks. Up to the present, primarily very simple techniques based on 2-D global scalar features have been applied in real-time manufacturing processes. More sophisticated techniques will have to be developed in order to deal with less structured industrial environments and permit more task versatility. These techniques will incorporate higher level modeling (e.g., highly organized graph models containing 2½-D and 3-D descriptions), more powerful feature-extraction methods (e.g., global structural features of object boundaries, surfaces, and volumes), and more robust matching procedures for efficiently comparing large sets of complex models with observed image features.

Recent trends indicate that multiple, disparate sensor types, including vision, range, and tactile sensors, will significantly improve the quality of the features that can be determined about a scene. The use of 3-D models is essential for reliably recognizing parts in the presence of significant uncertainty about their identities and positions in a robot's workspace. The choice of a 3-D representation that can be

efficiently used with a matching procedure is still an important research question, however. As more experience is gained, significant speed improvements can be expected with special-purpose hardware for performing this costly search task.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grants ECS-8352356 and ECS-8301521, and in part by the General Motors Foundation, Inc., Dearborn, Michigan.

REFERENCES

- AGIN, G. J. 1980. Computer vision systems for industrial inspection and assembly. *Computer* 13, 5 (May), 11-20.
- AGIN, G. J., AND BINFORD, T. O. 1976. Computer description of curved objects. *IEEE Trans. Comput.* 25, 4 (Apr.), 439-449.
- AGIN, G. J., AND DUDA, R. O. 1975. SRI vision research for advanced automation. In *Proceedings of the 2nd U.S.-Japan Computer Conference* (Tokyo, Japan, Aug.), pp. 113-117.
- AGIN, G. J., AND HIGHNAM, P. T. 1982. A movable light-stripe sensor for obtaining three-dimensional coordinate measurements. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Robotics and Industrial Inspection* (San Diego, Calif., Aug.), vol. 360. SPIE, Bellingham, Wash.
- ALEKSANDER, I., STONHAM, T. J., AND WILKIE, B. A. 1983. Computer vision systems for industry: Comparisons. In *Artificial Vision for Robots*, I. Aleksander, Ed. Chapman and Hall, New York, pp. 179-196.
- ALTSCHULER, M. D., POSDAMER, J. L., FRIEDER, G., ALTSCHULER, B. R., AND TABOADA, J. 1981. The numerical stereo camera. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on 3-D Machine Perception* (Washington, D.C., Apr.), vol. 283. SPIE, Bellingham, Wash., pp. 15-24.
- AYACHE, N. J. 1983. A model-based vision system to identify and locate partially visible industrial parts. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, D.C., June). IEEE, New York, pp. 492-494.
- BADLER, N., AND BAJCSY, R. 1978. Three-dimensional representations for computer graphics and computer vision. *ACM Comput. Gr.* 12, 3 (Aug.), 153-160.
- BAIRD, M. L. 1978. Sight I: A computer vision system for automated IC chip manufacture. *IEEE Trans. Syst. Man Cybern.* 8, 2 (Feb.), 133-139.

- BAJCSY, R. 1973. Computer identification of visual surface. *Comput. Gr. Image Process.* 2, 2 (Oct.), 118-130.
- BAJCSY, R., AND LIEBERMAN, L. 1976. Texture gradient as a depth cue. *Comput. Gr. Image Process.* 5, 1 (Mar.), 52-67.
- BALLARD, D. H. 1981a. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recogn.* 13, 2, 111-122.
- BALLARD, D. H. 1981b. Parameter networks: Towards a theory of low level vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence* (Vancouver, Canada, Aug.). Kaufmann, Los Altos, Calif., pp. 1068-1078.
- BALLARD, D. H., AND BROWN, C. M. 1982. *Computer Vision*. Prentice-Hall, Englewood Cliffs, N.J.
- BARROW, H. G., AND TENENBAUM, J. M. 1978. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. Academic Press, Orlando, Fla., pp. 3-26.
- BARROW, H. G., AND TENENBAUM, J. M. 1981. Computational vision. *Proc. IEEE* 69, 5 (May), 572-595.
- BAUMGART, B. G. 1972. Winged edge polyhedron representation. Tech. Rep. AIM-179, Computer Science Dept., Stanford Univ., Stanford, Calif.
- BESL, P. J., AND JAIN, R. C. 1985. Three-dimensional object recognition. *ACM Comput. Surv.* 17, 1 (Mar.), 75-145.
- BHAND, B. 1982. Surface representation and shape matching of 3-D objects. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing* (Las Vegas, Nev., June). IEEE, New York, pp. 349-354.
- BHANU, B. 1983. Recognition of occluded objects. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence* (Karlsruhe, West Germany, Aug.). Kaufmann, Los Altos, Calif., pp. 1136-1138.
- BHANU, B., AND FAUGERAS, O. D. 1984. Shape matching of two-dimensional objects. *IEEE Trans. Pattern Anal. Mach. Intell.* 6, 2 (Mar.), 137-156.
- BINFORD, T. O. 1971. Visual perception by computer. In *the IEEE Systems Science and Cybernetics Conference* (Miami, Fla., Dec.). IEEE, New York.
- BINFORD, T. O. 1982. Survey of model-based image analysis systems. *Int. J. Robotics Res.* 1, 1 (Spring), 18-63.
- BIRK, J. R., KELLEY, R. B., CHEN, N.-Y., AND WILSON, L. 1979. Image feature extraction using diameter-limited gradient direction histograms. *IEEE Trans. Pattern Anal. Mach. Intell.* 1, 2 (Apr.), 228-235.
- BIRK, J. R., KELLEY, R. B., AND MARTINS, H. 1981. An orienting robot for feeding workpieces stored in bins. *IEEE Trans. Syst. Man Cybern.* 11, 2 (Feb.), 151-160.
- BOLLES, R. C. 1979a. Symmetry analysis of two-dimensional patterns for computer vision. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence* (Tokyo, Japan, Aug.). Kaufmann, Los Altos, Calif., pp. 70-72.
- BOLLES, R. C. 1979b. Robust feature matching through maximal cliques. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Imaging Applications for Automated Industrial Inspection and Assembly* (Washington, D.C., Apr.), vol. 182. SPIE, Bellingham, Wash., pp. 140-149.
- BOLLES, R. C. 1981. Overview of applications of image understanding to industrial automation. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Techniques and Applications of Image Understanding* (Washington, D.C., Apr.), vol. 281. SPIE, Bellingham, Wash., pp. 134-140.
- BOLLES, R. C., AND CAIN, R. A. 1982. Recognizing and locating partially visible objects: The local-feature-focus method. *Int. J. Robotics Res.* 1, 3, 57-82.
- BOLLES, R. C., AND FISCHLER, M. A. 1981. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence* (Vancouver, Canada, Aug.). Kaufmann, Los Altos, Calif., pp. 637-643.
- BOLLES, R. C., HORAUD, P., AND HANNAH, M. J. 1984. 3DPO: A three-dimensional part orientation system. In *Robotics Research: The 1st International Symposium*, M. Brady and R. Paul, Eds. MIT Press, Cambridge, Mass., pp. 413-424.
- BRADY, M. 1982a. Computational approaches to image understanding. *ACM Comput. Surv.* 14, 1 (Mar.), 3-71.
- BRADY, M. 1982b. Parts description and acquisition using vision. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Robot Vision* (Arlington, Va., May), vol. 336. SPIE, Bellingham, Wash., pp. 20-28.
- BROOKS, R. A. 1979. Goal-directed edge linking and ribbon finding. In *Proceedings of the Image Understanding Workshop* (Menlo Park, Calif., Apr.). Science Applications, Arlington, Va., pp. 72-76.
- BROOKS, R. A. 1983a. Symbolic reasoning among 3-D models and 2-D images. *Artif. Intell.* 17, 1 (Aug.), 285-348.
- BROOKS, R. A. 1983b. Model-based three-dimensional interpretations of two-dimensional images. *IEEE Trans. Pattern Anal. Mach. Intell.* 5, 2 (Mar.), 140-150.
- BROOKS, R. A., AND BINFORD, T. O. 1981. Geometric modeling in vision for manufacturing. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Robot Vision* (Washington, D.C., Apr.), vol. 281. SPIE, Bellingham, Wash., pp. 141-159.
- BROU, P. 1984. Using the Gaussian image to find the orientation of objects. *Int. J. Robotics Res.* 3, 4 (Winter), 89-125.
- BRUNE, W., AND BITTER, K. H. 1983. S.A.M. Optoelectronic picture sensor in a flexible manufac-

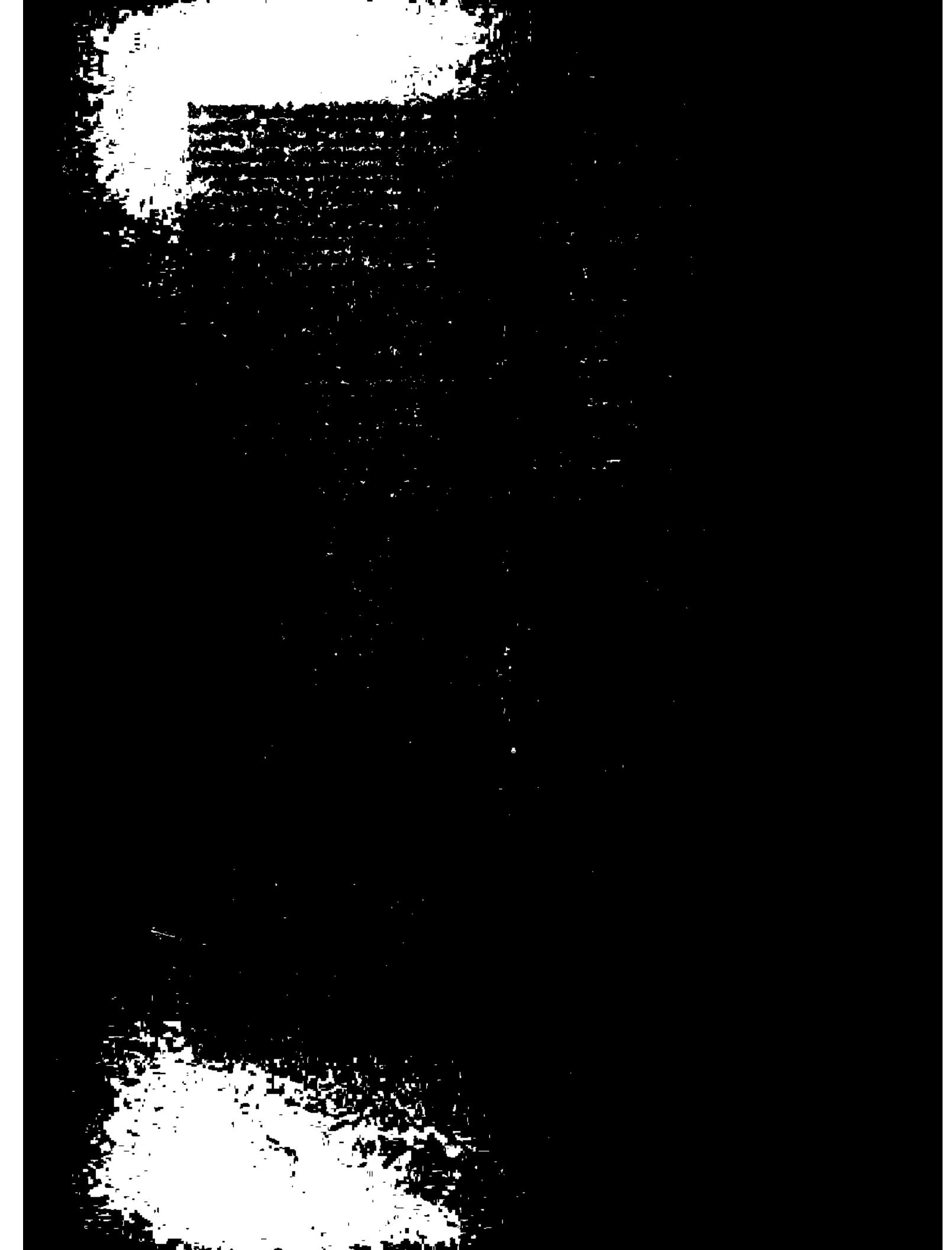
- turing system. In *Robot Vision*, A. Pugh, Ed. Springer-Verlag, New York, pp. 325-337.
- CASLER, R. J. 1983. Vision-guided robot part acquisition for assembly packaging applications. Tech. Paper MS83-219, Society of Manufacturing Engineers, Dearborn, Mich.
- CHAKRAVARTY, I. 1979. A generalized line and junction labeling scheme with applications to scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* 1, 2 (Apr.), 202-205.
- CHAKRAVARTY, I., AND FREEMAN, H. 1982. Characteristic views as a basis for three-dimensional object recognition. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Robot Vision* (Arlington, Va., May), vol. 336. SPIE, Bellingham, Wash., pp. 37-45.
- CHEN, M. J., AND MILGRAM, D. L. 1982. A development system for machine vision. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing* (Las Vegas, Nev., June). IEEE, New York, pp. 512-517.
- CHEN, N.-Y., BIRK, J. R., AND KELLEY, R. B. 1980. Estimating workpiece pose using the feature points method. *IEEE Trans. Auto. Control* 25, 6 (Dec.), 1027-1041.
- CHENG, J. K., AND HUANG, T. S. 1981. Image recognition by matching relational structure. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing* (Dallas, Tex., Aug.). IEEE, New York, pp. 542-547.
- CHENG, J. K., AND HUANG, T. S. 1982. Recognition of curvilinear objects by matching relational structure. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing* (Las Vegas, Nev., June). IEEE, New York, pp. 343-348.
- CHIN, R. T. 1982. Machine vision for discrete part handling in industry: A survey. In *Conference Record of the Workshop on Industrial Applications of Machine Vision* (Research Triangle Park, N.C., May). IEEE, New York, pp. 26-32.
- CHIN, R. T., AND HARLOW, C. A. 1982. Automated visual inspection: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 4, 6 (Nov.), 557-573.
- CHOW, C. K., AND KANEKO, T. 1972. Automatic boundary detection of the left ventricle from cineangiograms. *Comput. Biomed. Res.* 5, 4 (Aug.), 388-410.
- DESSIMOZ, J.-D. 1978a. Visual identification and location in a multiobject environment by contour tracking and curvature description. In *Proceedings of the 8th International Symposium on Industrial Robots* (Stuttgart, West Germany, May), pp. 746-776.
- DESSIMOZ, J.-D. 1978b. Recognition and handling of overlapping industrial parts. In *Proceedings of the International Symposium on Computer Vision and Sensor-Based Robots* (Warren, Mich., Sept.). General Motors Research Symposium, Warren, Mich.
- DUDA, R. O., NITZAN, D., AND BARRET, P. 1979. Use of range and reflectance data to find planar surface regions. *IEEE Trans. Pattern Anal. Mach. Intell.* 1, 3 (July), 259-271.
- FAUGERAS, O. D., AND HEBERT, M. 1983. A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence* (Karlsruhe, West Germany, Aug.). Kaufmann, Los Altos, Calif., pp. 996-1002.
- FAUGERAS, O. D., GERMAIN, F., KRYZE, G., BOISSONNAT, J., HEBERT, M., PONCE, J., PAUCHON, E., AND AYACHE, N. 1983. Towards a flexible vision system. In *Robot Vision*, A. Pugh, Ed. Springer-Verlag, New York, pp. 129-142.
- FAUGERAS, O. D., HEBERT, M., PAUCHON, E., AND PONCE, J. 1984. Object representation, identification, and positioning from range data. In *Robotics Research: The First International Symposium*, M. Brady and R. Paul, Eds. MIT Press, Cambridge, Mass., pp. 425-446.
- FOTH, J. P., EISENBARTH, C., ENDERLS, E., GEISSELMANN, H., RINGSHAUSER, H., AND ZIMMERMANN, G. 1981. Real-time processing of binary images for industrial applications. In *Digital Image Processing Systems*, L. Bolc and Z. Kulpa, Eds. Springer-Verlag, Berlin, pp. 61-168.
- FU, K. S. 1983. Robot vision for machine part recognition. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Robotics and Robot Sensing Systems* (San Diego, Calif., Aug.), vol. 442. SPIE, Bellingham, Wash.
- FUCHS, H., KEDEM, Z. M., AND NAYLOR, B. F. 1980. On visible surface generation by a priori tree structures. In *Proceedings of SIGGRAPH '80* (Seattle, Wash., July). ACM, New York, pp. 124-133.
- GLEASON, G. J., AND AGIN, G. J. 1979. A modular system for sensor-controlled manipulation and inspection. In *Proceedings of the 9th International Symposium on Industrial Robots* (Washington, D.C., Mar.). Society of Manufacturing Engineers, Dearborn, Mich., pp. 57-70.
- GOAD, C. 1983. Special-purpose automatic programming for 3D model-based vision. In *Proceedings of the Image Understanding Workshop* (Arlington, Va., June). Science Applications, Arlington, Va., pp. 94-104.
- GRIMSON, W. E. L., AND LOZANO-PEREZ, T. 1984. Model-based recognition and localization from sparse range or tactile data. *Int. J. Robotics Res.* 3, 3 (Fall), 3-35.
- HATTICH, W. 1982. Recognition of overlapping workpieces by model directed construction of object contours. *Digital Syst. Ind. Autom.* 1, 223-239.
- HENDERSON, T. C. 1982. Efficient segmentation method for range data. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Robot Vision* (Arlington, Va., May), vol. 336. SPIE, Bellingham, Wash., pp. 46-47.

- HENDERSON, T. C., AND BHANU, B. 1982. Three-point seed method for the extraction of planar faces from range data. In *Conference Record of the Workshop on Industrial Applications of Machine Vision* (Research Triangle Park, N.C., May). IEEE, New York, pp. 181-186.
- HOLLAND, S. W., ROSSOL, L., AND WARD, M. R. 1979. CONSIGHT-I: A vision-controlled robot system for transferring parts from belt conveyors. In *Computer Vision and Sensor-Based Robots*, G. G. Dodd and L. Rossol, Eds. Plenum, New York, pp. 81-97.
- HORN, B. K. P. 1975a. A problem in computer vision: Orienting silicon-integrated circuit chips for lead bonding. *Comput. Gr. Image Process.* 4, 3 (Sept.), 294-303.
- HORN, B. K. P. 1975b. Obtaining shape from shading information. In *The Psychology of Computer Vision*, P. H. Winston, Ed. McGraw-Hill, New York, pp. 115-155.
- HORN, B. K. P. 1979. SEQUINS and QUILLS—Representations for surface topography. Artificial Intelligence Laboratory Memo 536, MIT, Cambridge, Mass., May.
- HORN, B. K. P. 1984. Extended Gaussian images. *Proc. IEEE* 72, 12 (Dec.), 1671-1686.
- HSIEH, Y. Y., AND FU, K. S. 1979. A method for automatic IC chip alignment and wire bonding. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing* (Chicago, Ill., Aug.). IEEE, New York, pp. 101-108.
- HUECKEL, M. F. 1971. An operator which locates edges in digitized pictures. *J. ACM* 18, 1 (Jan.), 113-125.
- IGARASHI, K., NARUSE, M., MIYAZAKI, S., AND YAMADA, T. 1979. Fully automated integrated circuit wire bonding system. In *Proceedings of the 9th International Symposium on Industrial Robots* (Washington, D.C., Mar.). Society of Manufacturing Engineers, Dearborn, Mich., pp. 87-97.
- IKEUCHI, K. 1981a. Determining surface orientations of specular surfaces by using the photometric stereo method. *IEEE Trans. Pattern Anal. Mach. Intell.* 3, 6 (Nov.), 661-669.
- IKEUCHI, K. 1981b. Recognition of 3-D objects using the extended Gaussian image. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence* (Vancouver, Canada, Aug.). Kaufmann, Los Altos, Calif., pp. 595-600.
- IKEUCHI, K. 1983. Determining the attitude of an object from a needle map using the extended Gaussian image. Artificial Intelligence Laboratory Memo 714, M.I.T., Cambridge, Mass., Apr.
- IKEUCHI, K., AND SHIRAI, Y. 1982. A model-based vision system for recognition of machine parts. In *Proceedings of the National Conference on Artificial Intelligence* (Pittsburgh, Pa., Aug.). Kaufmann, Los Altos, Calif., pp. 18-21.
- IKEUCHI, K., HORN, B. K. P., NAGATA, S., CALLAHAN, T., AND FEINGOLD, O. 1984. Picking up an object from a pile of objects. In *Robotics Research: The First International Symposium*, M. Brady and R. Paul, Eds. MIT Press, Cambridge, Mass., pp. 139-162.
- JAKUBOWSKI, R. 1982. Syntactic characterization of machine parts shapes. *Cybern. Syst.* 13, 1 (Jan.-Mar.), 1-24.
- JAKUBOWSKI, R., AND KASPRZAK, A. 1977. A syntactic description and recognition of rotary machine elements. *IEEE Trans. Comput.* 26, 10 (Oct.), 1039-1042.
- JARVIS, J. F. 1980. Visual inspection automation. *Computer* 13, 5 (May), 32-39.
- JARVIS, R. A. 1983a. A perspective on range finding techniques. *IEEE Trans. Pattern Anal. Mach. Intell.* 5, 2 (Mar.), 122-139.
- JARVIS, R. A. 1983b. A laser time-of-flight range scanner for robotic vision. *IEEE Trans. Pattern Anal. Mach. Intell.* 5, 5 (Sept.), 505-512.
- KANADE, T., AND ASADA, H. 1981. Noncontact visual three-dimensional ranging devices. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on 3-D Machine Perception* (Washington, D.C., Apr.), vol. 283. SPIE, Bellingham, Wash., pp. 48-53.
- KASHIOKA, S., EJIRI, M., AND SAKAMOTO, Y. 1976. A transistor wire-bonding system utilizing multiple local pattern matching techniques. *IEEE Trans. Syst. Man Cybern.* 6, 8 (Aug.), 562-569.
- KASHIOKA, S., TAKEDA, S., SHIMA, Y., UNO, T., AND HAMADA, T. 1977. An approach to the integrated intelligent robot with multiple sensory feedback: Visual recognition techniques. In *Proceedings of the 7th International Symposium on Industrial Robots* (Tokyo, Japan, Oct.). Japan Industrial Robot Association, pp. 531-538.
- KELLEY, R. B. 1983. Binary and gray-scale robot vision. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Robotics and Robot Sensing Systems* (San Diego, Calif., Aug.), vol. 442. SPIE, Bellingham, Wash.
- KELLEY, R. B., BIRK, J. R., MARTINS, H. A. S., AND TELLA, R. 1982. A robot system which acquires cylindrical workpieces from bins. *IEEE Trans. Syst. Man Cybern.* 12, 2 (Mar./Apr.), 204-213.
- KELLEY, R. B., MARTINS, H. A. S., BIRK, J. R., AND DESSIMON, J.-D. 1983. Three vision algorithms for acquiring workpieces from bins. *Proc. IEEE* 71, 7 (July), 803-820.
- KENDER, J. R. 1980. Shape from texture. Ph.D. dissertation, Computer Science Dept., Carnegie-Mellon Univ., Pittsburgh, Pa.
- KINNUCAN, P. 1983. Machines that see. *High Technol.* 3, 4 (Apr.), 30-36.
- KITCHIN, P. W., AND PUGH, A. 1983. Processing of binary images. In *Robot Vision*, A. Pugh, Ed. Springer-Verlag, New York, pp. 21-42.
- KOENDERINK, J. J., AND VANDOORN, A. J. 1976a. The singularities of the visual mapping. *Biol. Cybern.* 24, 51-59.
- KOENDERINK, J. J., AND VANDOORN, A. J. 1976b. Visual perception of rigidity of solid shape. *J. Math. Biol.* 3, 79-85.

- KOENDERINK, J. J., AND VANDOORN, A. J. 1979. The internal representation of solid shape with respect to vision. *Biol. Cybern.* 32, 211-216.
- KRÜGER, R. P., AND THOMPSON, W. B. 1981. A technical and economic assessment of computer vision for industrial inspection and robotic assembly. *Proc. IEEE* 69, 12 (Dec.), 1524-1538.
- LIEBERMAN, L. 1979. Model-driven vision for industrial automation. In *Advances in Digital Image Processing*, P. Stucki, Ed. Plenum, New York, pp. 235-246.
- MARIMONT, D. H. 1982. Segmentation in Acronym. In *Proceedings of the Image Understanding Workshop* (Palo Alto, Calif., Sept.). Science Applications, Arlington, Va., pp. 223-229.
- MARR, D. 1978. Representing visual information. In *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. Academic Press, Orlando, Fla., pp. 61-80.
- MARR, D. 1982. *Vision*. Freeman, San Francisco.
- MESE, M., YAMAZAKI, I., AND HAMADA, T. 1977. An automatic position recognition technique for LSI assembly. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (Cambridge, Mass., Aug.). Kaufmann, Los Altos, Calif., pp. 685-693.
- MILGRAM, D. L., AND BJORKLUND, C. M. 1980. Range image processing: Planar surface extraction. In *Proceedings of the 5th International Conference on Pattern Recognition* (Miami Beach, Fla., Dec.). IEEE, New York, pp. 912-919.
- MYERS, W. 1980. Industry begins to use visual pattern recognition. *Computer* 13, 5 (May), 21-31.
- NEVATIA, R., AND BINFORD, T. O. 1977. Description and recognition of curved objects. *Artif. Intell.* 8, 1 (Jan.), 77-98.
- OSHIMA, M., AND SHIRAI, Y. 1979. A scene description method using three-dimensional information. *Pattern Recogn.* 11, 1, 9-17.
- OSHIMA, M., AND SHIRAI, Y. 1983. Object recognition using three-dimensional information. *IEEE Trans. Pattern Anal. Mach. Intell.* 5, 4 (July), 353-361.
- PAGE, C. J., AND PUGH, A. 1981. Visually interactive gripping of engineering parts from random orientation. *Digital Syst. Ind. Autom.* 1, 1, 11-44.
- PERKINS, W. A. 1978. A model-based vision system for industrial parts. *IEEE Trans. Comput.* 27, 2 (Feb.), 126-143.
- PERKINS, W. A. 1980. Area segmentation of images using edge points. *IEEE Trans. Pattern Anal. Mach. Intell.* 2, 1 (Jan.), 8-15.
- PERSOON, E., AND FU, K. S. 1977. Shape discrimination using Fourier descriptors. *IEEE Trans. Syst. Man Cybern.* 7, 3 (Mar.), 170-179.
- PIPITONE, F. J., AND MARSHALL, T. G. 1983. A wide-field scanning triangulation rangefinder for machine vision. *Int. J. Robotics Res.* 2, 1 (Spring), 39-49.
- POJE, J. F., AND DELP, E. J. 1982. A review of techniques for obtaining depth information with applications to machine vision. Tech. Rep. RSD-TR-2-82, Center for Robotics and Integrated Manufacturing, Univ. of Michigan, Ann Arbor.
- POPLESTONE, R. J., BROWN, C. M., AMBLER, A. P., AND CRAWFORD, G. F. 1975. Forming models of plane-and-cylinder faceted bodies from light stripes. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence* (Tbilisi, USSR, Sept.). Kaufmann, Los Altos, Calif., pp. 664-668.
- PORTER, G. B., AND MUNDY, J. L. 1980. Visual inspection system design. *Computer* 13, 5 (May), 40-49.
- POT, J., COIFFET, P., AND RIVES, P. 1983. Comparison of five methods for the recognition of industrial parts. In *Developments in Robotics*, B. Rooks, Ed. Springer-Verlag, New York.
- POTMESIL, M. 1983. Generating models of solid objects by matching 3D surface segments. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence* (Karlsruhe, West Germany, Aug.). Kaufmann, Los Altos, Calif., pp. 1089-1093.
- PUGH, A., ED. 1983. *Robot Vision*. Springer-Verlag, New York.
- RAY, R., BIRK, J., AND KELLEY, R. B. 1983. Error analysis of surface normals determined by radiometry. *IEEE Trans. Pattern Anal. Mach. Intell.* 5, 6 (Nov.), 631-645.
- REQUICHA, A. A. G. 1980. Representations for rigid solids: Theory, methods, and systems. *ACM Comput. Surv.* 12, 4 (Dec.), 437-464.
- ROSEN, C. A. 1979. Machine vision and robotics: Industrial requirements. In *Computer Vision and Sensor-Based Robots*, G. G. Dodd and L. Rossol, Eds. Plenum, New York, pp. 3-20.
- ROSEN, C. A., AND GLEASON, G. J. 1981. Evaluating vision system performance. *Robotics Today* (Fall).
- ROSENFIELD, A., AND DAVIS, L. S. 1979. Image segmentation and image models. *Proc. IEEE* 67, 5 (May), 764-772.
- ROSSOL, L. 1983. Computer vision in industry. In *Robot Vision*, A. Pugh, Ed. Springer-Verlag, New York, pp. 11-18.
- SCHACHTER, B. J. 1983. A matching algorithm for robot vision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, D.C., June). IEEE, New York, pp. 490-491.
- SEGEN, J. 1983. Locating randomly oriented objects from partial views. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Robot Vision and Sensory Controls*, (Cambridge, Mass., Nov.), vol. 449. SPIE, Bellingham, Wash.
- SHIRAI, Y. 1972. Recognition of polyhedrons with a range finder. *Pattern Recogn.* 4, 3 (Oct.), 243-250.
- SHIRAI, Y. 1975. Edge finding, segmentation of edges and recognition of complex objects. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence* (Tbilisi, USSR, Sept.). Kaufmann, Los Altos, Calif., pp. 674-681.

- SHIRAI, Y. 1978. Recognition of real-world objects using edge cues. In *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. Academic Press, New York, pp. 353-362.
- SHNEIER, M. 1979. A compact relational structure representation. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence* (Tokyo, Japan, Aug.). Kaufmann, Los Altos, Calif., pp. 818-826.
- SHNEIER, M. 1981. Models and strategies for matching in industrial vision. Tech. Rep. TR-1073, Computer Science Dept., Univ. of Maryland, College Park, July.
- SILBERBERG, T. M., DAVIS, L. S., AND HARWOOD, D. 1984. An iterative Hough procedure for three-dimensional object recognition. *Pattern Recognit.* 17, 6, 621-629.
- SILVER, W. M. 1980. Determining shape and reflectance using multiple images. M.Sc. thesis, M.I.T., Cambridge, Mass.
- SMITH, D. A. 1979. Using enhanced spherical images for object representation. Artificial Intelligence Laboratory Memo 530, M.I.T., Cambridge, Mass., May.
- STEVENS, K. A. 1981. The information content of texture gradients. *Biol. Cybern.* 42, 95-105.
- STOCKMAN, G. C. 1980. Recognition of parts and their orientation for automatic machining, handling and inspection. Rep. NSF-SIBR-Phase I, NTIS Order PB 80-178817.
- STOCKMAN, G. C., KOPSTEIN, K., AND BENNETT, S. 1982. Matching images to models for registration and object detection via clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* 4, 3 (May), 229-241.
- SUGIHARA, K. 1979. Range-data analysis guided by a junction dictionary. *Artif. Intell.* 12, 1 (May), 41-69.
- TAKEYASU, K., KASAI, M., SHIMOMURA, R., GOTO, T., AND MATSUMOTO, Y. 1977. An approach to the integrated intelligent robot with multiple sensory feedback. In *Proceedings of the 7th International Symposium on Industrial Robots* (Tokyo, Japan, Oct.), pp. 523-530.
- TENENBAUM, J. M., BARROW, H. G., AND BOLLES, R. C. 1979. Prospects for industrial vision. In *Computer Vision and Sensor-Based Robots*, G. G. Dodd and L. Rossol, Eds. Plenum, New York, pp. 239-256.
- TROMBLY, J. E. 1982. Recent applications of computer aided vision in inspection and part sorting. In *Proceedings of the Robot VI Conference* (Detroit, Mich., Mar.). Society of Manufacturing Engineers, Dearborn, Mich.
- TROPP, H. 1980. Analysis-by-synthesis search for semantic segmentation applied to workpiece recognition. In *Proceedings of the 5th International Conference on Pattern Recognition* (Miami Beach, Fla., Dec.). IEEE, New York, pp. 241-244.
- TROPP, H. 1981. Analysis-by-synthesis search to interpret degraded image data. In *Proceedings of the 1st International Conference on Robot Vision and Sensory Controls* (Stratford-upon-Avon, England, Apr.). IFS, Kempston, England, pp. 25-33.
- TROPP, H., GEISSELMANN, H., AND FOITH, J. P. 1982. Some applications of the fast industrial vision system S.A.M. In *Conference Record of the Workshop on Industrial Applications of Machine Vision* (Research Triangle Park, N.C., May). IEEE, New York, pp. 73-79.
- TURNEY, J. L., MUDGE, T. N., AND VOLZ, R. A. 1985. Recognizing partially occluded parts. *IEEE Trans. Pattern Anal. Mach. Intell.* 7, 4 (July), 410-421.
- UMETANI, Y., AND TAGUCHI, K. 1979. Feature properties to discriminate complex shapes. In *Proceedings of the 9th International Symposium on Industrial Robots* (Washington, D.C., Mar.). Society of Manufacturing Engineers, Dearborn, Mich., pp. 367-378.
- UMETANI, Y., AND TAGUCHI, K. 1982. Discrimination of general shapes by psychological feature properties. *Digital Syst. Ind. Autom.* 1, 2-3, 179-196.
- VAMOS, T. 1977. Industrial objects and machine parts recognition. In *Applications of Syntactic Pattern Recognition*, K. S. Fu, Ed. Springer-Verlag, New York.
- VILLERS, P. 1983. Present industrial use of vision sensors for robot guidance. In *Robot Vision*, A. Pugh, Ed. Springer-Verlag, New York, pp. 157-168.
- WEST, P. C. 1982. Overview of machine vision. Tech. Paper MS82-184, Society of Manufacturing Engineers, Dearborn, Mich.
- WITKIN, A. P. 1981. Recovering surface shape and orientation from texture. *Artif. Intell.* 17, 1 (Aug.), 17-47.
- WOODHAM, R. J. 1978. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Image Understanding Systems and Industrial Applications* (San Diego, Calif., Aug.), vol. 155. SPIE, Bellingham, Wash., pp. 136-143.
- YACHIDA, M., AND TSUJI, S. 1977. A versatile machine vision system for complex industrial parts. *IEEE Trans. Comput.* 26, 9 (Sept.), 882-894.
- YACHIDA, M., AND TSUJI, S. 1980. Industrial computer vision in Japan. *Computer* 13, 5 (May), 50-63.
- ZAHN, C. T., AND ROSKIEW, R. Z. 1972. Fourier descriptors for plane closed curves. *IEEE Trans. Comput.* 21, 3 (Mar.), 269-281.
- ZUECH, N., AND RAY, R. 1983. Vision guided robotic arm control for part acquisition. In *Proceedings of the Control Engineering Conference* (West Lafayette, Ind.). Purdue Univ., West Lafayette, Ind.

Received January 1984; final revision accepted February 1986



Chapter 6: Dynamic Vision

Early computer vision systems were concerned primarily with static scenes. However, the world is dynamic. Designing computer-vision systems capable of analyzing dynamic scenes has received increasing attention in the last few years. For a computer-vision system engaged in the performance of nontrivial real-world operations and tasks, the ability to cope with moving and changing objects and viewpoints is vital. Most biological vision systems have evolved to cope with the changing world.

The input to a dynamic-scene analysis system is a sequence of image frames taken from a changing world. The camera used to acquire the image sequence may also be in motion. Each frame represents an image of the scene at a particular instant in time. The changes in a scene may be due to camera motion, object motion, illumination changes, or changes in object structure, size, or shape. Changes in a scene are usually assumed to be due to camera and/or object motion. Objects are usually assumed to be either rigid or quasi-rigid. Other changes are not allowed. The tasks of a dynamic-scene analysis system are to

- Detect changes;
- Determine the motion characteristics of the observer and the objects;
 - Characterize the motion using high-level abstraction;
 - Recover the structure of the objects; and
- Recognize moving objects.

Future systems will possibly be required to first observe a scene and then describe in natural language the events taking place.

A scene usually contains several objects. An image of the scene at a given time represents a projection of part of the scene; the part of the scene projected depends on the position of the camera. The following cases represent the four possibilities for the dynamic-camera/world setup:

- (1) Stationary camera, stationary objects (*SCSO*);
- (2) Stationary camera, moving objects (*SCMO*);
- (3) Moving camera, stationary objects (*MCSO*); and
- (4) Moving camera, moving objects (*MCMO*).

The first case is simply static-scene analysis. In many applications, processing a single image to obtain the required information may be necessary and/or possible. However, many more applications exist that require information to be extracted from a dynamic environment. Some applications require a vision system to understand a dynamic process, such as cell motion.

Clearly, a sequence of image frames offers much more information to aid in understanding a scene, but significantly increases the amount of data to be processed by the system. Applying static-scene analysis techniques to each frame of a sequence requires an enormous amount of computation, while still suffering from all of the problems of static-scene analysis. Fortunately, research in dynamic-scene analysis has shown that information recovery may be easier in dynamic scenes than in static scenes; in some cases, total computational effort may be significantly less and performance better (e.g., segmenting a scene).

SCMO scenes have received the most attention in dynamic-scene analysis. The objectives of such scene analysis usually are to detect motion, to extract masks of moving objects with the aim of recognizing them, and to compute their motion characteristics. *MCSO* scenes have recently received some attention, but *MCMO* scenes have been virtually ignored. *MCMO* is the most generalized case and possibly presents the most difficult situation in dynamic-scene analysis. Many techniques that have been developed assuming a stationary camera are not applicable if the camera is allowed to move. Likewise, techniques that have been developed for a moving camera have generally assumed a stationary scene and are usually not applicable if the objects are allowed to move. *SCMO* and *MCSO* have found many applications and have been studied by researchers in various contexts under various assumptions.

Dynamic-scene analysis can be considered to be composed of three phases: peripheral, attentive, and cognitive. The peripheral phase is concerned with extraction of approximate information, which is very helpful in the later phases. This information indicates the activity in a scene; it is used to decide the parts of the scene that need careful analysis. The attentive phase concentrates its analysis on the active parts of the scene and extracts information that can be used for object recognition, analysis of object motion, preparation of a history of events taking place in the scene, or other related phenomena. The cognitive phase applies knowledge about objects, motion verbs, and other application-dependent concepts to analyze the scene in terms of the objects present and the events taking place.

The input to a dynamic-scene analysis system is a frame sequence, represented by $F(x,y,t)$, where x and y are the spatial coordinates in the frame representing the scene at time t . The value of the function represents the intensity of the pixel. The image is assumed to be obtained using a camera located at the origin of a three-dimensional coordinate system. The projection used in this observer-centered system may be either perspective or orthographic. The three-dimensional coordinates of a point are (X,Y,Z) and the projection of the point onto the image plane is denoted by (x,y) . The line of sight, or the optical axis, of the camera is assumed to be along the Z -axis. Since the frames are usually taken at regular intervals, we will assume that t represents the t^{th} frame of the sequence, rather than the frame taken at absolute time t .

Change detection

Any perceptible motion in a scene results in some change in the sequence of frames of the scene. If such changes are detected, motion characteristics can be analyzed. A good quantitative estimate of the motion components of an object can be obtained if the motion is restricted to a plane that is parallel to the image plane; for three-dimensional motion, only qualitative estimates are possible. Historically, dynamic-scene analysis is based on the detection of change in a frame sequence. By analyzing frame-to-frame changes, a global analysis of the sequence can be performed. Changes can be detected at different levels: pixel, edge, or region. Changes detected at the pixel level can be aggregated to obtain useful information with which the computational requirements of later phases can be constrained.

Pixel-level change detection. The most obvious method of detecting change between two frames is to directly compare the corresponding pixels of the frames to determine whether or not they are the same. In the simplest form, a binary difference picture $DP_{jk}(x,y)$ between frames $F(x,y,j)$ and $F(x,y,k)$ is obtained by

$$DP_{jk}(x,y) = \begin{cases} 1 & \text{if } |F(x,y,j) - F(x,y,k)| > \tau \\ 0 & \text{otherwise} \end{cases} \quad (6.1)$$

where τ is a threshold.

In the difference picture, pixels that have value 1 may be considered to be the result of object motion, assuming that the frames have been properly registered and that the illumination in the image has remained constant. Because of noise, such a simple test on real scenes usually results in unsatisfactory results. A simple size filter may be used to ignore pixels not forming a connected cluster of a minimum size. Then only those pixels in a difference picture with a value of 1 that belong to a four-connected (or eight-connected) component larger than some threshold size will be attributed to motion. The fact that noise entries are usually isolated and that changes due to the motion of surfaces form connected clusters in difference pictures comprise the motivations behind the use of this filter. The filter is very effective in reducing noise, but it also unfortunately filters some desirable signals, such as those from slow or small moving objects.

To make change detection more robust, regions or groups of pixels at the same location in two frames may be considered and their intensity characteristics may be compared more rigorously. A method that illustrates making change detection more robust is based on comparing the frames using the likelihood ratio.²¹¹ Thus, we can compute

$$\lambda = \frac{\left[\frac{\sigma_1^2 + \sigma_2^2}{2} + \left(\frac{\mu_1 - \mu_2}{2} \right)^2 \right]^2}{\sigma_1^2 \sigma_2^2} \quad (6.2)$$

where μ denotes the mean gray value and σ denotes the square root of the variance from the frames for the sample areas. Then we use

$$DP_{jk}(x,y) = \begin{cases} 1 & \text{if } \lambda > \tau \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

where τ is a threshold.

The likelihood ratio can be applied to areas only and not to single pixels. This limitation presents a minor problem, which can be solved by considering corresponding areas of the frames. The likelihood ratio test, combined with the use of a size filter, work

quite well for noise removal. The likelihood ratio test can be applied to every point in each frame by considering overlapping areas centered on each pixel of the image.

Superpixels effectively raise the size threshold for the detection of slow and small moving objects; thus, the problem of missing detection of such objects is exacerbated. However, this problem can be solved by analyzing change over a sequence of frames, instead of between just two frames, and using an accumulative-difference picture (ADP) to detect the motion of slow and small moving objects. An accumulative-difference picture is formed by comparing every frame of an image sequence to a reference frame, and increasing the entry in the accumulative-difference picture by one whenever the likelihood ratio for the area exceeds the threshold. Thus, an accumulative-difference picture ADP_k is acquired over k frames by

$$\begin{aligned} ADP_0(x,y) &= 0 \\ ADP_k(x,y) &= ADP_{k-1}(x,y) + DP_{1k}(x,y) \end{aligned} \quad (6.4)$$

The first frame of a sequence is usually the reference frame, and the accumulative-difference picture ADP_0 is initialized to zero, as illustrated in Figure 6.1.²¹²

	9	10	11	12	13	14	15	16	9	10	11	12	13	14	15	16	9	10	11	12	13	14	15	16	9	10	11	12	13	14	15	16											
(a)	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000											
(b)	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000												
(c)	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000												
(d)	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000												
(e)	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	A98765438887654321																									

Figure 6.1: Calculation of Accumulative Difference Picture (ADP); (a-e) Frames 1,2,3,4, and 11; (f-l) corresponding ADP's (from Jain)²¹²

The likelihood test just discussed is based on the assumption of uniform second-order statistics over a region. The performance of the likelihood ratio test can be improved significantly by using facets and quadratic surfaces to approximate the intensity values of pixels belonging to superpixels. These higher order approximations allow for better characterization of intensity values and result in more robust change detection.

The most attractive aspect of the difference picture is its simplicity. In its simplest form, the difference picture appears to be noise-prone. Electronic noise of the camera and changes in illumination and registration of the camera can result in many false alarms. A likelihood ratio, in conjunction with a size filter, can eliminate most of the camera noise. Changes in illumination will create problems for any intensity-based approach and can only be handled at a symbolic level. Misregistration of frames results in the assignment of false motion components. If the misregistration is not severe, accumulative-difference pictures can eliminate its effects.

Emphasis should be placed on the fact that measuring dissimilarities at the pixel level can detect only intensity changes. In dynamic-scene analysis, this is the lowest level of analysis. After such intensity changes have been detected, other processes are required to interpret these changes. Experience has shown that the most efficient use of the difference picture is to have peripheral processes direct the attention of interpretation processes to areas of the scene where some activity is taking place. Approximate information about events in a scene can be extracted using some features of difference pictures.

Static segmentation and matching. Segmentation is the task of identifying meaningful components in the context of an application of an image and grouping the pixels belonging to such components. Segmentation need not necessarily be performed in terms of objects; it may also be performed in terms of features — or predicates based on intensity characteristics. If an object or feature appears in two or more images, segmentation may be necessary in order to identify the object in the images. The process of identifying the same object or feature in two or more frames is called the "correspondence process."

Static-scene analysis techniques can be used to segment, or at least to partially segment, each frame of a dynamic sequence. Matching can then be used to determine correspondences and to detect changes in the location of corresponding segments. Cross-correlation and Fourier domain features have been used as a form of matching to detect cloud motion. Several systems have been developed that segment each frame of a sequence to find regions, corners, edges, or other features in the frames. Features in consecutive frames are then matched to detect any displacement. Some restriction on the possible matches for a feature can be achieved by predicting the new location of the feature based on the displacements in previous frames.

In the approaches just described, segmentation presents the major difficulty: Segmentation of an image in a static scene has been a difficult problem. Even after Herculean efforts, most attempts to segment complex real-world scenes give only good segmentations. However, better segmentation is widely believed to be producible using motion detection, in contrast to using segmentation and matching.

Using motion for segmentation

The goal of many dynamic-scene analysis systems is to recognize moving objects and to find their motion characteristics. If the system uses a stationary camera, the segmentation task generally involves separating moving components from stationary components in the scene. Then, the individual moving objects are identified based on either their velocity or on other characteristics. For systems using a moving camera, the segmentation task may be the same as above, or it may include further segmentation of the scene's stationary components by exploiting the camera's motion. Most research efforts for the segmentation of dynamic scenes have assumed a stationary camera.

Researchers in perception have argued that motion cues are helpful in segmentation. Computer vision techniques for segmenting SCMO scenes perform well compared to those for segmenting stationary scenes. In a system using a moving camera, segmentation into stationary and nonstationary components has only recently received any attention. One problem in segmenting moving-camera scenes is the fact that every surface in the scene has image plane motion. For segmenting moving-camera scenes, the motion assigned to components in the images that is due to the motion of the camera should be removed. The fact that the image motion of a surface depends on both the surface's distance from the camera and the structure of the surface complicates the situation.

Segmentation can be performed using either region-based or edge-based approaches. In this section, some approaches for segmenting dynamic scenes are discussed.

A moving edge in a frame is (1) moving and (2) an edge. Moving edges can be detected by combining spatial and temporal gradients using an AND operator. This AND can be implemented through multiplication. Thus, the time-varying edgeness of a point in a frame $E(x,y,t)$ is given by

$$\begin{aligned}
 E_{xy}(x, y, t) &= \frac{dF(x, y, t)}{ds} \cdot \frac{dF(x, y, t)}{dt} \\
 &= E(x, y, t) \cdot D(x, y)
 \end{aligned} \tag{6.5}$$

Various conventional edge detectors can be used to compute the spatial gradient, and a simple difference can be used to compute the temporal gradient. In most cases, this edge detector works effectively. By applying a threshold to the product, rather than first differencing and then applying an edge detector or first detecting edges and then computing their temporal gradient, this method overcomes the problem of slow-moving or weak edges being missed.

Using difference pictures. Difference and accumulative-difference pictures find the areas in a scene that are changing. (An area is usually changing because of object movement.) Although change detection results based on difference pictures are sensitive to noise, the areas produced by a difference picture are good places from which to start segmentation. In fact, as discussed by Jain,²¹³ segmentation of a scene can be performed with very little computation by using accumulative-difference pictures.

Using velocities. The relationship between the spatial (F_x and F_y) and temporal (F_t) gradients of intensities can be exploited for segmenting images. These gradients are related to the velocity components (u, v) by the following equation:

$$F_t = \frac{\partial F(x, y, t)}{\partial t} = -(F_x \cdot u + F_y \cdot v) \tag{6.6}$$

Assuming the same velocity component for points coming from the same object in an image, a Hough transform approach can be used to segment a scene into different moving objects.²¹⁴

Optical flow

Optical flow is the distribution of velocity, relative to the observer, over the points of an image. Optical flow carries information that is valuable in dynamic-scene analysis. Several methods of dynamic-scene analysis have been proposed; these methods assume that optical-flow information is available. Although optical flow has received significant attention from researchers, the techniques developed for computing optical flow unfortunately produce results whose quality will not allow the valuable information to be recovered. Currently used methods of computing optical flow and information that is intrinsic to optical flow are discussed in this section.

Computation of the optical flow. Optical flow is determined by the velocity vector of each pixel in an image. Several methods have been devised for calculating optical flow based on two or more frames of a sequence. These methods can be classified into two general categories: feature-based and gradient-based. If a stationary camera is used, most of the points in an image frame will have zero velocity. This is assuming that a very small subset of the scene is in motion, which is usually true. Thus, most applications for optical flow involve a moving camera.

Feature-based methods. Feature-based methods for computing optical flow first select some features in the image frames, then match these features and calculate the disparities between frames. As discussed in an earlier section, the correspondence can be solved on a stereo-image pair using relaxation. The same approach can be used to solve the correspondence problem in dynamic scenes.²¹⁵ However, the problem of selecting features and establishing correspondence is not easy. Moreover, this method produces velocity vectors only at sparse points.

Gradient-based methods. Gradient-based methods exploit the relationship between the spatial and temporal gradients of intensity. As discussed earlier, this relationship can be used to segment images based on the velocity of points. The relationship between the spatial and temporal gradients and the velocity components is

$$F_x u + F_y v + F_t = 0 \tag{6.7}$$

In this equation, F_x , F_y , and F_t can be computed directly from the image. Then, at every point in an image, there are two unknowns, u and v , yet there is only one equation. Thus, the velocity components at a point cannot be obtained without making further assumptions.

The velocity field can be assumed to vary smoothly over an image. Under this assumption, an iterative approach for computing optical flow using two or more frames can be developed. The following iterative equations are used for the computation of optical flow:

$$\begin{aligned} u &= u_{av} - F_x \frac{P}{D} \\ v &= v_{av} - F_y \frac{P}{D} \\ \text{where } P &= F_x u_{av} + F_y v_{av} + F_t \\ D &= \lambda^2 + F_x^2 + F_y^2 \end{aligned} \quad (6.8)$$

In these equations, F_x , F_y , F_t , and λ represent the spatial gradients in the x - and y -directions, the temporal gradient, and a multiplier, respectively. When only two frames are used, the computation is iterated over the same frames many times. For more than two frames, each iteration uses a new frame. Because object surfaces may be at different depths, the smoothness constraint is not satisfied at the boundaries of objects. When overlapping objects are moving in different directions, this constraint will also be violated. These abrupt changes in the velocity field at the boundaries cause problems. Figure 6.2^{5,122} illustrates the results of applying the above method of optical-flow calculation to synthetic data for a rotating checkered sphere.

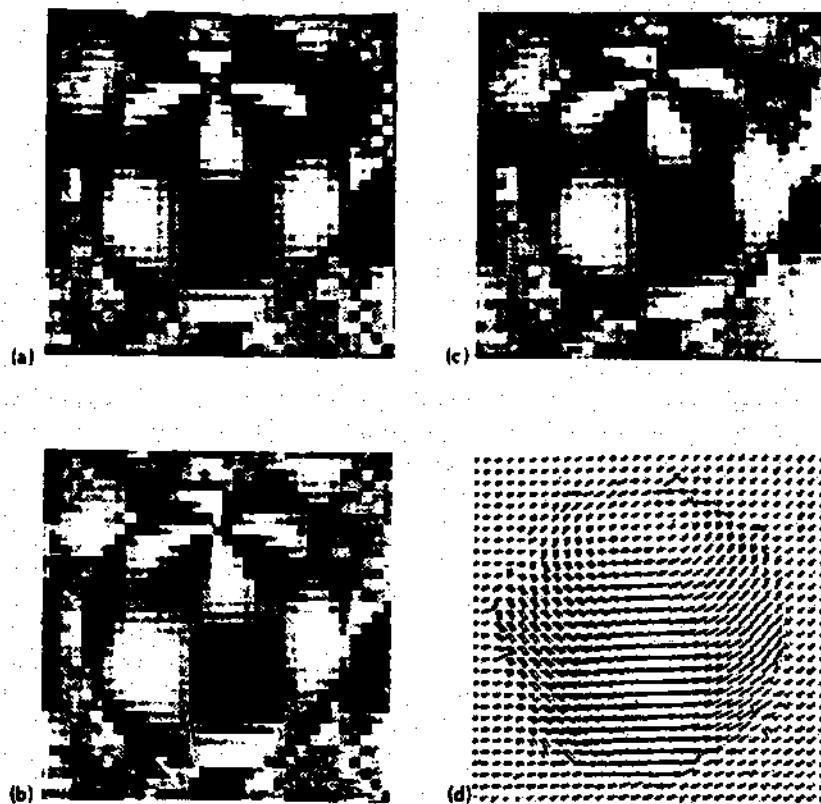


Figure 6.2: Calculation of Optical Flow; (a-c) three frames of a rotating checkered sphere; (d) calculated flow using 32 frames (from Ballard and Brown⁵; originally from Horn and Schunck¹²²)

An important fact to remember about gradient-based methods is that they assume a linear variation of intensities and compute the pointwise velocities under this assumption. This assumption is typically expected to be satisfied at edge points in images; hence, the velocity can be computed at these points.

Information in optical flow. Many researchers have studied what kind of information can be extracted from an optical-flow field, assuming that high-quality optical flow has been computed. Assume the following: (1) an environment that contains rigid, stationary surfaces at known depths and (2) an observer—i.e., the camera—that locomotes through this environment. The optical flow can be derived from the known three-dimensional structure of the environment. Thus, the structure of the environment can be obtained, in principle, from the optical-flow field that is computed.

Areas with smooth velocity gradients correspond to single surfaces in the image and contain information about the structure of the surface. Areas with large gradients contain information about occlusion and boundaries, since only different objects at different depths can move at different speeds relative to the camera. Using an observer-based coordinate system, a relationship between smooth velocity gradients and surface orientation can be derived. The orientation is specified with respect to the direction of motion of the observer.

The translational component of object motion is directed toward a point in the image called the “focus of expansion” (FOE), when the observer is approaching, or the “focus of contraction” (FOC), when the observer is receding. (Figure 6.3⁵ illustrates the FOE due to observer motion.) This point is the intersection of the direction of object motion in the image plane. Surface structure can be recovered from the first and second spatial derivatives of the translational component. The angular velocity is fully determined by the rotational component.

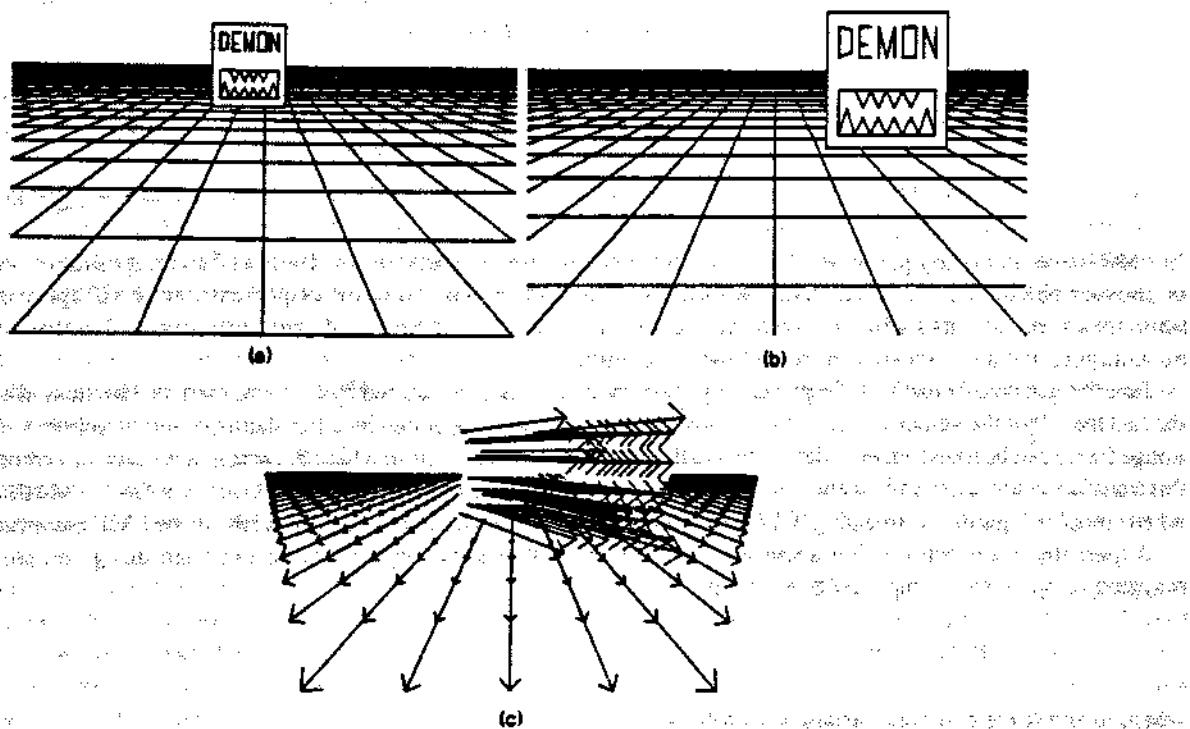


Figure 6.3: Illustration of focus of expansion (FOE) due to observer motion. (a-b) two time frames; (c) different FOEs for static floor and moving object (from D.H. Ballard and C.M. Brown, *Computer Vision*, © 1982 by Prentice-Hall. Reprinted with permission).

If the FOE is correctly determined, it can be used for computing the translational components of optical flow. Since all flow vectors meet at the FOE, their direction is already known and only their magnitudes remain to be computed. Thus, the two-dimensional problem of optical-flow computation is reduced to a one-dimensional problem. Although this fact has been noted by many researchers, it has not been applied, possibly because of the uncertainty in the proposed approaches for locating the FOE in real scenes.

Segmentation using a moving camera

If the camera is moving, then every point in the image has nonzero velocity relative to the camera. (Note that an exception would be the pathological case in which object points are moving with the same velocity as that of the camera.) The velocity relative

to the camera depends on both the velocity of the point itself and the distance of the point from the camera. Approaches based on difference pictures may be extended for segmenting moving-camera scenes. However, if the aim is to extract images of moving objects, then additional information will be required to decide whether the motion at a point is due solely to its depth or is due to a combination of its depth and its motion. Gradient-based approaches also will require additional information.

If the camera's direction of motion is known, then the FOE with respect to the stationary components in a scene can easily be computed. The FOE will have coordinates

$$x_f = \frac{dx}{dz} \quad \text{and} \quad y_f = \frac{dy}{dz} \quad (6.9)$$

in the image plane. The velocity vectors of all of the stationary points in a scene project onto the image plane so that they intersect at the FOE. A transformation with respect to the FOE can be used to simplify the task of segmentation. The ego-motion polar (EMP) transformation of an image transforms a frame $F(x,y,t)$ into $E(r,\theta,t)$ using

$$E(r,\theta,t) = F(x,y,t) \quad \text{where} \quad r = \sqrt{(x - x_f)^2 + (y - y_f)^2} \quad \text{and} \quad \theta = \tan^{-1} \left(\frac{y - y_f}{x - x_f} \right). \quad (6.10)$$

In EMP space, stationary points are displaced only along the θ -axis between the frames of an image sequence, while points on moving objects are displaced along the r -axis as well as the θ axis. Thus, the displacement in EMP space can be used to segment a scene into its stationary and nonstationary components. The results of experiments on real scenes have been very encouraging and are discussed in the following paragraphs.

Jain^{216,217} proposed a method of representing object motion in images acquired by a moving camera. His method is derived from the fact that all of the velocity vectors of stationary objects in a scene acquired by a translating observer intersect at the FOE. An image frame can be transformed, with respect to the FOE, to a second frame in which the abscissa is r and the ordinate is θ . Under this transformation, a dynamic scene can be segmented into its moving and stationary components, as discussed earlier. Moreover, when complex logarithmic mapping (CLM) is performed about the FOE, interesting properties in the EMP space can be obtained.

Apparently, more information about moving, as well as stationary, objects can be extracted using complex logarithmic mapping as opposed to simple polar mapping.

$$\omega = \log \alpha \quad (6.11)$$

where ω and α are complex variables, as follows:

$$\begin{aligned} \alpha &= x + iy \\ &= r(\cos\theta + i\sin\theta) \\ &\approx re^{i\theta} \end{aligned} \quad \text{and} \quad \omega = u(z) + iv(z) \quad (6.12)$$

Under this transformation, it can be shown that

$$\begin{aligned} u(r,\theta) &= \log r \\ v(r,\theta) &= \theta. \end{aligned} \quad (6.13)$$

These results state the following fact: If the observer is moving, the horizontal displacement of a stationary point in CLM-space depends only on the depth of the point and, furthermore, the vertical displacement is zero. This fact is apparently very useful, not only in segmenting dynamic scenes into moving and stationary components, but also in determining the depth of stationary points. Schwartz²¹⁸⁻²²¹ and Cavanaugh^{222,223} studied this complex logarithmic mapping in the context of biological

vision systems. Schwartz found that retino-striate mapping can be approximated by a complex log function. Complex logarithmic mapping appears to be responsible for size, rotation, and projection invariances in biological systems. Cavanaugh²² argues that such mapping is justified only in limited cases. Jain²¹⁷ showed that some of the limitations with respect to the projection invariance may be removed if the mapping is obtained with respect to the FOE. The complex EMP-space produces an observer-centered representation of an image sequence through the use of the ego-motion of the observer. This representation may play an important role in MCMO dynamic scenes.

Recovering three-dimensional information

The interpretation of two-dimensional displacements in terms of three-dimensional motion is complicated. During the picture formation process, information loss occurs, which results in a two-dimensional projection of three-dimensional events. Recovery of three-dimensional-motion parameters and three-dimensional object structure has been an active research area. An assumption about the rigidity of objects may be helpful in recovering the structure of objects. The rigidity assumption states that any set of elements undergoing a two-dimensional transformation that has a unique interpretation as a rigid body moving in space should be so interpreted. Research in human perception suggests that the human visual system exploits this assumption.

Approaches to recovering three-dimensional structure from image sequences can be divided into two general classes, as follows:

- (1) Recovering structure using tokens and
- (2) Recovering structure using the trajectories.

Recovering structure using tokens. Suppose that an interest operator is applied to consecutive frames of a sequence and that some interesting points (or "tokens") — such as corners — have been extracted. Also suppose that the correspondence problem between interesting points has been solved, using any method discussed earlier. If token correspondence has been established, then the three-dimensional location of four noncoplanar points can be recovered from their orthogonal projections, giving an implicit three-dimensional structure for an object. If the points are noncoplanar, then a unique structure can be recovered; if the points are coplanar, then the structure can be recovered to a reflection.

Feature-based methods for the recovery of three-dimensional structure or for the estimation of motion parameters require two difficult steps, as follows:

- (1) Determining the precise location of points (or tokens) in an image and
- (2) Determining the correspondence between points (or tokens) in an image pair.

If interest operators are applied based on small neighborhoods, then the number of tokens extracted in a real image is very large, making correspondence a difficult problem. If interest operators are applied based on large neighborhoods and higher order gray-level characteristics, then the number of tokens extracted is more reasonable for determining correspondences; however, the locations of tokens may not be precise. Apparently, results obtained using the methods just discussed may not be reliable, even if the location is obtained to pixel resolution.

Recovering structure using trajectories. All of the methods just discussed depend on a set of points in two or three frames. If a token is traced over several frames by solving correspondences, a two-dimensional trajectory of the token is obtained. Methods for recovering three-dimensional structure and motion from trajectories apparently may be more reliable than methods based on sets of features in a few frames. Curve-fitting techniques can be used to interpolate a trajectory in order to obtain better resolution of the two-dimensional path. Moreover, the correspondence problem may be simplified by considering more than two frames and extending relaxation across the frames.

Motion understanding

Dynamic-scene analysis may result in the extraction of

- Images of moving objects;
- Three-dimensional object structures; and
- Frame-to-frame object displacements.

In many applications, the aim may be to name the moving objects and to describe the events taking place in the scene. Object recognition may be performed based on an image or on the three-dimensional structure of the object. Apparently, motion characteristics of objects may also help in recognition. Different objects have different kinds of motion: cars run and airplanes fly. The motion characteristics of objects are difficult to obtain directly from frame-to-frame displacements. Much human recognition and object analysis require neither fine detail nor precise analysis of small details or parts. Details are only brought into play when the object is the focus of attention.

An object that has complex motion in each of its several parts can initially be abstracted to a simple moving block undergoing rigid motion. The abstracted motion will be a simple translation or rotation, whereas the real motion is very complex. At the next level of analysis, the motion of the separate parts of the object can be analyzed; knowledge about the abstract motion of the object may be helpful in analyzing the motion of the individual components of the object. Apparently, a correct approach to determining the more detailed motion of lower level parts is to compensate somehow for the known motion of the higher level abstraction of the object.

Research trends

One of the first tasks in dynamic-scene analysis is detection of moving objects. Thompson and Pong²²⁴ conclude that detection using visual information alone is quite difficult, especially when the camera is moving. They develop detection algorithms to use when information about camera motion and/or scene structure is also available. Jain²²⁵ and Liou and Jain²²⁶ present approaches to motion detection based on the transformation of image sequences using knowledge of camera motion.

Fourteen papers were selected that are representative of papers dealing with the topics covered in this chapter. Six are included in this book and the other eight in the companion book *Computer Vision: Advances and Applications*. We begin the first chapter in *Principles* with the paper by Horn and Schunck entitled "Determining Optical Flow" which describes a robust method for finding optical-flow patterns that is based on a constraint on intensity gradients. Since flow velocity has two components, an additional constraint is needed. Smoothness of the displacement vector field is used in their iterative algorithm as the second constraint. However, a difficulty is presented because such smoothness constraints are violated along occluding contours. In the paper entitled "An Investigation of Smoothness Constraints for the Estimation of Displacement Vector Fields From Image Sequences," Nagel and Enkelmann in *Principles* overcome this difficulty by formulating and evaluating an oriented-smoothness constraint. In *Advances*, Meygret and Thonnat in "Segmentation of Optical Flow and Three-Dimensional Data for the Interpretation of Mobile Objects" describe an approach for the detection of three-dimensional moving objects using stereo and motion information. In their approach, the scene is detected and then it is interpreted in terms of isolated rigid or deformable objects.

Determination of optical flow has been a very active research area; Aisbett,²²⁷ Heeger,²²⁸ Mitiche et al.,²²⁹ and Nagel²³⁰ present some recent approaches. In *Principles*, Barnard and Thompson in "Disparity Analysis of Images," introduce a feature-based matching algorithm to estimate geometric disparity between two images. Small, distinct features are detected in both images, and initial estimates of the probability of a match between candidate pairs of features in the two images are obtained. A relaxation algorithm that uses local continuity of disparity is applied to improve these estimates. Clocksin²³¹ and Longuet-Higgins and Prazdny²³² describe the significance of optical flow and its potential in extracting information about surface orientation and structure.

The nonlinear system of equations relating optical flow and its first- and second-order derivatives to object structure and motion parameters are derived, and solutions for planar and curved surface patches are obtained, by Waxman et al. in the paper entitled "Closed-Form Solutions to Image Flow Equations for Three-Dimensional Structure and Motion" found in *Advances*. Optical flow (which is a measure of changes in image brightness) and the underlying motion field (which is the perspective projection of the true three-dimensional velocity field of moving surfaces in space) are in general different. For example, a smooth Lambertian sphere rotating in space has a motion field, although it has no corresponding optical-flow field. In "Motion Field and Optical Flow: Qualitative Properties," Verri and Poggio in *Advances* show that stable qualitative properties of the motion field, which contain useful information about the three-dimensional motion and structure of objects, can be obtained from the optical flow. In "Determining Three-Dimensional Motion and Structure From Optical Flow Generated by Several Moving Objects," Adiv describes in *Advances* an approach for interpreting sparse, noisy, and partially incorrect flow fields. In this approach, connected segments of flow vectors subject to the rigid motion of a roughly planar surface constraint are first obtained; then they are grouped under the hypothesis of a single rigid moving object. The hypotheses are tested for compatibility with all segments in the group in order to recover three-dimensional motion parameters. Ullman²³³ proposed structure-from-motion approaches using N points in M frames. This research direction has been very popular in the last few years. In "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects With Curved Surfaces," Tsai and Huang in *Advances* show that seven point correspondences in two perspective views are sufficient to determine the three-dimensional motion parameters of a curved surface.

However, finding solutions to nonlinear equations is required. They show that, by obtaining eight point correspondences, motion parameters can be determined by solving eight linear equations and a singular value decomposition. Also in *Advances*, the paper entitled "Motion and Structure From Two Perspective Views: Algorithm, Error Analysis, and Error Estimation," by Weng et al., presents a method for estimating motion parameters and scene structure. Weng et al. discuss the performance of Tsai and Huang's algorithm in the presence of errors. Faugeras and Maybank,²³⁴ Mitiche et al.,²³⁵ and Jerian and Jain²³⁶ present more recent structure-from-motion approaches. Jerian and Jain²³⁷ give a classification of several approaches, with their convergence properties and noise sensitivities.

In the papers just discussed, the techniques described attempt to derive motion parameters using two images. In *Principles*, Sethi and Jain in "Finding Trajectories of Feature Points in a Monocular Image Sequence," consider a sequence of images to determine trajectories of feature points. By applying the smoothness-of-motion constraint on the image sequences, they formulate the correspondence problem as an optimization problem, and they describe an iterative algorithm to solve this problem. The presence of noise in the image coordinates of the object match points results in inaccuracies in the estimation of motion parameters. These inaccuracies can be alleviated by using a larger number of match points to obtain an overdetermined set of estimation equations. An alternative technique to alleviate these inaccuracies is to use a larger number of image frames. Broida and Chellappa describe such a technique (which is based on an iterated extended Kalman filter) in *Advances* in the paper entitled "Estimation of Object Motion Parameters From Noisy Images."

The use of a large number of image frames taken at short intervals also helps in minimizing the correspondence problem, since the amount of change in successive images is expected to be very small. This concept has led to so-called "epipolar-plane image analysis," in which images are acquired using a moving camera. Explicit representation of both the spatial and temporal structures of such image sequences is captured in a spatio-temporal surface. A tracking mechanism that operates locally on these evolving surfaces to obtain three-dimensional scene reconstruction is described by Baker and Bolles in *Principles* in the paper entitled "Generalizing Epipolar-Plane Image Analysis on the Spatio-Temporal Surface." Several other approaches based on the spatio-temporal image solid are discussed by Heeger,²²⁸ Watson and Ahumada, Jr.,²³⁸ and Liou and Jain.²²⁶

Determining optical flow and solving the correspondence problem have been two difficult problems in dynamic vision. In the last few years, several approaches that bypass optical flow and correspondence for direct computation of motion properties have been proposed by Jain,²¹³ Aloimonos and Rigoutsos,^{239,240} and Negahdaripour and Horn.²⁴¹ These approaches appear promising.

In "Analysis of a Sequence of Stereo Scenes Containing Multiple Moving Objects Using Rigidity Constraints," Zhang et al. in *Principles* approach the motion determination problem as a stereo-matching and motion-estimation problem. Rigidity constraints are used to register two stereo frames. First, ego-motion is determined; then, it is taken into account in computing the motion of objects. The stereo-and-motion approach is also used by Grosso et al. in their system, which is described in "Three-Dimensional Object Reconstruction Using Stereo and Motion" in *Advances*. However, in their system, the objects remain stationary, and multiple views are obtained by moving the cameras around the objects while maintaining the direction of gaze fixed toward a point in space. A survey of many dynamic-scene analysis methods is given by Aggarwal and Nandakumar.²⁴² Tsotsos et al.²⁴³ describe a framework for the abstraction of motion concepts from image sequences. Included in the framework are semantic nets for knowledge representation and associated algorithms operating in a competing and cooperating feedback mode.

References Cited Chapter 6

211. H.H. Nagel, "Formation of an Object Concept by Analysis of Systematic Time Variation in the Optically Perceptible Environment," *Computer Graphics and Image Processing*, Vol. 7, 1978, pp. 149-194.
212. R.C. Jain, "Dynamic Scene Analysis Using Pixel-Based Processes," *Computer*, Vol. 12, No. 1, 1981, pp. 12-18.
213. R.C. Jain, "Segmentation of Frame Sequence Obtained by a Moving Observer," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 6, No. 5, 1984, pp. 624-629.
214. C.L. Fennema and W.B. Thompson, "Velocity Determination in Scenes Containing Several Moving Objects," *Computer Graphics and Image Processing*, Vol. 9, No. 4, 1979, pp. 301-315.
215. S.T. Barnard and W.B. Thompson, "Disparity Analysis of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 2, No. 4, 1980, pp. 822-825.
216. R.C. Jain, "Directed Computation of the Focus of Expansion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 5, No. 1, 1983, pp. 58-64.
217. R.C. Jain, "Complex Logarithmic Mapping and the Focus of Expansion," *SIGGRAPH/SIGART Workshop on Motion: Representation and Control*, 1983, pp. 42-49.
218. E.L. Schwartz, "A Quantitative Model of the Functional Architecture of Human Striate Cortex with Application to Vision Illusion and Cortical Texture Analysis," *Biological Cybernetics*, Vol. 37, 1980, pp. 63-76.
219. Schwartz, E.L., "Computational Anatomy and Functional Architecture of Striate Cortex: A Spatial Mapping Approach to Perceptual Coding," *Vision Research*, Vol. 20, 1980, pp. 645-669.
220. E.L. Schwartz, "Cortical Anatomy, Size Invariance, and Spatial Frequency Analysis," *Perception*, Vol. 10, 1981, pp. 455-468.
221. E.L. Schwartz, "Columnar Architecture and Computational Anatomy in Primate Visual Cortex: Segmentation and Feature Extraction via Spatial Frequency Coded Difference Mapping," *Biological Cybernetics*, Vol. 42, 1982, pp. 157-168.
222. P. Cavanaugh, "Size and Position Invariance in the Vision System," *Perception*, Vol. 7, 1978, pp. 167-177.
223. P. Cavanaugh, "Size Invariance: Reply to Schwartz," *Perception*, Vol. 10, 1981, pp. 469-474.
224. W.B. Thompson and T.C. Pong, "Detecting Moving Objects," *Int'l J. Computer Vision*, Vol. 4, 1990, pp. 39-57.
225. R.C. Jain, "Difference and Accumulative Difference Pictures in Dynamic Scene Analysis," *Image and Vision Computing*, Vol. 2, No. 2, 1984, pp. 99-108.
226. S.P. Liou and R.C. Jain, "Motion Detection in Spatio-Temporal Space," *Computer Vision, Graphics, and Image Processing*, Vol. 45, 1989, pp. 227-250.
227. J. Aisbett, "Optical Flow with an Intensity-Weighted Smoothing," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 11, No. 5, 1989, pp. 512-522.
228. D.J. Heeger, "Optical Flow from Spatiotemporal Filters," *Int'l J. Computer Vision*, 1988, pp. 279-301.
229. A. Mitiche, Y.F. Wang, and J.K. Aggarwal, "Experiments in Computing Optical Flow with the Gradient-Based, Multiconstraint Method," *Pattern Recognition*, Vol. 20, No. 2, 1987, pp. 173-179.
230. H.H. Nagel, "On the Estimation of Optical Flow: Relations between Different Approaches and Some New Results," *Artificial Intelligence*, Vol. 33, 1987, pp. 299-324.
231. W.F. Clocksin, "Perception of Surface Slant and Edge Labels from Optical Flow: A Computational Approach," *Perception*, Vol. 9, No. 3, 1980, pp. 253-269.
232. H.C. Longuet-Higgins and K. Prazdny, "The Interpretation of a Moving Retinal Image," *Proc. Royal Society of London, B*, 1980, Vol. 208, pp. 385-397.
233. S. Ullman, *The Interpretation of Visual Motion*, MIT Press, Cambridge, Mass., 1979.
234. O.D. Faugeras and S. Maybank, "Motion from Point Matches: Multiplicity of Solutions," *Int'l J. Computer Vision*, Vol. 4, No. 3, 1990, pp. 225-246.
235. A. Mitiche, O. Faugeras, and J.K. Aggarwal, "Counting Straight Lines," *Computer Vision, Graphics, and Image Processing*, Vol. 47, 1989, pp. 353-360.
236. C. Jerian and R.C. Jain, "Structure from Motion: A Critical Analysis of Methods," *IEEE Trans. Systems, Man, and Cybernetics*, 1991.
237. C. Jerian and R.C. Jain, "Polynomial Methods for Structure from Motion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 12, No. 12, 1990, pp. 1150-1166.
238. A.B. Watson and A.J. Ahumada, Jr., "Model of Human Visual-Motion Sensing," *J. Opt. Soc. Am. A.*, Vol. 2, No. 2, 1985, pp. 322-342.
239. J.Y. Aloimonos and I. Rigoutsos, "Determining the 3-D Motion of a Rigid Surface Patch Without Correspondence Under Perspective Projection. I. Planar Surfaces; II Curved Surfaces," *Proc. Nat'l Conf. on Artificial Intelligence*, 1986, pp. 681-688.
240. J.Y. Aloimonos and I. Rigoutsos, "Determining the 3-D Motion of a Rigid Planar Patch Without Correspondence, Under Perspective Projection," *Proc. Workshop on Motion*, 1986, pp. 167-174.
241. S. Negahdaripour and B.K.P. Horn, "Direct Passive Navigation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 9, No. 1, 1987, pp. 168-176.
242. J.K. Aggarwal and N. Nandhakumar, "On the Computation of Motion from Sequences of Images: A Review," *Proc. IEEE*, Vol. 76, No. 8, IEEE Press, New York, N.Y., 1988, pp. 917-935.
243. J.K. Tsotsos, et al., "A Framework for Visual Motion Understanding," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 2, No. 6, 1980, pp. 5653-573.

"Determining Optical Flow" by B.K.P. Horn and B.G. Schunck
from *Artificial Intelligence*, Volume 17, 1981, pages 185-203.
Copyright 1981 by North-Holland Publishing Company, reprinted
with permission.

ARTIFICIAL INTELLIGENCE

Determining Optical Flow

Berthold K.P. Horn and Brian G. Schunck

Artificial Intelligence Laboratory, Massachusetts Institute of
Technology, Cambridge, MA 02139, U.S.A.

ABSTRACT

Optical flow cannot be computed locally, since only one independent measurement is available from the image sequence at a point, while the flow velocity has two components. A second constraint is needed. A method for finding the optical flow pattern is presented which assumes that the apparent velocity of the brightness pattern varies smoothly almost everywhere in the image. An iterative implementation is shown which successfully computes the optical flow for a number of synthetic image sequences. The algorithm is robust in that it can handle image sequences that are quantized rather coarsely in space and time. It is also insensitive to quantization of brightness levels and additive noise. Examples are included where the assumption of smoothness is violated at singular points or along lines in the image.

1. Introduction

Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image. Optical flow can arise from relative motion of objects and the viewer [6, 7]. Consequently, optical flow can give important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement [8]. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects [27]. Attempts have been made to perform such segmentation using differences between successive image frames [15, 16, 17, 20, 25]. Several papers address the problem of recovering the motions of objects relative to the viewer from the optical flow [10, 18, 19, 21, 29]. Some recent papers provide a clear exposition of this enterprise [30, 31]. The mathematics can be made rather difficult, by the way, by choosing an inconvenient coordinate system. In some cases, information about the shape of an object may also be recovered [3, 18, 19].

These papers begin by assuming that the optical flow has already been determined. Although some reference has been made to schemes for comput-

Artificial Intelligence 17 (1981) 185-203

0004-3702/81/0000-0000/\$02.50 © North-Holland

ing the flow from successive views of a scene [5, 10], the specifics of a scheme for determining the flow from the image have not been described. Related work has been done in an attempt to formulate a model for the short range motion detection processes in human vision [2, 22]. The pixel recursive equations of Netravali and Robbins [28], designed for coding motion in television signals, bear some similarity to the iterative equations developed in this paper. A recent review [26] of computational techniques for the analysis of image sequences contains over 150 references.

The optical flow cannot be computed at a point in the image independently of neighboring points without introducing additional constraints, because the velocity field at each image point has two components while the change in image brightness at a point in the image plane due to motion yields only one constraint. Consider, for example, a patch of a pattern where brightness¹ varies as a function of one image coordinate but not the other. Movement of the pattern in one direction alters the brightness at a particular point, but motion in the other direction yields no change. Thus components of movement in the latter direction cannot be determined locally.

2. Relationship to Object Motion

The relationship between the optical flow in the image plane and the velocities of objects in the three dimensional world is not necessarily obvious. We perceive motion when a changing picture is projected onto a stationary screen, for example. Conversely, a moving object may give rise to a constant brightness pattern. Consider, for example, a uniform sphere which exhibits shading because its surface elements are oriented in many different directions. Yet, when it is rotated, the optical flow is zero at all points in the image, since the shading does not move with the surface. Also, specular reflections move with a velocity characteristic of the virtual image, not the surface in which light is reflected.

For convenience, we tackle a particularly simple world where the apparent velocity of brightness patterns can be directly identified with the movement of surfaces in the scene.

3. The Restricted Problem Domain

To avoid variations in brightness due to shading effects we initially assume that the surface being imaged is flat. We further assume that the incident illumination is uniform across the surface. The brightness at a point in the image is then proportional to the reflectance of the surface at the corresponding point on the object. Also, we assume at first that reflectance varies smoothly and has no spatial discontinuities. This latter condition assures us that the image brightness is differentiable. We exclude situations where objects occlude one another, in part, because discontinuities in reflectance are found at object boundaries. In two of the experiments discussed later, some of the problems occasioned by occluding edges are exposed.

In the simple situation described, the motion of the brightness patterns in the image is determined directly by the motions of corresponding points on the surface of the object. Computing the velocities of points on the object is a matter of simple geometry once the optical flow is known.

¹In this paper, the term brightness means image irradiance. The brightness pattern is the distribution of irradiance in the image.

4. Constraints

We will derive an equation that relates the change in image brightness at a point to the motion of the brightness pattern. Let the image brightness at the point (x, y) in the image plane at time t be denoted by $E(x, y, t)$. Now consider what happens when the pattern moves. The brightness of a particular point in the pattern is constant, so that

$$\frac{dE}{dt} = 0.$$

Using the chain rule for differentiation we see that,

$$\frac{\partial E}{\partial x} \frac{dx}{dt} + \frac{\partial E}{\partial y} \frac{dy}{dt} + \frac{\partial E}{\partial t} = 0.$$

(See Appendix A for a more detailed derivation.) If we let

$$u = \frac{dx}{dt} \quad \text{and} \quad v = \frac{dy}{dt},$$

then it is easy to see that we have a single linear equation in the two unknowns u and v ,

$$E_x u + E_y v + E_t = 0,$$

where we have also introduced the additional abbreviations E_x , E_y , and E_t for the partial derivatives of image brightness with respect to x , y and t , respectively. The constraint on the local flow velocity expressed by this equation is illustrated in Fig. 1. Writing the equation in still another way,

$$(E_x, E_y) \cdot (u, v) = -E_t.$$

Thus the component of the movement in the direction of the brightness gradient (E_x, E_y) equals

$$\frac{E_t}{\sqrt{E_x^2 + E_y^2}}.$$

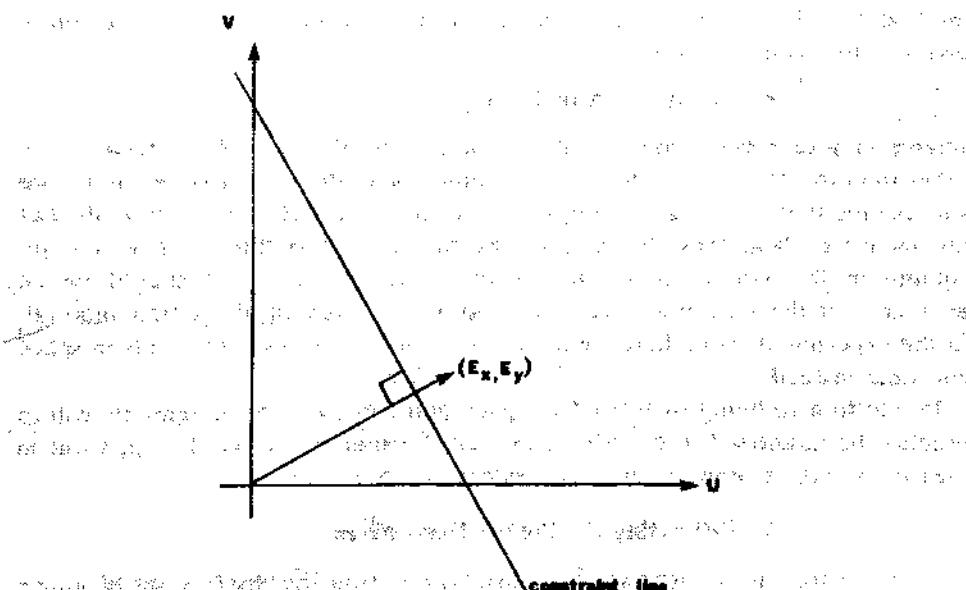


FIG. 1. The basic rate of change of image brightness equation constrains the optical flow velocity. The velocity (u, v) has to lie along a line perpendicular to the brightness gradient vector (E_x, E_y) . The distance of this line from the origin equals E_t divided by the magnitude of (E_x, E_y) .

We cannot, however, determine the component of the movement in the direction of the iso-brightness contours, at right angles to the brightness gradient. As a consequence, the flow velocity (u, v) cannot be computed locally without introducing additional constraints.

5. The Smoothness Constraint

If every point of the brightness pattern can move independently, there is little hope of recovering the velocities. More commonly we view opaque objects of finite size undergoing rigid motion or deformation. In this case neighboring points on the objects have similar velocities and the velocity field of the brightness patterns in the image varies smoothly almost everywhere. Discontinuities in flow can be expected where one object occludes another. An algorithm based on a smoothness constraint is likely to have difficulties with occluding edges as a result.

One way to express the additional constraint is to minimize the square of the magnitude of the gradient of the optical flow velocity:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \quad \text{and} \quad \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2.$$

Another measure of the smoothness of the optical flow field is the sum of the squares of the Laplacians of the x - and y -components of the flow. The Laplacians of u and v are defined as

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad \text{and} \quad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}.$$

In simple situations, both Laplacians are zero. If the viewer translates parallel to a flat object, rotates about a line perpendicular to the surface or travels orthogonally to the surface, then the second partial derivatives of both u and v vanish (assuming perspective projection in the image formation).

We will use here the square of the magnitude of the gradient as smoothness measure. Note that our approach is in contrast with that of Fennema and Thompson [5], who propose an algorithm that incorporates additional assumptions such as constant flow velocities within discrete regions of the image. Their method, based on cluster analysis, cannot deal with rotating objects, since these give rise to a continuum of flow velocities.

6. Quantization and Noise

Images may be sampled at intervals on a fixed grid of points. While tessellations other than the obvious one have certain advantages [9, 23], for convenience we will assume that the image is sampled on a square grid at regular intervals. Let the measured brightness be $E_{i,j,k}$ at the intersection of the i th row and j th column in the k th image frame. Ideally, each measurement should be an average over the area of a picture cell and over the length of the time interval. In the experiments cited here we have taken samples at discrete points in space and time instead.

In addition to being quantized in space and time, the measurements will in practice be quantized in brightness as well. Further, noise will be apparent in measurements obtained in any real system.

7. Estimating the Partial Derivatives

We must estimate the derivatives of brightness from the discrete set of image brightness measurements available. It is important that the estimates of E_x , E_y , and E_t be consistent. That is, they should all refer to the same point in the image at the same time. While there are many formulas for approximate

differentiation [4,11], we will use a set which gives us an estimate of E_x , E_y , E_t at a point in the center of a cube formed by eight measurements. The relationship in space and time between these measurements is shown in Fig. 2. Each of the estimates is the average of four first differences taken over adjacent measurements in the cube.

$$\begin{aligned} E_x &= \frac{1}{4}(E_{ij+1,k} - E_{ij,k} + E_{i+1,j+1,k} - E_{i+1,j,k} \\ &\quad + E_{ij+1,k+1} - E_{ij,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}), \\ E_y &= \frac{1}{4}(E_{i+1,j,k} - E_{ij,k} + E_{i+1,j+1,k} - E_{ij+1,k} \\ &\quad + E_{i+1,j,k+1} - E_{ij,k+1} + E_{i+1,j+1,k+1} - E_{ij+1,k+1}), \\ E_t &= \frac{1}{4}(E_{ij,k+1} - E_{ij,k} + E_{i+1,j,k+1} - E_{i+1,j,k} \\ &\quad + E_{ij+1,k+1} - E_{ij+1,k} + E_{i+1,j+1,k+1} - E_{i+1,j+1,k}). \end{aligned}$$

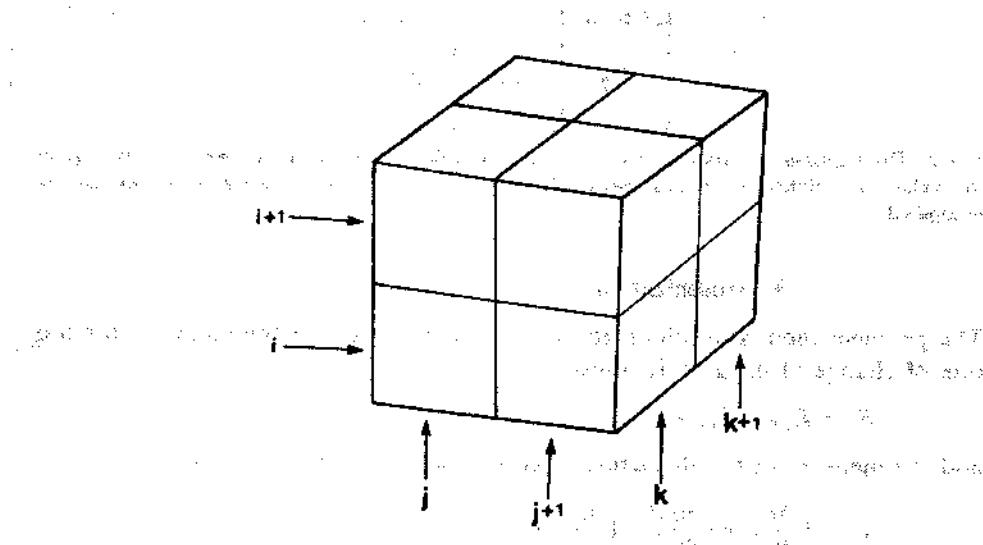


FIG. 2. The three partial derivatives of images brightness at the center of the cube are each estimated from the average of first differences along four parallel edges of the cube. Here the column index i corresponds to the x direction in the image, the row index j to the y direction, while k lies in the time direction.

Here the unit of length is the grid spacing interval in each image frame and the unit of time is the image frame sampling period. We avoid estimation formulae with larger support, since these typically are equivalent to formulae of small support applied to smoothed images [14].

8. Estimating the Laplacian of the Flow Velocities

We also need to approximate the Laplacians of u and v . One convenient approximation takes the following form

$$\nabla^2 u \approx \kappa(\bar{u}_{ij,k} - u_{ij,k}) \text{ and } \nabla^2 v \approx \kappa(\bar{v}_{ij,k} - v_{ij,k}),$$

where the local averages \bar{u} and \bar{v} are defined as follows

$$\begin{aligned} \bar{u}_{ij,k} &= \frac{1}{8}(u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k} \\ &\quad + \frac{1}{12}(u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}), \end{aligned}$$

$$\begin{aligned} \bar{v}_{ij,k} &= \frac{1}{8}(v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k} \\ &\quad + \frac{1}{12}(v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}). \end{aligned}$$

The proportionality factor κ equals 3 if the average is computed as shown and we again assume that the unit of length equals the grid spacing interval. Fig. 3 illustrates the assignment of weights to neighboring points.

$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$
$\frac{1}{6}$	-1	$\frac{1}{6}$
$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$

FIG. 3. The Laplacian is estimated by subtracting the value at a point from a weighted average of the values at neighboring points. Shown here are suitable weights by which values can be multiplied.

9. Minimization

The problem then is to minimize the sum of the errors in the equation for the rate of change of image brightness,

$$\mathcal{E}_b = E_x u + E_y v + E_n$$

and the measure of the departure from smoothness in the velocity flow,

$$g_c^2 = \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2.$$

What should be the relative weight of these two factors? In practice the image brightness measurements will be corrupted by quantization error and noise so that we cannot expect \mathcal{E}_b to be identically zero. This quantity will tend to have an error magnitude that is proportional to the noise in the measurement. This fact guides us in choosing a suitable weighting factor, denoted by α^2 , as will be seen later.

Let the total error to be minimized be

$$g^2 = \int \int (\alpha^2 g_c^2 + \mathcal{E}_b^2) dx dy.$$

The minimization is to be accomplished by finding suitable values for the optical flow velocity (u, v) . Using the calculus of variation we obtain

$$E_x^2 u + E_x E_y v = \alpha^2 \nabla^2 u - E_x E_n$$

$$E_x E_y u + E_y^2 v = \alpha^2 \nabla^2 v - E_y E_n$$

Using the approximation to the Laplacian introduced in the previous section,

$$(\alpha^2 + E_x^2) u + E_x E_y v = (\alpha^2 \bar{u} - E_x E_n),$$

$$E_x E_y u + (\alpha^2 + E_y^2) v = (\alpha^2 \bar{v} - E_y E_n).$$

The determinant of the coefficient matrix equals $\alpha^2(\alpha^2 + E_x^2 + E_y^2)$. Solving for u and v we find that

$$(\alpha^2 + E_x^2 + E_y^2) u = +(\alpha^2 + E_y^2) \bar{u} - E_x E_y \bar{v} - E_x E_n,$$

$$(\alpha^2 + E_x^2 + E_y^2) v = -E_x E_y \bar{u} + (\alpha^2 + E_x^2) \bar{v} - E_y E_n.$$

10. Difference of Flow at a Point from Local Average

These equations can be written in the alternate form

$$(\alpha^2 + E_x^2 + E_y^2)(u - \bar{u}) = -E_x[E_x\bar{u} + E_y\bar{v} + E_t],$$

$$(\alpha^2 + E_x^2 + E_y^2)(v - \bar{v}) = -E_y[E_x\bar{u} + E_y\bar{v} + E_t].$$

This shows that the value of the flow velocity (u, v) which minimizes the error \mathcal{E}^2 lies in the direction towards the constraint line along a line that intersects the constraint line at right angles. This relationship is illustrated geometrically in Fig. 4. The distance from the local average is proportional to the error in the basic formula for rate of change of brightness when \bar{u} , \bar{v} are substituted for u and v . Finally we can see that α^2 plays a significant role only for areas where the brightness gradient is small, preventing haphazard adjustments to the estimated flow velocity occasioned by noise in the estimated derivatives. This parameter should be roughly equal to the expected noise in the estimate of $E_x^2 + E_y^2$.

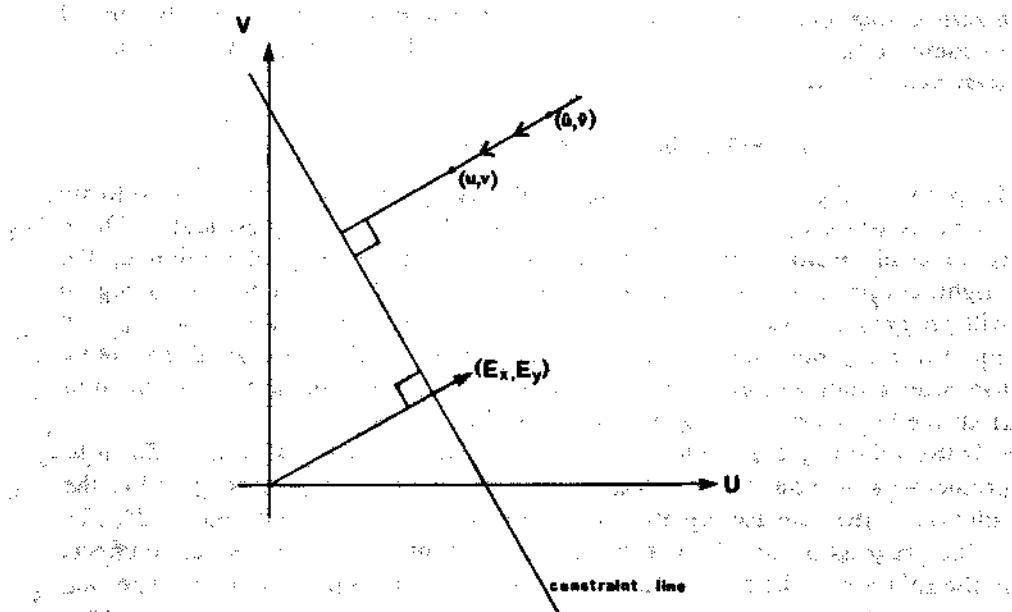


FIG. 4. The value of the flow velocity which minimizes the error lies on a line drawn from the local average of the flow velocity perpendicular to the constraint line.

11. Constrained Minimization

When we allow α^2 to tend to zero we obtain the solution to a constrained minimization problem. Applying the method of Lagrange multipliers [33, 34] to the problem of minimizing \mathcal{E}_c^2 while maintaining $\mathcal{E}_b = 0$ leads to

$$E_y \nabla^2 u = E_x \nabla^2 v, \quad E_x u + E_y v + E_t = 0$$

Approximating the Laplacian by the difference of the velocity at a point and the average of its neighbors then gives us

$$(E_x^2 + E_y^2)(u - \bar{u}) = -E_x[E_x\bar{u} + E_y\bar{v} + E_t],$$

$$(E_x^2 + E_y^2)(v - \bar{v}) = -E_y[E_x\bar{u} + E_y\bar{v} + E_t].$$

Referring again to Fig. 4, we note that the point computed here lies at the intersection of the constraint line and the line at right angles through the point (\bar{u}, \bar{v}) . We will not use these equations since we do expect errors in the estimation of the partial derivatives.

12. Iterative Solution

We now have a pair of equations for each point in the image. It would be very costly to solve these equations simultaneously by one of the standard methods, such as Gauss-Jordan elimination [11, 13]. The corresponding matrix is sparse and very large since the number of rows and columns equals twice the number of picture cells in the image. Iterative methods, such as the Gauss-Seidel method [11, 13], suggest themselves. We can compute a new set of velocity estimates (u^{n+1}, v^{n+1}) from the estimated derivatives and the average of the previous velocity estimates (u^n, v^n) by

$$\begin{aligned} u^{n+1} &= \bar{u}^n - E_x [E_x \bar{u}^n + E_y \bar{v}^n + E_t] / (\alpha^2 + E_x^2 + E_y^2), \\ v^{n+1} &= \bar{v}^n - E_y [E_x \bar{u}^n + E_y \bar{v}^n + E_t] / (\alpha^2 + E_x^2 + E_y^2). \end{aligned}$$

(It is interesting to note that the new estimates at a particular point do not depend directly on the previous estimates at the same point.)

The natural boundary conditions for the variational problem turns out to be a zero normal derivative. At the edge of the image, some of the points needed to compute the local average of velocity lie outside the image. Here we simply copy velocities from adjacent points further in.

13. Filling In Uniform Regions

In parts of the image where the brightness gradient is zero, the velocity estimates will simply be averages of the neighboring velocity estimates. There is no local information to constrain the apparent velocity of motion of the brightness pattern in these areas. Eventually the values around such a region will propagate inwards. If the velocities on the border of the region are all equal to the same value, then points in the region will be assigned that value too, after a sufficient number of iterations. Velocity information is thus filled in from the boundary of a region of constant brightness.

If the values on the border are not all the same, it is a little more difficult to predict what will happen. In all cases, the values filled in will correspond to the solution of the Laplace equation for the given boundary condition [1, 24, 32].

The progress of this filling-in phenomena is similar to the propagation effects in the solution of the heat equation for a uniform flat plate, where the time rate of change of temperature is proportional to the Laplacian. This gives us a means of understanding the iterative method in physical terms and of estimating the number of steps required. The number of iterations should be larger than the number of picture cells across the largest region that must be filled in. If the size of such regions is not known in advance one may use the cross-section of the whole image as a conservative estimate.

14. Tightness of Constraint

When brightness in a region is a linear function of the image coordinates we can only obtain the component of optical flow in the direction of the gradient. The component at right angles is filled in from the boundary of the region as described before. In general the solution is most accurately determined in regions where the brightness gradient is not too small and varies in direction from point to point. Information which constrains both components of the optical flow velocity is then available in a relatively small neighborhood. Too violent fluctuations in brightness on the other hand are not desirable since the estimates of the derivatives will be corrupted as the result of undersampling and aliasing.

15. Choice of Iterative Scheme

As a practical matter one has a choice of how to interlace the iterations with the time steps. On the one hand, one could iterate until the solution has stabilized before advancing to the next image frame. On the other hand, given a good initial guess one may need only one iteration per time-step. A good initial guess for the optical flow velocities is usually available from the previous time-step.

The advantages of the latter approach include an ability to deal with more images per unit time and better estimates of optical flow velocities in certain regions. Areas in which the brightness gradient is small lead to uncertain, noisy estimates obtained partly by filling in from the surround. These estimates are improved by considering further images. The noise in measurements of the images will be independent and tend to cancel out. Perhaps more importantly, different parts of the pattern will drift by a given point in the image. The direction of the brightness gradient will vary with time, providing information about both components of the optical flow velocity.

A practical implementation would most likely employ one iteration per time step for these reasons. We illustrate both approaches in the experiments.

16. Experiments

The iterative scheme has been implemented and applied to image sequences corresponding to a number of simple flow patterns. The results shown here are for a relatively low resolution image of 32 by 32 picture cells. The brightness measurements were intentionally corrupted by approximately 1% noise and then quantized into 256 levels to simulate a real imaging situation. The underlying surface reflectance pattern was a linear combination of spatially orthogonal sinusoids. Their wavelength was chosen to give reasonably strong brightness gradients without leading to undersampling problems. Discontinuities were avoided to ensure that the required derivatives exist everywhere.

Shown in Fig. 5, for example, are four frames of a sequence of images depicting a sphere rotating about an axis inclined towards the viewer. A smoothly varying reflectance pattern is painted on the surface of the sphere. The sphere is illuminated uniformly from all directions so that there is no shading. We chose to work with synthetic image sequences so that we can compare the results of the optical flow computation with the exact values calculated using the transformation equations relating image coordinates to coordinates on the underlying surface reflectance pattern.

17. Results

The first flow to be investigated was a simple linear translation of the entire brightness pattern. The resulting computed flow is shown as a needle diagram in Fig. 6 for 1, 4, 16, and 64 iterations. The estimated flow velocities are depicted as short lines, showing the apparent displacement during one time step. In this example a single time step was taken so that the computations are based on just two images. Initially the estimates of flow velocity are zero. Consequently the first iteration shows vectors in the direction of the brightness gradient. Later, the estimates approach the correct values in all parts of the image. Few changes occur after 32 iterations when the velocity vectors have errors of about 10%. The estimates tend to be too small, rather than too large, perhaps because of a tendency to underestimate the derivatives. The worst errors occur, as one might expect, where the brightness gradient is small.

In the second experiment one iteration was used per time step on the same

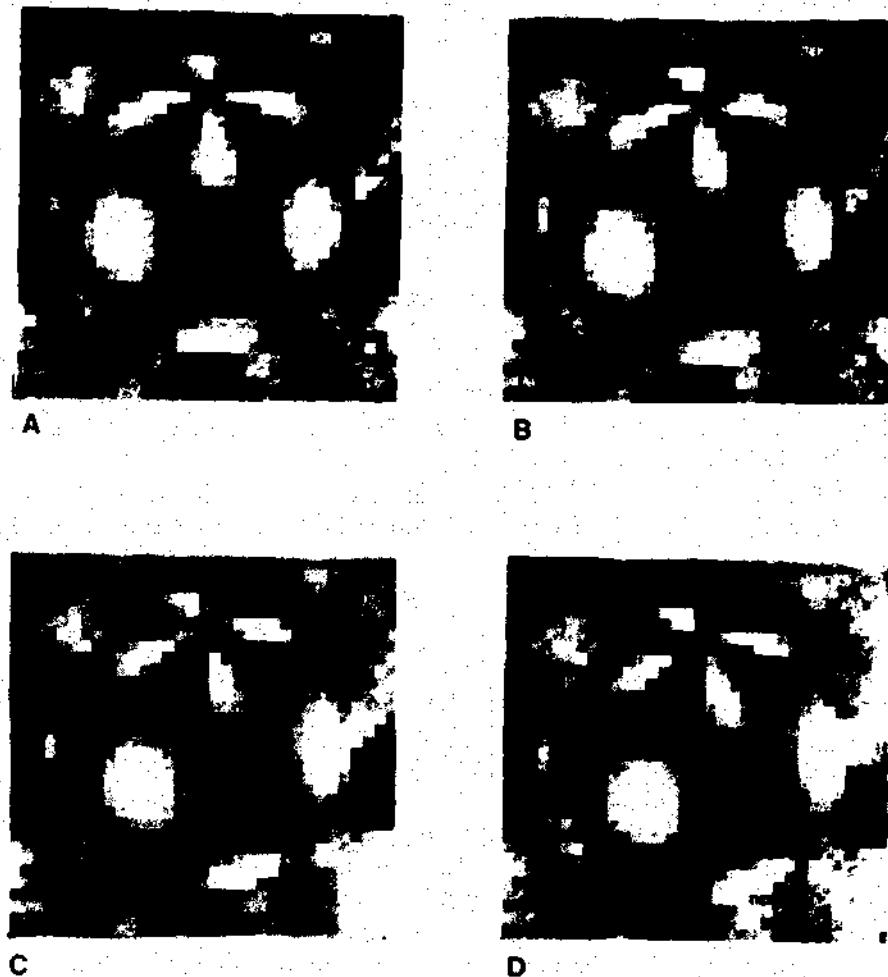


FIG. 5. Four frames out of a sequence of images of a sphere rotating about an axis inclined towards the viewer. The sphere is covered with a pattern which varies smoothly from place to place. The sphere is portrayed against a fixed, lightly textured background. Image sequences like these are processed by the optical flow algorithm.

linear translation image sequence. The resulting computed flow is shown in Fig. 7 for 1, 4, 16, and 64 time steps. The estimates approach the correct values more rapidly and do not have a tendency to be too small, as in the previous experiment. Few changes occur after 16 iterations when the velocity vectors have errors of about 7%. The worst errors occur, as one might expect, where the noise in recent measurements of brightness was worst. While individual estimates of velocity may not be very accurate, the average over the whole image was within 1% of the correct value.

Next, the method was applied to simple rotation and simple contraction of the brightness pattern. The results after 32 time steps are shown in Fig. 8. Note that the magnitude of the velocity is proportional to the distance from the origin of the flow in both of these cases. (By origin we mean the point in the image where the velocity is zero.)

In the examples so far the Laplacian of both flow velocity components is zero everywhere. We also studied more difficult cases where this was not the case.

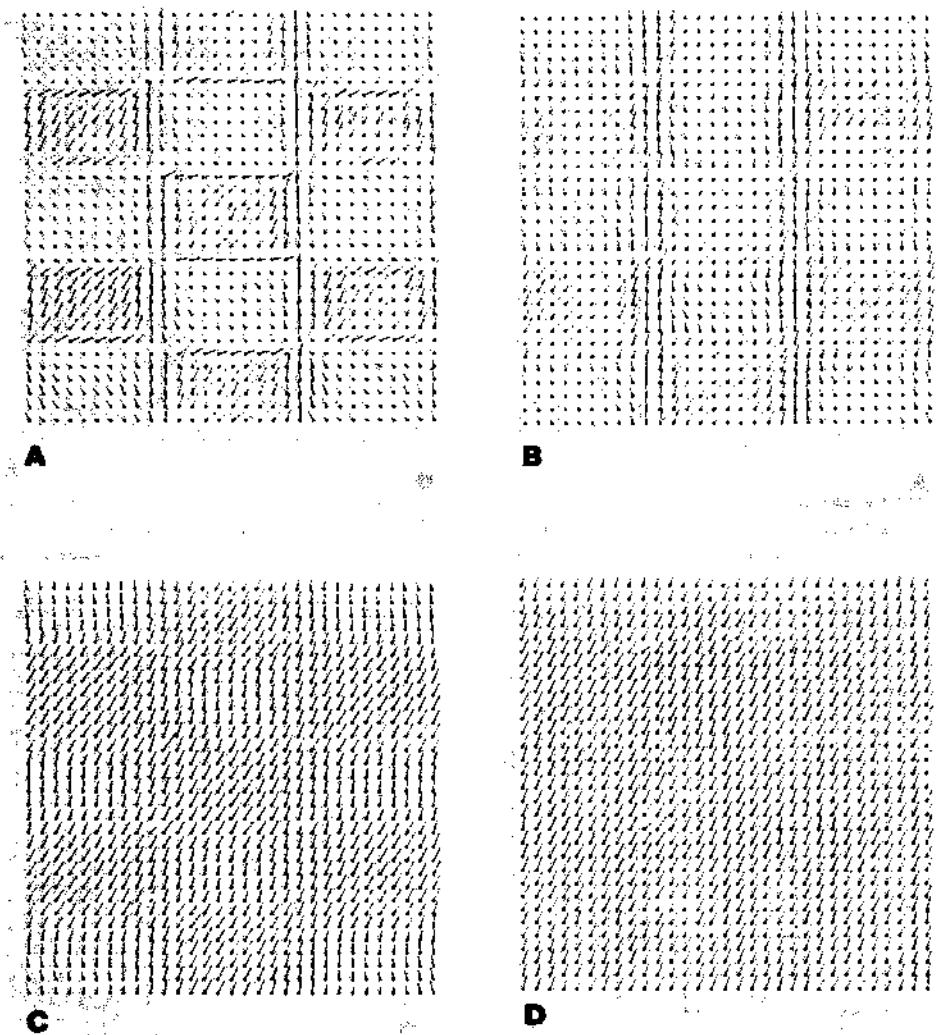


FIG. 6. Flow pattern computed for simple translation of a brightness pattern. The estimates after 1, 4, 16, and 64 iterations are shown. The velocity is 0.5 picture cells in the x direction and 1.0 picture cells in the y direction per time interval. Two images are used as input, depicting the situation at two times separated by one time interval.

In particular, if we let the magnitude of the velocity vary as the inverse of the distance from the origin we generate flow around a line vertex and two dimensional flow into a sink. The computed flow patterns are shown in Fig. 9. In these examples, the computation involved many iterations based on a single time step. The worst errors occur near the singularity at the origin of the flow pattern, where velocities are found which are much larger than one picture cell per time step.

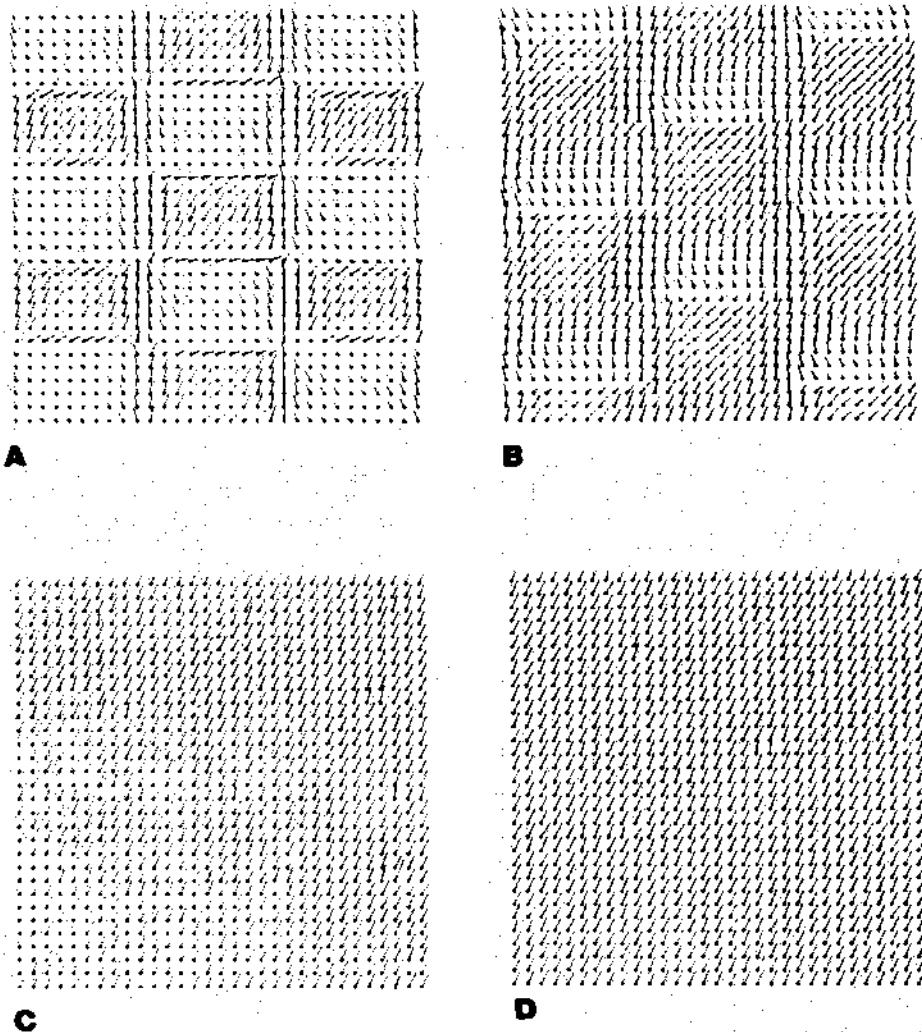


FIG. 7. Flow pattern computed for simple translation of a brightness pattern. The estimates after 1, 4, 16, and 64 time steps are shown. Here one iteration is used per time step. Convergence is more rapid and the velocities are estimated more accurately.

Finally we considered rigid body motions. Shown in Fig. 10 are the flows computed for a cylinder rotating about its axis and for a rotating sphere. In both cases the Laplacian of the flow is not zero and in fact the Laplacian for one of the velocity components becomes infinite on the occluding bound. Since the velocities themselves remain finite, reasonable solutions are still obtained. The correct flow patterns are shown in Fig. 11. Comparing the computed and exact values shows that the worst errors occur on the occluding boundary. These

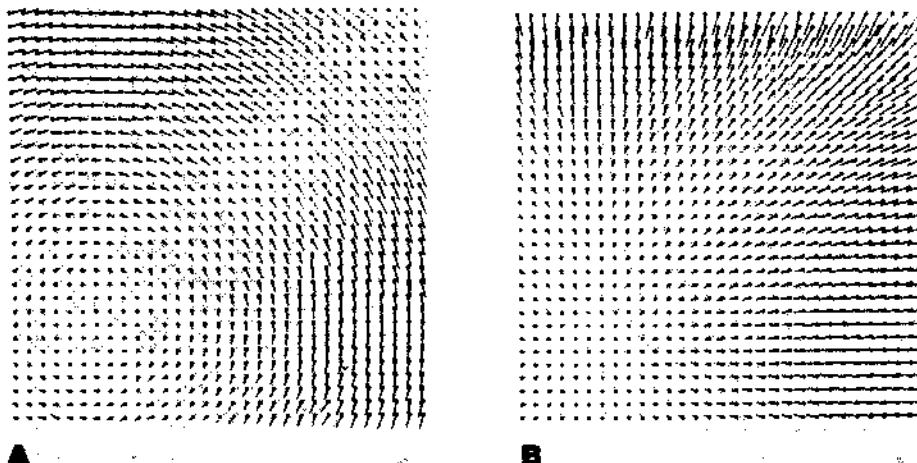


FIG. 8. Flow patterns computed for simple rotation and simple contraction of a brightness pattern. In the first case, the pattern is rotated about 2.8 degrees per time step, while it is contracted about 5% per time step in the second case. The estimates after 32 time steps are shown.

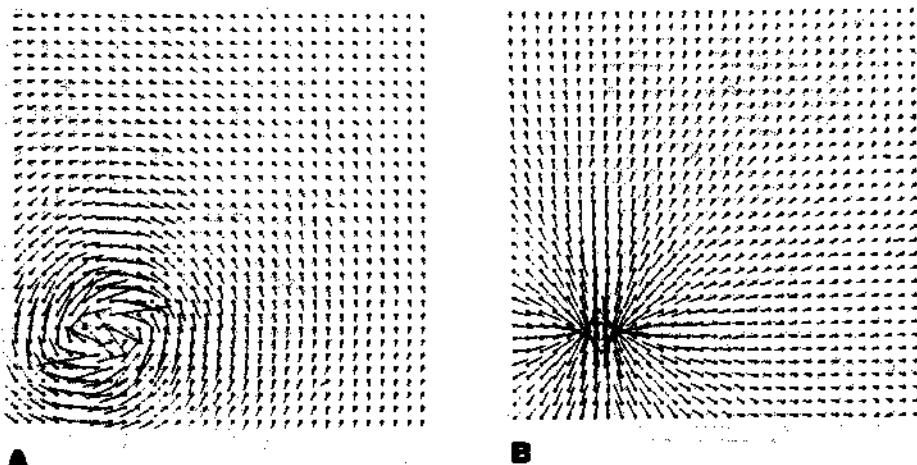


FIG. 9. Flow patterns computed for flow around a line vortex and two dimensional flow into a sink. In each case the estimates after 32 iterations are shown.

boundaries constitute a one dimensional subset of the plane and so one can expect that the relative number of points at which the estimated flow is seriously in error will decrease as the resolution of the image is made finer.

In Appendix B it is shown that there is a direct relationship between the Laplacian of the flow velocity components and the Laplacian of the surface height. This can be used to see how our smoothness constraint will fare for different objects. For example, a rotating polyhedron will give rise to flow

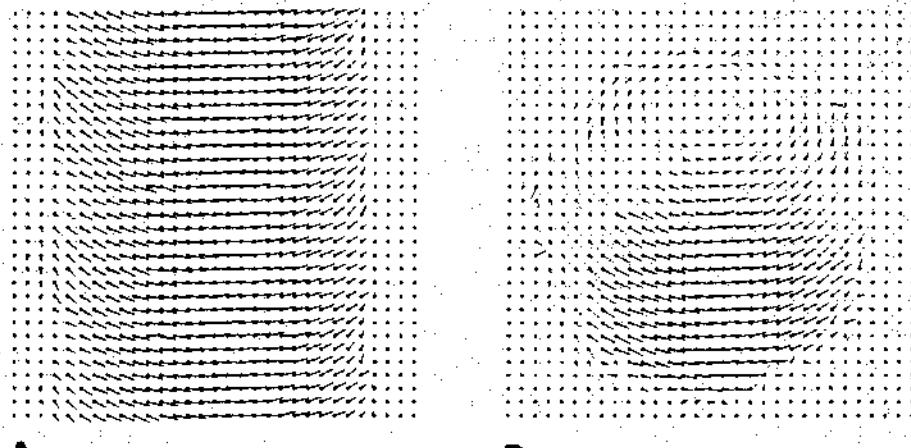


FIG. 10. Flow patterns computed for a cylinder rotating about its axis and for a rotating sphere. The axis of the cylinder is inclined 30 degrees towards the viewer and that of the sphere 45 degrees. Both are rotating at about 5 degrees per time step. The estimates shown are obtained after 32 time steps.

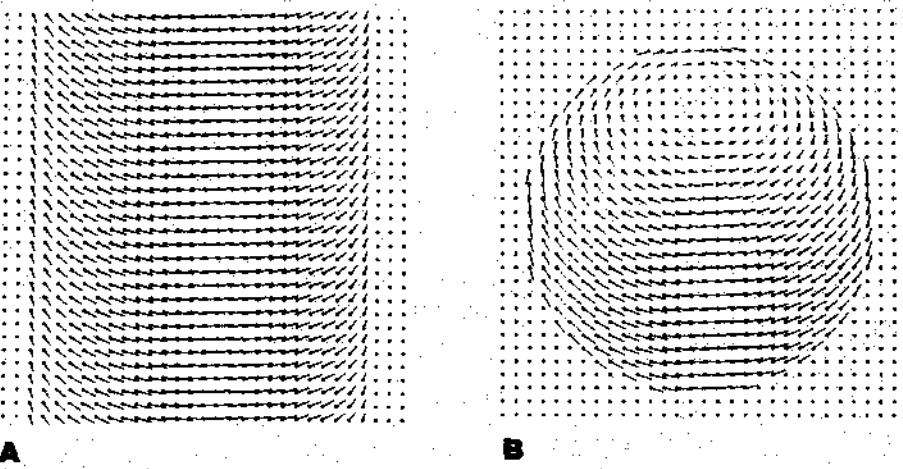


FIG. 11. Exact flow patterns for the cylinder and the sphere.

which has zero Laplacian except on the image lines which are the projections of the edges of the body.

18. Summary

A method has been developed for computing optical flow from a sequence of images. It is based on the observation that the flow velocity has two components and that the basic equation for the rate of change of image brightness

provides only one constraint. Smoothness of the flow was introduced as a second constraint. An iterative method for solving the resulting equation was then developed. A simple implementation provided visual confirmation of convergence of the solution in the form of needle diagrams. Examples of several different types of optical flow patterns were studied. These included cases where the Laplacian of the flow was zero as well as cases where it became infinite at singular points or along bounding curves.

The computed optical flow is somewhat inaccurate since it is based on noisy, quantized measurements. Proposed methods for obtaining information about the shapes of objects using derivatives (divergence and curl) of the optical flow field may turn out to be impractical since the inaccuracies will be amplified.

ACKNOWLEDGMENT

This research was conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-75-C0643. One of the authors (Horn) would like to thank Professor H.-H. Nagel for his hospitality. The basic equations were conceived during a visit to the University of Hamburg, stimulated by Professor Nagel's long-standing interest in motion vision. The other author (Schunck) would like to thank W.E.L. Grimson and E. Hildreth for many interesting discussions and much knowledgeable criticism. W.E.L. Grimson and Katsushi Ikeuchi helped to illuminate a conceptual bug in an earlier version of this paper. We should also like to thank J. Jones for preparing the drawings.

Appendix A. Rate of Change of Image Brightness

Consider a patch of the brightness pattern that is displaced a distance δx in the x -direction and δy in the y -direction in time δt . The brightness of the patch is assumed to remain constant so that

$$E(x, y, t) = E(x + \delta x, y + \delta y, t + \delta t).$$

Expanding the right-hand side about the point (x, y, t) we get,

$$E(x, y, t) = E(x, y, t) + \delta x \frac{\partial E}{\partial x} + \delta y \frac{\partial E}{\partial y} + \delta t \frac{\partial E}{\partial t} + \epsilon.$$

Where ϵ contains second and higher order terms in δx , δy , and δt . After subtracting $E(x, y, t)$ from both sides and dividing through by δt we have

$$\frac{\delta x}{\delta t} \frac{\partial E}{\partial x} + \frac{\delta y}{\delta t} \frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} + O(\delta t) = 0,$$

where $O(\delta t)$ is a term of order δt (we assume that δx and δy vary as δt). In the limit as $\delta t \rightarrow 0$ this becomes

$$\frac{\partial E}{\partial x} \frac{dx}{dt} + \frac{\partial E}{\partial y} \frac{dy}{dt} + \frac{\partial E}{\partial t} = 0.$$

Appendix B. Smoothness of Flow for Rigid Body Motions

Let a rigid body rotate about an axis $(\omega_x, \omega_y, \omega_z)$, where the magnitude of the vector equals the angular velocity of the motion. If this axis passes through the origin, then the velocity of a point (x, y, z) equals the cross product of $(\omega_x, \omega_y, \omega_z)$, and (x, y, z) . There is a direct relationship between the image coordinates and the x and y coordinates here if we assume that the image is generated by orthographic projection. The x and y components of the velocity can be written,

$$u = \omega_y z - \omega_z y, \quad v = \omega_x z - \omega_z x.$$

Consequently,

$$\nabla^2 u = +\omega_y \nabla^2 z, \quad \nabla^2 v = -\omega_x \nabla^2 z.$$

This illustrates that the smoothness of the optical flow is related directly to the smoothness of the rotating body and that the Laplacian of the flow velocity will become infinite on the occluding bound, since the partial derivatives of z with respect to x and y become infinite there.

REFERENCES

1. Ames, W.F., *Numerical Methods for Partial Differential Equations* (Academic Press, New York, 1977).
2. Batali, J. and Ullman, S., Motion detection and analysis, *Proc. of the ARPA Image Understanding Workshop*, 7-8 November 1979 (Science Applications Inc., Arlington, VA 1979) pp. 69-75.
3. Clocksin, W., Determining the orientation of surfaces from optical flow, *Proc. of the Third AISB Conference*, Hamburg (1978) pp. 93-102.
4. Conte, S.D. and de Boor, C., *Elementary Numerical Analysis* (McGraw-Hill, New York, 1965, 1972).
5. Fennema, C.L. and Thompson, W.B., Velocity determination in scenes containing several moving objects, *Computer Graphics and Image Processing* 9 (4) (1979) 301-315.
6. Gibson, J.J., *The Perception of the Visual World* (Riverside Press, Cambridge, 1950).
7. Gibson, J.J., *The Senses Considered as Perceptual Systems* (Houghton-Mifflin, Boston, MA, 1966).
8. Gibson, J.J., On the analysis of change in the optic array, *Scandinavian J. Psychol.* 18 (1977) 161-163.
9. Gray, S.B., Local properties of binary images in two dimensions, *IEEE Trans. on Computers* 20 (5) (1971) 551-561.
10. Hadani, I., Ishai, G. and Gur, M., Visual stability and space perception in monocular vision: Mathematical model, *J. Optical Soc. Am.* 70 (1) (1980) 60-65.
11. Hamming, R.W., *Numerical Methods for Scientists and Engineers* (McGraw-Hill, New York, 1962).
12. Hildebrand, F.B., *Methods of Applied Mathematics* (Prentice-Hall, Englewood Cliffs, NJ, 1952, 1965).
13. Hildebrand, F.B., *Introduction to Numerical Analysis* (McGraw-Hill, New York, 1956, 1974).
14. Horn, B.K.P., (1979) Hill shading and the reflectance map, *Proc. IEEE* 69 (1) (1981) 14-47.
15. Jain, R., Martin, W.N. and Aggarwal, J.K., Segmentation through the detection of changes due to motion, *Computer Graphics and Image Processing* 11 (1) (1979) 13-34.
16. Jain, R., Militzer, D. and Nagel, H.-H., Separating non-stationary from stationary scene components in a sequence of real world TV-images, *Proc. of the 5th Int. Joint Conf. on Artificial Intelligence*, August 1977, Cambridge, MA, 612-618.
17. Jain, R. and Nagel, H.-H., On the analysis of accumulative difference pictures from image sequences of real world scenes, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 1 (2) (1979) 206-214.
18. Koenderink, J.J. and van Doorn, A.J., Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer, *Optica Acta* 22 (9) 773-791.
19. Koenderink, J.J. and van Doorn, A.J., Visual perception of rigidity of solid shape, *J. Math. Biol.* 3 (79) (1976) 79-85.
20. Limb, J.O. and Murphy, J.A., Estimating the velocity of moving images in television signals, *Computer Graphics and Image Processing* 4 (4) (1975) 311-327.
21. Longuet-Higgins, H.C. and Prazdny, K., The interpretation of moving retinal image, *Proc. of the Royal Soc. B* 208 (1980) 385-387.

DETERMINING OPTICAL FLOW

22. Marr, D. and Ullman, S., Directional selectivity and its use in early visual processing, *Artificial Intelligence Laboratory Memo No. 524*, Massachusetts Institute of Technology (June 1979), to appear in *Proc. Roy. Soc. B*.
23. Mersereau, R.M., The processing of hexagonally sampled two-dimensional signals, *Proc. of the IEEE* **67** (6) (1979) 930-949.
24. Milne, W.E., *Numerical Solution of Differential Equations* (Dover, New York, 1953, 1979).
25. Nagel, H.-H., Analyzing sequences of TV-frames, *Proc. of the 5th Int. Joint Conf. on Artificial Intelligence*, August 1977, Cambridge, MA, 626.
26. Nagel, H.-H., Analysis techniques for image sequences, *Proc. of the 4th Int. Joint Conf. on Pattern Recognition*, 4-10 November 1978, Kyoto, Japan.
27. Nakayama, K. and Loomis, J.M., Optical velocity patterns, velocity-sensitive neurons and space perception, *Perception* **3** (1974) 63-80.
28. Netravali, A.N. and Robbins, J.D., Motion-compensated television coding: Part I, *The Bell System Tech. J.* **58** (3) (1979) 631-670.
29. Prazdny, K., Computing egomotion and surface slant from optical flow, Ph.D. Thesis, Computer Science Department, University of Essex, Colchester (1979).
30. Prazdny, K., Egomotion and relative depth map from optical flow, *Biol. Cybernet.* **36** (1980) 87-102.
31. Prazdny, K., The information in optical flows, Computer Science Department, University of Essex, Colchester (1980) mimeographed.
32. Richtmyer, R.D. and Morton, K.W., *Difference Methods for Initial-Value Problems* (Interscience, John Wiley & Sons, New York, 1957, 1967).
33. Russell, D.L., *Calculus of Variations and Control Theory* (Academic Press, New York, 1976).
34. Young, W. and Mandelstam, S., *Variational Principles in Dynamics and Quantum Theory* (Dover, New York, 1968, 1979).

Received March 1980

An Investigation of Smoothness Constraints for the Estimation of Displacement Vector Fields from Image Sequences

HANS-HELLMUT NAGEL, MEMBER, IEEE, AND WILFRIED ENKELMANN

Abstract—A mapping between one frame from an image sequence and the preceding or following frame can be represented as a displacement vector field. In most situations, the mere gray value variations do not provide sufficient information in order to estimate such a displacement vector field. Supplementary constraints are necessary, for example the postulate that a displacement vector field varies smoothly as a function of the image position. Taken as a general requirement, this creates difficulties at gray value transitions which correspond to occluding contours. Nagel therefore introduced the “oriented smoothness” requirement which restricts variations of the displacement vector field only in directions with small or no variation of gray values. This contribution reports results of an investigation about how such an “oriented smoothness” constraint may be formulated and evaluated.

Index Terms—Displacement vector fields, image registration, image sequences, optical flow, smoothness constraints, stereo matching.

I. INTRODUCTION

THE planar displacement vector field $\mathbf{u}(\mathbf{x}) = (u(x), v(x))^T$ links the pixel at image location $\mathbf{x} = (x, y)^T$ in one frame to the corresponding pixel position in another frame. The reliable estimation of displacement vector fields is of great importance for the extraction of spatial information from stereo-pairs as well as for the interpretation of image sequences about scenes with temporal variations—see, e.g., [18].

In most situations the mere gray value variations do not provide sufficient information to completely determine $\mathbf{u}(\mathbf{x})$. Horn and Schunck [11], therefore, postulated that displacement vector fields should vary smoothly as a function of the image coordinate vector \mathbf{x} . This postulate enabled them to estimate both components u and v of $\mathbf{u}(\mathbf{x})$. Such a general smoothness requirement, however, forces the estimated displacement vector field to vary smoothly even across the image of occluding edges. Several approaches have been investigated in order to cope with this difficulty.

Manuscript received October 11, 1984; revised February 11, 1986. Recommended for acceptance by S. L. Tanimoto. This work was supported in part by the Deutsche Forschungsgemeinschaft.

H.-H. Nagel is with the Fraunhofer-Institut für Informations- und Datenverarbeitung, Sebastian-Kneipp-Str. 12-14, 7500 Karlsruhe 1, West Germany, and the Fakultät für Informatik, Universität Karlsruhe, West Germany.

W. Enkelmann was with the Fachbereich Informatik, Universität Hamburg, Schlüterstr. 70, 2000 Hamburg 13, West Germany. He is now with the Fraunhofer-Institut für Informations- und Datenverarbeitung, Sebastian-Kneipp-Str. 12-14, 7500 Karlsruhe 1, West Germany.

IEEE Log Number 8609321.

Yachida [25] assumed that a displacement vector can be estimated at prominent points. He propagated such estimates into neighboring areas with large gray value gradients, based on the method of Horn and Schunck [11]. His iterative improvement scheme employed the inverse variance of displacement estimates from a 5×5 window as a weight in order to suppress the propagation of displacement estimates with large local variations.

Cornelius and Kanade [2] deactivated the smoothness requirement in the neighborhood of zero-crossing contours in order to avoid that estimates spill across potential discontinuities of the displacement vector field.

Wu *et al.* [24] (see also [3]) propagated a displacement estimate only along a contour line between corner points. At each new contour point, they combined the estimated displacement vector from the previous contour point with new estimates of the contour direction and of the displacement component perpendicular to the contour in order to update the tangential component of the displacement vector.

Hildreth [8] minimized the sum of two terms, integrated along a zero-crossing contour. The first term is the squared difference between the “measured” and the estimated displacement component perpendicular to the contour. The second term represents the squared derivative of the displacement vector field with respect to the arclength along the zero-crossing contour, expressing the smoothness requirement (see also [7], [9], [10]).

Nagel [16], [17] suggested an approach which does not require the explicit determination of gray value transition fronts such as edge lines or zero-crossing contours. Whereas Yachida [25] employed some indirect evidence—namely the inverse variance of the estimated displacement vector field—as a weight, the approach of Nagel relied upon the gray value variation directly in order to constrain the variation of the displacement vector field.

The idea to let the gray value variations themselves influence the smoothness constraints originally occurred to Nagel [15] in a slightly different context. Since then, a continuous interaction between theoretical and experimental investigations has resulted in a gradual development of how this idea might be formulated mathematically and how such a formulation might be evaluated.

The introduction of a supplementary constraint evidently contains some arbitrariness. It appears useful, there-

fore, to delineate the gradual development of a constraint formulation; potential alternatives are exposed as well as their advantages and disadvantages. In this manner, it is hoped to contribute to a better understanding of the entire problem area.

The next section quickly recapitulates the development up to the formulation presented in [17]. A first solution approach for the resulting system of partial differential equations is outlined in Section III. Difficulties encountered during the implementation of this solution approach resulted in a series of modifications to both the solution approaches as well as the mathematical formulation of the "oriented smoothness" constraint. These steps are discussed in subsequent sections. The final section presents encouraging results from the currently pursued constraint formulation applied to examples taken from a real-world image sequence.

II. INITIAL STEPS TOWARDS AN "ORIENTED SMOOTHNESS" CONSTRAINT

Based on an analysis of experimental attempts to characterize "corners" by Kitchen and Rosenfeld [12], [13] as well as independent ones by Dreschler and Nagel [4], [5], Nagel [15] found both approaches to be compatible with the following definition:

Identify a "gray value corner" with the position of maximum planar curvature in the locus line of steepest gray value slope.

Dreschler and Nagel [6] compared the locations obtained from this definition with locations of maximum planar curvature in zero-crossing contours in the Laplacian applied to real world images with and without Gaussian smoothing. More recently, Zuniga and Haralick [26] compared their results for slight variations of the approach as defined above to those of the original attempts by Kitchen and Rosenfeld [12], [13] and those by Dreschler and Nagel [4], [5]. All these results support the hypothesis that the definition given above captures essential aspects of visually prominent corners in gray value images.

Since the position of a gray value corner is well defined, it should be possible to estimate both components of its displacement vector. This could indeed be shown by Nagel [15] based on the following approach.

The gray value $g(x)$ is taken to be a twice continuously differentiable function of the image position vector x . A local coordinate system can be chosen such that a gray value corner is characterized by

$$g_x = \frac{\partial g}{\partial x} = \text{maximum} \neq 0$$

$$g_{xx} = \frac{\partial^2 g}{\partial x^2} = 0 \quad (1a)$$

$$g_y = \frac{\partial g}{\partial y} = 0$$

$$g_{yy} = \frac{\partial^2 g}{\partial y^2} = \text{maximum} \neq 0 \quad (1b)$$

The observed gray values in an image window are approximated by a bivariate polynomial of second order. Nagel [15] derived a closed form solution for both components u and v of the displacement vector u at the center position of a symmetric window provided the approximating polynomial conforms to the requirements of (1).

A. Iterative Refinement for Corner Displacement Estimates

Since the conditions (1) will be rarely satisfied exactly at a pixel position, Nagel [15] developed an iterative approach to refine the displacement estimate whenever (1) would be satisfied only approximately. This iterative approach provided the starting point for the development to be presented in this contribution. It will be outlined, therefore, to serve as a reference for later discussions.

Assume that all pixels within the image window under consideration have been provided with a serial index i . All pixels within this window are assumed to be displaced by the unknown displacement vector u for which an estimate u_0 may be available such that

$$u = u_0 + Du \quad (2)$$

Let $g1(x) = g(x, t_1)$ represent the gray value at location x in frame 1 and analogously $g2(x) = g(x, t_2)$ for frame 2. The displacement correction Du and thereby u itself should be estimated by minimizing a weighted sum of squared gray value differences within the image window. These differences are denoted by

$$dg = g2(x) - g1(x - u) \quad (3a)$$

$$= g2(x) - g1(x - u_0 - Du) \quad (3b)$$

where the column vector dg has the components

$$dg_i = g2(x_i) - g1(x_i - u_0 - Du) \quad (3c)$$

for $i = 1, \dots, N$ with N indicating the number of pixels in the window.

The expression to be minimized is given as

$$(dg)^T W_G (dg) \rightarrow \text{minimum.} \quad (4)$$

The symmetric weight matrix W_G depends on the variance σ^2 of the gray value measurements and on the window size, but does not depend on the gray values themselves. W_G has been given in [15]. The dependence of dg and thus the expression (4) on $Du = (Du_x, Du_y)^T$ may be made explicit by substituting a bivariate approximation polynomial for $g1(x - u_0 - Du)$. Since Du is assumed to be small, higher than linear terms in the components of Du may be neglected.

$$\begin{aligned} g1(x - u_0 - Du) \\ \approx g1(x - u_0) - (\nabla g1)^T Du + u_0^T (\nabla \nabla g1) Du \end{aligned} \quad (5)$$

where the following notational abbreviations have been used

$$\nabla g1 = \begin{pmatrix} g1_x(x) \\ g1_y(x) \end{pmatrix} \quad (6a)$$

$$\nabla \nabla g_1 = \begin{pmatrix} g_{1,xx}(x) & g_{1,xy}(x) \\ g_{1,xy}(x) & g_{1,yy}(x) \end{pmatrix}. \quad (6b)$$

The first and second partial derivatives in (6) are taken to be the coefficients of the second-order bivariate polynomial which is obtained by a least squares approximation to $g_1(x)$ within the window under consideration:

$$\sum_{\text{window}} [g_1(x) - g_1 - (\nabla g_1)^T x - \frac{1}{2} x^T (\nabla \nabla g_1) x]^2 \Rightarrow \text{minimum.} \quad (7)$$

The vector $\nabla g_1 - u_0^T (\nabla \nabla g_1)$ which enters into the scalar product with Du in (5) is nothing but the approximation to the gradient of $g_1(x)$ at location $x = u_0$.

If the matrix of partial derivatives of dg with respect to the components Du and Dv of Du is denoted by B^T , i.e.

$$B^T = \frac{\partial dg}{\partial Du} \quad (8)$$

we may write (4) in the form

$$(g_2(x) - g_1(x - u_0) + BDu)^T W_G \cdot (g_2(x) - g_1(x - u_0) + BDu) \quad (4a)$$

from which we obtain—provided $\det(B^T W_G B) \neq 0$ —

$$Du = -(B^T W_G B)^{-1} B^T W_G (g_2(x) - g_1(x - u_0)). \quad (9)$$

Nagel [15] has shown that this solution for the correction vector Du yields immediately a closed form solution for gray value corners, i.e., if the coefficients of the approximation polynomial satisfy (1). An implementation of this iterative improvement gave results supporting our expectations [20].

Although the approach based on (9) should improve the displacement estimate in the immediate vicinity of a gray value corner, it appeared attractive to employ it for the propagation of this displacement estimate into the surrounding environment. This was expected to work as long as $\det(B^T W_G B)$ would remain significantly different from zero. An experimental investigation in this direction encountered convergence problems even if the determinant of $B^T W_G B$ yielded values exceeding 10^4 .

Upon closer inspection, the cause turned out to be a gross imbalance between the two eigenvalues of $B^T W_G B$. Using the abbreviation

$$h_{1,x} = g_{1,x} - g_{1,xx}u_0 - g_{1,xy}v_0 \quad (10a)$$

$$h_{1,y} = g_{1,y} - g_{1,xy}u_0 - g_{1,yy}v_0 \quad (10b)$$

and assuming a square window, i.e.,

$$\bar{x}^2 = \bar{y}^2$$

the 2×2 matrix $B^T W_G B$ may be written

$$B^T W_G B = \frac{N}{2\sigma^2} \begin{pmatrix} h_{1,x}^2 + \bar{x}^2(g_{1,xx}^2 + g_{1,xy}^2) & h_{1,x}h_{1,y} + \bar{x}^2g_{1,xy}(g_{1,xx} + g_{1,yy}) \\ h_{1,x}h_{1,y} + \bar{x}^2g_{1,xy}(g_{1,xx} + g_{1,yy}) & h_{1,y}^2 + \bar{x}^2(g_{1,xy}^2 + g_{1,yy}^2) \end{pmatrix}. \quad (11a)$$

This matrix contains remarkable information about the local gray value variation within the window—see [19]. This becomes even more obvious if the local coordinate system is aligned with the principal curvature directions of $g(x)$ at the window center: in this case the mixed second derivative g_{xy} vanishes, yielding

$$B^T W_G B = \frac{N}{2\sigma^2} \begin{pmatrix} h_{1,x}^2 + \bar{x}^2g_{1,xx}^2 & h_{1,x}h_{1,y} \\ h_{1,x}h_{1,y} & h_{1,y}^2 + \bar{x}^2g_{1,yy}^2 \end{pmatrix}. \quad (11b)$$

If both second partial derivatives vanish, i.e., if the gray value function $g(x)$ is locally planar such as along a straight line zero-crossing section, $\det(B^T W_G B)$ becomes zero. It does not vanish, however, if only one principal curvature becomes zero, for example in the situation specified by (1) with $u_0 = 0$. In this case one obtains

$$B^T W_G B = \frac{N}{2\sigma^2} \begin{pmatrix} g_{1,x}^2 & 0 \\ 0 & \bar{x}^2g_{1,yy}^2 \end{pmatrix}. \quad (11c)$$

This demonstrates that both the gray value slope and the change in the direction of the gray value slope explicitly influence this matrix.

For the following discussion it is assumed that the local coordinate system is aligned with the eigenvectors of $B^T W_G B$. Then we may write

$$B^T W_G B = \begin{pmatrix} \kappa & 0 \\ 0 & \lambda \end{pmatrix}. \quad (12)$$

If one of the eigenvalues, say λ , is much smaller than the other, the determinant given by

$$\det(B^T W_G B) = \kappa\lambda \quad (13)$$

may still yield a sizable numerical value. Such a situation, however, implies that both the gradient and the curvature along the direction corresponding to the eigenvector associated with λ are small. Locally, the gray value variation $g(x)$ can be approximated by a cylinder with an axis parallel to the eigenvector direction associated with λ . In this case, the gray value variation does not provide enough information to reliably estimate the displacement vector component along this direction.

Since (13) as well as the following one

$$\text{trace}(B^T W_G B) = \kappa + \lambda \quad (14)$$

are invariant against rotation of the x -coordinate system, one can write

$$\frac{\det(B^T W_G B)}{[\frac{1}{2} \text{trace}(B^T W_G B)]^2} = \left(\frac{\sqrt{\kappa\lambda}}{\frac{1}{2}(\kappa + \lambda)} \right)^2. \quad (15)$$

This is the squared ratio of the geometric and arithmetic mean of the eigenvalues. If $\lambda \ll \kappa$, one may write

$$\frac{\det(B^T W_G B)}{[\frac{1}{2} \text{trace}(B^T W_G B)]^2} = 4 \frac{\kappa\lambda}{(\kappa + \lambda)^2} = 4 \frac{\lambda}{\kappa}. \quad (16)$$

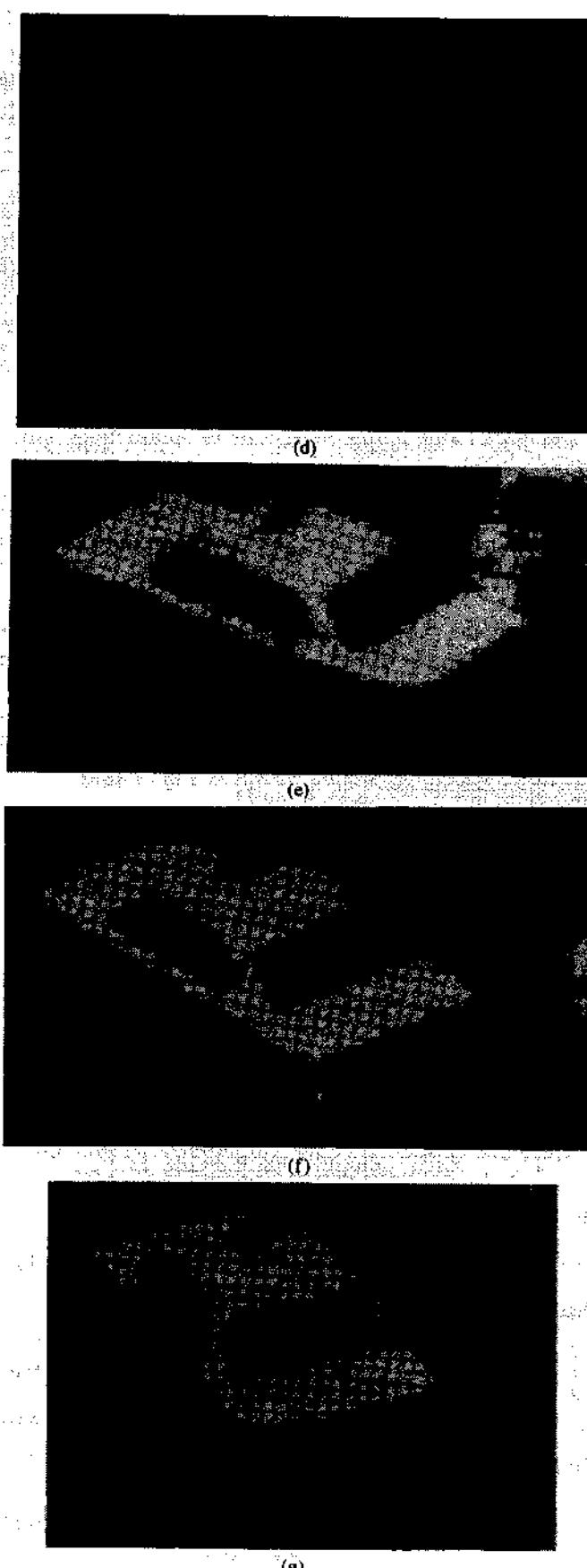
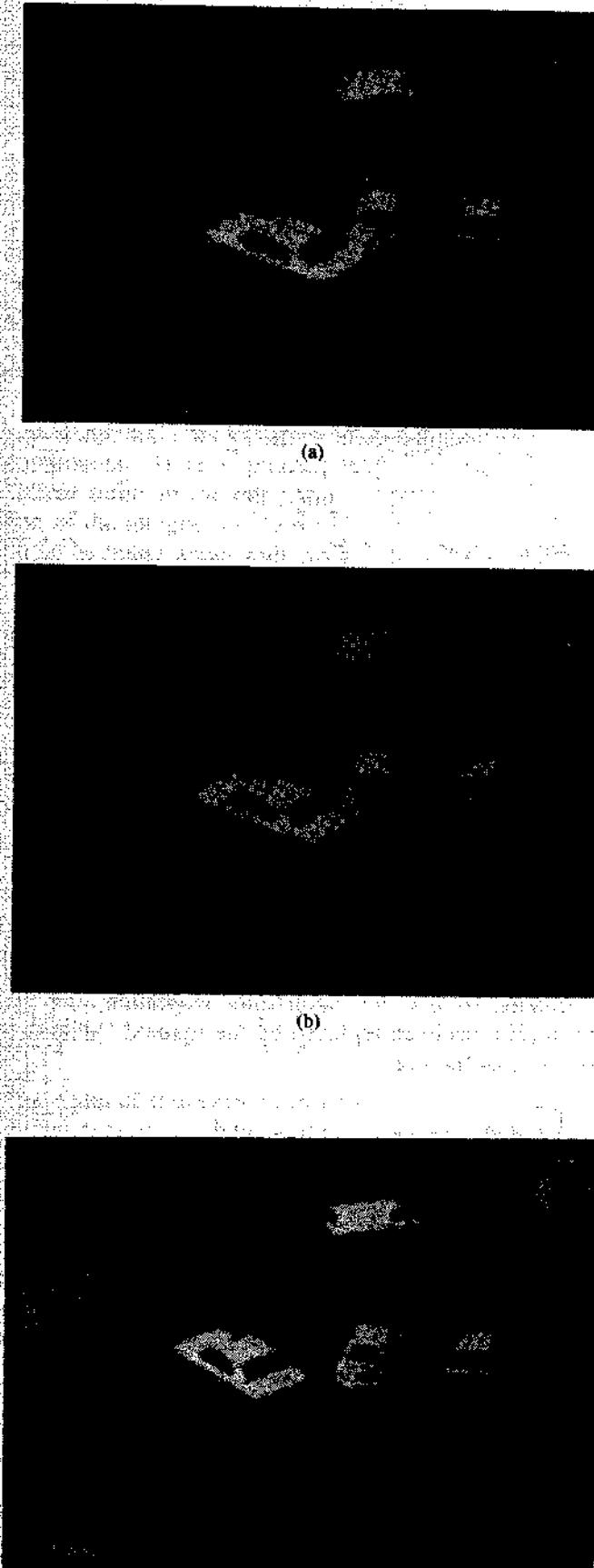


Fig. 1. Several frames from the real-world image sequence used to study the approach described in this contribution: (a) frame 11; (b) frame 12; (c) frame 20; (d) frame 30. A window around the bright taxicab in the center of the image (e) from frame 11, (f) from frame 20, and (g) from frame 30.

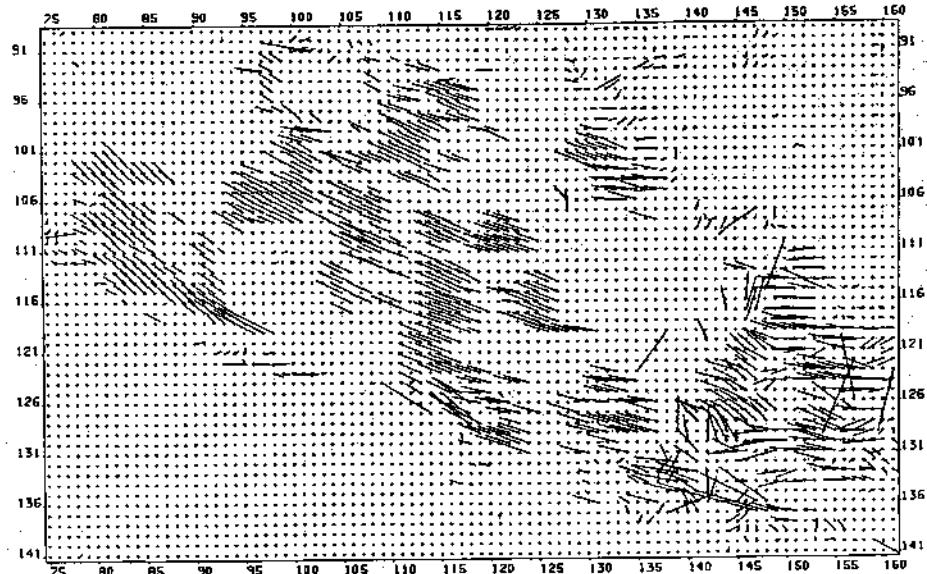


Fig. 2. Displacement vectors estimated according to (9) and (10) in the vicinity of gray value corners.

Comparing the ratio (16) to a threshold is, therefore, a straightforward test to suppress those windows where the gray value variation is insufficient for a reliable estimation of both displacement vector components. An implementation of this approach applied to the image sequence of Fig. 1(e) yielded the results shown in Fig. 2 [21].

B. Heuristic Introduction of "Oriented Smoothness"

The idea to let the gray value variation directly influence the smoothness requirement had already been suggested by Nagel [15]. The investigation described in the preceding section shed new light on how $B^T W_G B$ incorporates essential information about both slope and curvature of the gray value variation. This insight induced Nagel [16] to suggest that $(B^T W_G B)^{-1}$ might be explored as a suitable weight matrix for an "oriented smoothness" constraint.

A general smoothness constraint has been introduced by Horn and Schunck [11] as a minimization problem

$$\iint dxdy \{(\nabla g^T u + g_i)^2 + \alpha^2(u_x^2 + u_y^2 + v_x^2 + v_y^2)\} \Rightarrow \min. \quad (17)$$

This has been rewritten by Nagel in the form

$$\iint dxdy \{(\nabla g^T u + g_i)^2 + \alpha^2 \text{trace}[(\nabla u)^T (\nabla u)]\} \Rightarrow \min. \quad (18)$$

where

$$\nabla u = \begin{pmatrix} u_x & v_x \\ u_y & v_y \end{pmatrix} \quad (19)$$

represents the matrix of partial derivatives of the displacement vector components with respect to the image coordinates. The second term in (18) represents the smoothness requirement introduced by Horn and Schunck [11].

The factor α^2 denotes the strength of the smoothness requirement relative to the first term.

The idea of Nagel [16] has been to introduce the inverse of $B^T W_G B$ as a weight matrix into the smoothness term:

$$\begin{aligned} \iint dxdy \{(\nabla g^T u + g_i)^2 + \alpha^2 \text{trace}[(\nabla u)^T (B^T W_G B)^{-1} (\nabla u)]\} \Rightarrow \min. \end{aligned} \quad (20)$$

$B^T W_G B$ has been shown in [16] to be positive semidefinite. Since the second order terms in the polynomial approximation of $g(x)$ are sometimes important, the first term in (20) has been replaced by the squared "displaced gray value difference":

$$\begin{aligned} \iint dxdy \{(g_2(x) - g_1(x - u))^2 + \alpha^2 \text{trace}[(\nabla u)^T (B^T W_G B)^{-1} (\nabla u)]\} \Rightarrow \min. \end{aligned} \quad (21)$$

The idea behind this approach becomes more evident if it is assumed that the coordinate system happens to be aligned with the eigenvector directions of $B^T W_G B$ at the location under discussion.

Using (12), the integrand may then be written as

$$(g_2(x) - g_1(x - u))^2 + \alpha^2 \text{trace} \left((\nabla u)^T \begin{pmatrix} \kappa & 0 \\ 0 & \lambda \end{pmatrix}^{-1} (\nabla u) \right) \quad (22a)$$

or

$$(g_2(x) - g_1(x - u))^2 + \alpha^2 \left(\frac{u_x^2}{\kappa} + \frac{u_y^2}{\lambda} + \frac{v_x^2}{\kappa} + \frac{v_y^2}{\lambda} \right). \quad (22b)$$

If both κ and λ are large, the gray value variation contains sufficient information to estimate both u and v based on the interframe displaced gray value difference ($g_2(x) - g_1(x - u)$). In this situation, the smoothness contribution to the estimation of both u and v would be suppressed by a factor $1/\kappa$ or $1/\lambda$, respectively.

If the gray value variation $g_1(x)$ corresponds locally to a cylinder parallel to the x, y -plane, one of the eigenvalues, say λ , will be much smaller than the other as discussed in the preceding section. In this case, the smoothness contribution would be suppressed only in the direction with strong gray value variation, i.e., the eigenvector direction associated with κ .

If both eigenvalues are small then the smoothness term would constrain the variation of both displacement vector components: large variations of u or v would contribute sizable terms to the integrand. Therefore, the minimization of the integral would tend to diminish variations of u and v in image areas with more or less constant gray values.

C. Derivation of the "Oriented Smoothness" Constraint

The heuristic modification of (17) appeared plausible enough to search for some rationalization. Nagel [17] could show that a weight matrix with essentially the structure of $B^T W_G B$ could be obtained by the following line of argumentation:

1) The variation of the displacement vector u in the direction perpendicular to the gradient should become as small as possible. This variation is captured by the two-component row vector

$$\begin{pmatrix} g_y \\ -g_x \end{pmatrix}^T \nabla u. \quad (23)$$

The norm of this vector can be written in the form:

$$\begin{aligned} & \left(\begin{pmatrix} g_y \\ -g_x \end{pmatrix}^T \nabla u \right) \left(\begin{pmatrix} g_y \\ -g_x \end{pmatrix}^T \nabla u \right)^T \\ &= \text{trace} \left((\nabla u)^T \begin{pmatrix} g_y \\ -g_x \end{pmatrix} \begin{pmatrix} g_y \\ -g_x \end{pmatrix}^T (\nabla u) \right). \quad (24) \end{aligned}$$

2) The variation of the displacement vector u should be as small as possible in the direction perpendicular to the principal curvature direction associated with a large curvature. The variation of u in the direction perpendicular to the principal curvature orientations can be expressed in a manner not depending on a particular choice of the coordinate system by the matrix

$$\begin{aligned} & \begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix}^T (\nabla u) \\ &= \begin{pmatrix} g_{yy}u_x - g_{xy}u_y & g_{yy}v_x - g_{xy}v_y \\ -g_{xy}u_x + g_{xx}u_y & -g_{xy}v_x + g_{xx}v_y \end{pmatrix}. \quad (25) \end{aligned}$$

The sum of the scalar products for each of the two column vectors with themselves can be written in the form

$$\begin{aligned} & \text{trace} \left\{ \left[\begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix}^T (\nabla u) \right]^T \left[\begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix}^T (\nabla u) \right] \right\} \\ &= \text{trace} \left\{ (\nabla u)^T \left[\begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix} \begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix}^T \right] (\nabla u) \right\}. \quad (26a) \end{aligned}$$

$$= \text{trace} \left\{ (\nabla u)^T \left[\begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix} \begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix}^T \right] (\nabla u) \right\}. \quad (26b)$$

3) The "oriented smoothness" constraint demands that the variation of u in the direction perpendicular to significant gray value variations is minimized. This can be achieved by minimizing a weighted sum of the expressions in (24) and (26).

$$F = \left\{ \left(\begin{pmatrix} g_y \\ -g_x \end{pmatrix}^T \nabla u \right)^2 + b^2 \left(\begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix}^T \begin{pmatrix} g_{yy} & -g_{xy} \\ -g_{xy} & g_{xx} \end{pmatrix}^T (\nabla u) \right)^2 \right\} \quad (27)$$

where the factor b^2 denotes the relative weight of the two contributions. One possible way to express the oriented smoothness constraint consists in minimizing

$$(\nabla u)^T F (\nabla u). \quad (28)$$

4) In order to obtain an expression analogous to (21), Nagel [17] argued that the expression (28) should be normalized. The determinant of F appeared as a suitable scalar norm of this matrix. If (28) is divided by $\det(F)$, one obtains the weight matrix

$$C^{-1} = \frac{F}{\det F} \quad (29)$$

with C given by

$$C = \begin{pmatrix} g_x^2 + b^2(g_{xx}^2 + g_{xy}^2) & g_xg_y + b^2g_{xy}(g_{xx} + g_{yy}) \\ g_xg_y + b^2g_{xy}(g_{xx} + g_{yy}) & g_y^2 + b^2(g_{xy}^2 + g_{yy}^2) \end{pmatrix} \quad (30a)$$

$$= ((\nabla g)(\nabla g)^T + b^2(\nabla \nabla g)(\nabla \nabla g)^T) \quad (30b)$$

Comparison of (30a) to equation (11a) shows the structural analogy between C and $B^T W_G B$.

III. FIRST ATTEMPTS AT A SOLUTION

The estimation of a displacement vector field describing the displacement between gray value images $g_1(x)$ and $g_2(x)$ requires the minimization of the following integral—see (21):

$$\begin{aligned} & \iint dxdy \{(g_2(x) - g_1(x - u))^2 \\ &+ \alpha^2 \text{trace} ((\nabla u)^T C^{-1} (\nabla u))\} \rightarrow \min. \quad (31) \end{aligned}$$

As has been pointed out in [17], this integral can be specialized to the one introduced by Horn and Schunck [11] if the following simplifications are applied:

1) The influence of the smoothness constraint related to principal curvature directions is suppressed, i.e., the factor b^2 in (30) is set to zero.

2) The directional sensitivity of the smoothness requirement is suppressed, i.e., the outer product matrix $(\nabla g)(\nabla g)^T$ in (30) is replaced by the unit matrix.

3) The "displaced interframe gray value difference" between the outer pair of parentheses of (31) is replaced by a first order Taylor approximation.

The following derivation assumes that only the gray value distribution in frame 1 enters into the derivatives appearing in the matrix C of (31).

The Euler-Lagrange equations for the minimization problem (31) yield

$$-(g_2(x) - g_1(x - u)) \frac{\partial g_1(x - u)}{\partial u} - \alpha^2 \left(\begin{pmatrix} \frac{d}{dx} \\ \frac{d}{dy} \end{pmatrix}^T C^{-1} \begin{pmatrix} u_x \\ u_y \end{pmatrix} = 0 \quad (32a)$$

$$-(g_2(x) - g_1(x - u)) \frac{\partial g_1(x - u)}{\partial v} - \alpha^2 \left(\begin{pmatrix} \frac{d}{dx} \\ \frac{d}{dy} \end{pmatrix}^T C^{-1} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = 0. \quad (32b)$$

In order to visualize the effect of the oriented smoothness term, assume that the coordinate system is aligned with the eigenvector directions of C at the image location of interest. This results in

$$C = \begin{pmatrix} c_1 & 0 \\ 0 & c_2 \end{pmatrix}. \quad (33)$$

The second term in (32a) may then be written as

$$\begin{aligned} & \left(\begin{pmatrix} \frac{d}{dx} \\ \frac{d}{dy} \end{pmatrix}^T \begin{pmatrix} c_1^{-1} & 0 \\ 0 & c_2^{-1} \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix} \right. \\ & \quad \left. = \begin{pmatrix} -\frac{1}{c_1^2} \frac{dc_1}{dx} \\ -\frac{1}{c_2^2} \frac{dc_2}{dy} \end{pmatrix}^T \begin{pmatrix} u_x \\ u_y \end{pmatrix} + \frac{u_{xx}}{c_1} + \frac{u_{yy}}{c_2} \right). \quad (34) \end{aligned}$$

If both eigenvalues are large, they suppress the contribution of the smoothness term compared to the first term in (32). In order to simplify the subsequent attempts to solve (32), the components of C are considered to vary

slowly from pixel to pixel so that the derivatives of C can be neglected. This assumption enables us to simplify (32) into

$$-(g_2(x) - g_1(x - u)) \frac{\partial g_1(x - u)}{\partial u} - \alpha^2 \text{trace} \left(C^{-1} \begin{pmatrix} u_{xx} & u_{xy} \\ u_{xy} & u_{yy} \end{pmatrix} \right) = 0 \quad (35a)$$

$$-(g_2(x) - g_1(x - u)) \frac{\partial g_1(x - u)}{\partial v} - \alpha^2 \text{trace} \left(C^{-1} \begin{pmatrix} v_{xx} & v_{xy} \\ v_{xy} & v_{yy} \end{pmatrix} \right) = 0. \quad (35b)$$

These equations (35) provide the starting point for the first two solution approaches.

A. Iterative Approach Based on the Gray Value Difference

This approach, originally suggested in [17], assumes that an approximate solution $u_0(x)$ for the displacement vector field is known. Substitution of $u_0(x)$ for u will in general not satisfy (35). The problem then consists in the determination of a correction vector field $Du(x)$ such that (35) become satisfied. The initial approach along this line neglected the influence of the correction term $Du(x)$ in the determination of $\nabla \nabla u$ and $\nabla \nabla v$, respectively. It only investigated the effect of $Du(x)$ on the first term in (35).

Since Du is considered to be small, we retain only first order terms in components of Du and average over the environment which is used to estimate the first and second partial derivatives of the gray values as well as of the displacement vector field u . This approach—see Appendix 1 for the detailed derivation of (A1-7)—yielded:

$$\begin{aligned} CDu = & -[g_2(x) - g_1(x - u_0)] \bar{\nabla} g_1 \\ & + \alpha^2 \left[\frac{\text{trace}(C^{-1} \bar{\nabla} \nabla u_0)}{\text{trace}(C^{-1} \bar{\nabla} \nabla v_0)} \right]. \quad (36) \end{aligned}$$

Here, the overbar indicates the result of the averaging process. The solution approach envisaged in [17] consisted of the following steps:

- 1) Initialize $u_0(x) = 0$.
- 2) Select those image areas where $\det(C) \neq 0$ and, in analogy to (16),

$$\frac{\det C}{(\frac{1}{2} \text{trace } C)^2} > \text{threshold}, \quad (37)$$

i.e., where there is sufficient gray value variation to estimate both components of the displacement vector without recourse to a smoothness requirement. This should essentially cover the image areas around gray value corners as shown in [21]—see Fig. 2.

- 3) Set $\alpha^2 = 0$, estimate the displacement vector at image locations selected according to step 2) and improve these estimates by an iterative procedure analogous to the one described in Section II-A.

4) Switch on the "oriented smoothness" constraint by letting α^2 become nonzero and spread the estimates obtained in step 3) into other image areas hitherto spared.

The implementation of this approach made obvious that the following problem could not be handled as anticipated. Image areas excluded in step 2) are characterized by an ill-conditioned matrix C . Originally, it has been anticipated to substitute minimal values for the elements of C . These minimal values should have been chosen based on the error estimates for first and second partial derivatives of $g1$ as given in the Appendix of [15]. It turned out, however, that these estimates appeared to be too small to prevent numerical difficulties. Therefore, the inversion of C to obtain the solution of equation (36) resulted in severe numerical difficulties.

Once this explanation had been corroborated, it appeared sensible to drop the simplification of relying only on the first term in (35) in order to estimate an initial correction vector field Du .

B. Extended Iterative Approach

Following the analysis discussed at the end of the preceding section, the effect of a correction Du on the terms containing $\nabla\nabla u$ and $\nabla\nabla v$ has been investigated as well—see Appendix 2 for details. It resulted in the following equation:

$$\begin{aligned} & [(\nabla g1)(\nabla g1)^T + b^2(\nabla\nabla g1)(\nabla\nabla g1)^T \\ & \quad + \alpha^2 m \operatorname{trace}(C^{-1}I)Du] \\ & = -[g2(x) - g1(x - u_0)]\nabla g1 \\ & \quad + \alpha^2 \left[\frac{\operatorname{trace}(C^{-1}\nabla\nabla u_0)}{\operatorname{trace}(C^{-1}\nabla\nabla v_0)} \right]. \end{aligned} \quad (38)$$

Here, I denotes a 2×2 unit matrix and m stands for the magnitude value of the weight factor at the center pixel position in the operator mask for $\partial^2/\partial x^2$. In the case of a 5×5 Beaudet operator mask [1], this results in the value $m = \frac{2}{35}$.

The important difference between (36) and (38) consists of the addition of the term $\alpha^2 m \cdot \operatorname{trace}(C^{-1})$ in both diagonal elements of the coefficient matrix for the unknown correction vector Du .

Its influence may be visualized by the following qualitative argument. Assume that the same solution steps as outlined in Section III-A are performed on the basis of (38). Since α^2 will be zero throughout the first three steps, the difference between (36) and (38) will become relevant only in step 4). Extension of solutions obtained around gray value corners into areas with ill-conditioned C -matrices will result in large values for $\operatorname{trace}(C^{-1})$. The addition of such large values to the diagonal elements of the coefficient matrix for Du should prevent the coefficient matrix for Du from becoming singular. It is expected, therefore, that one may be able to invert this coefficient matrix and hence get a solution for Du .

Implementation of this approach did not yield results

commensurate with our expectations. The trouble could be diagnosed to be related to the constant α^2 which indicates the relative weight between the smoothness term and the gray value difference term in (31).

In order to achieve a noticeable effect by the term $\alpha^2 m \operatorname{trace}(C^{-1})$, α^2 had to be given values between 500 and 1000 which appeared to be unreasonably high. Such values had the consequence that the smoothness contribution on the right-hand side of (38) would dominate the other term on the right hand side which is based on the gray value difference. Since for low iteration indexes in step 4) the values of $u_0(x)$ would differ from zero only around gray value corners, the estimates of $\nabla\nabla u_0$ and $\nabla\nabla v_0$ turned out to be somewhat erratic initially. Large values for α^2 would exacerbate this behavior.

It is illuminating to discuss equation (38) at the beginning of step 4) in a situation where $u_0(x)$ is still zero throughout the image window under consideration. We may assume that the local coordinate system is aligned with the principal curvature directions of the gray value distribution within this window, i.e., $g1_{xy} = 0$. Using these conventions, (38) reduces to

$$[C + I\alpha^2 m \operatorname{trace}(C^{-1})Du] = -[g2(x) - g1(x - u_0)]\nabla g1 \quad (39)$$

because $u_0(x) = 0$ implies $\nabla\nabla u_0 = \nabla\nabla v_0 = 0$. This vector equation for the components Du and Dv of the correction vector Du can be solved explicitly—see Appendix 3:

$$Du = -\frac{g2(x) - g1(x - u_0)}{g1_x + \alpha^2 m \frac{1 + b^2 g1_{yy}^2 / g1_x^2}{b^2 g1_x^2 g1_{yy}^2}} \quad (40a)$$

$$Dv = -\frac{g2_y}{g1_{yy} + \alpha^2 m \frac{1 + g1_x^2 / (b^2 g1_{yy}^2)}{b^2 g1_x^2 g1_{yy}^2}} \quad (40b)$$

The expressions (40) facilitate a quantitative discussion of the influence exerted during the initial iterations by an oriented smoothness constraint in the form given by (29) and (31).

If the smoothness constraint is omitted, i.e., $\alpha^2 = 0$, the results can be related to previously known entities:

- Du is equal to minus the gray value difference divided by the gradient.
- Dv is equal to the closed form solution obtained by Nagel [15] at gray value corners.

Since all terms in the correction factor are positive, the contributions with $\alpha^2 \neq 0$ reduce these estimates.

For 192×256 pixel 8-bit gray value images like those in Fig. 1, a 5×5 operator mask will yield numerical values well exceeding 10 for both $g1_x$ and $g1_{yy}$ at gray value corners. The value of b^2 has been set equal to

$$\xi^2 = 2$$

for this mask size. Since $m = \frac{2}{33}$, even values of 10^3 for α^2 do not result in noticeable corrections. This does not matter at gray value corners since there the uncorrected estimates are known to be acceptable.

The smoothness constraint should influence the estimate along straight line gray value transitions because there only one component of the displacement vector will be determined by the gray value distribution. We may study the behavior of Du according to (40) by "unbending" the corner, that is by reducing the curvature $g1_{yy}$ of a locus line with strong gray value slope $g1_x$. In the limit of vanishing $g1_{yy}$ both components of Du as given by (40) will tend to zero.

If we study the origin of this factor, we see that it is related to trace (C^{-1}) and there to the fact that our smoothness weight matrix F has been normalized by its determinant—see (29).

As a consequence of this analysis, it has been attempted to use some other convention than division by $\det(F)$ in order to normalize F .

IV. DIFFERENT NORMALIZATIONS OF THE WEIGHT MATRIX

Another way to formulate the insight provided by the discussion of (40) consists in the following consideration. If the locus line of maximum gray value slope—i.e., large $g1_x$ —exhibits only a small curvature $g1_{yy}$, then the term $b^2 g1_{yy}^2$ in the numerator of trace (C^{-1})—see (A3-6b)—may be dropped compared to $g1_x^2$. As a result, trace (C^{-1}) becomes approximately equal to $1/(b \cdot g1_{yy})^2$ which assumes values between 0.1 and 1. Even with $\alpha^2 = 100$, the term $\alpha^2 m / (b^2 g1_{yy}^2)$ added to the diagonal elements of C in (39) does not significantly reduce the imbalance between the eigenvalues of the coefficient matrix on the left hand side of this equation. The consequence is to raise the weight of the smoothness term by looking for another normalization factor in (29).

A. Normalization by Trace (F) Instead of by $\det(F)$

One possibility which springs to the mind consists of taking trace (F) rather than $\det(F)$ (see also [23]). Instead of (31) we therefore investigate the minimization of:

$$\iint dx dy \left\{ [g2(x) - g1(x - u)]^2 + \alpha^2 \text{trace} \left[(\nabla u)^T \frac{F}{\text{trace } F} (\nabla u) \right] \right\} \Rightarrow \min. \quad (41)$$

If we take into account the contribution of the smoothness term in (41), we obtain the following result for a gray value corner:

$$Du = -\frac{\overline{g2(x)} - \overline{g1(x - u_0)}}{g1_x} \cdot \frac{1}{1 + \frac{\alpha^2 m}{g1_x^2}} + \alpha^2 \frac{b^2 g1_{yy}^2 u_{0,yy} + g1_x^2 u_{0,yy}}{(g1_x^2 + b^2 g1_{yy}^2)(g1_x^2 + \alpha^2 m)} \quad (42a)$$

$$Dv = -\frac{g2_y}{g1_{yy}} \cdot \frac{1}{1 + \frac{\alpha^2 m}{b^2 g1_{yy}^2}} + \alpha^2 \frac{b^2 g1_{yy}^2 v_{0,yy} + g1_x^2 v_{0,yy}}{(g1_x^2 + b^2 g1_{yy}^2)(b^2 g1_{yy}^2 + \alpha^2 m)}. \quad (42b)$$

If we completely "unbend" the gray value corner, i.e., if we set $g1_{yy} = 0$, we obtain instead of (42):

$$Du = -\frac{\overline{g2(x)} - \overline{g1(x - u_0)}}{g1_x} \cdot \frac{1}{1 + \frac{\alpha^2 m}{g1_x^2}} + \alpha^2 \frac{u_{0,yy}}{(g1_x^2 + \alpha^2 m)} \quad (43a)$$

$$Dv = \frac{1}{m} v_{0,yy}. \quad (43b)$$

Equation (43) shows the influence of the "oriented smoothness" constraint in an especially clear manner. At maximum slope of a strong straight line gray value transition, i.e., large values for $g1_x$ and small values for all other first and second order partial derivatives, the component perpendicular to the edge will essentially correspond to $(-g1_x/g1_{yy})$. The correction for the displacement vector component parallel to the edge will only vanish if the second partial derivative of the displacement vector field in this direction vanishes.

B. Using F without Normalization

In addition to the choice of both $\det(F)$ as well as trace (F) as a normalization factor for the weight matrix in an "oriented smoothness" constraint, we have explored the alternative of using F given by (27) without normalization [22]. Instead of (31) we minimize

$$\iint dx dy \{ [g2(x) - g1(x - u)]^2 + \alpha^2 \text{trace} [(\nabla u)^T F (\nabla u)] \} \Rightarrow \min. \quad (44)$$

In analogy to (40), the first correction vector Du at a location of maximum gray value slope after starting with $u_0(x) = 0$ will be

$$Du = -\frac{\overline{g2(x)} - \overline{g1(x - u_0)}}{g1_x} \cdot \frac{1}{1 + \alpha^2 m (1 + b^2 g1_{yy}^2 / g1_x^2)} \quad (45a)$$

$$Dv = -\frac{g2_y}{g1_{yy}} \cdot \frac{1}{1 + \alpha^2 m (1 + g1_x^2 / (b^2 g1_{yy}^2))}. \quad (45b)$$

The factor α^2 can be chosen to be smaller than in the case of (40). One would expect that α^2 should be of the order of the variance of gray value differences $[g2(x) - g1(x - u)]$ although this consideration does not take into

account the "orientation" aspect of the smoothness constraint. Nevertheless, a value of α^2 in the range of $2\sigma^2$ appears to be more attractive *a priori* than 10 or 100 times this value.

V. HEURISTIC MODIFICATIONS

Although the preceding sections have shown how it is possible to derive closed form expressions for the correction terms and thereby how to study the influence of certain parameters analytically, we feel that we still have some way to go towards a solution of the nonlinear system of partial differential equations resulting from the various formulations for the "oriented smoothness" constraint. Unless we can study such solutions, however, it is difficult to judge the relative merits of these and other possible formulations.

In addition to the approach outlined in the preceding section we have studied in our current implementation several heuristic modifications. These have been motivated by attempts to counterbalance certain effects observed during an extended range of experiments. We did not hesitate to make some effects more visible or to suppress them by heuristic modifications. Our experience has shown us that once an effect could be clearly identified, it has been possible to study it not only experimentally but also through specific modifications to the theoretical approach. It appears useful, therefore, to present these more or less heuristic modifications and to demonstrate their effect.

A. Enhancement of the Smoothness Term in the Coefficient Matrix for Du

Larger values of α^2 result in greater contributions to the diagonal terms of the coefficient matrix on the left-hand side of (38) and the analogous equations for the other choices of the weight matrix discussed in Section IV. This facilitates obtaining estimates for both components of the displacement vector along almost straight line gray value transitions. The disadvantage is that the terms multiplied by α^2 on the right-hand side of these equations may achieve a weight which does not appear to be necessarily beneficial during early iterations. We therefore introduced a heuristic correction factor c such that, for example, the term $c \cdot \alpha^2 m \cdot \text{trace}(F)$ is added to the diagonal elements of C in order to obtain the coefficient matrix for Du . This factor c has been set to 8 in the case where the weight matrix $W = F$ has been used.

B. Lower Limit for Eigenvalues of the Coefficient Matrix

In order to prevent a singularity of the coefficient matrix for Du in areas with practically homogeneous gray values, the eigenvalues of C are replaced by a lower limit $\gamma = 2\sigma^2$ ($\sim 5-12$) whenever they drop below this threshold. If both eigenvalues are smaller than γ the first term on the right hand side of (38)—and analogously for the other choices of normalization—will be suppressed.

C. Averaging u_0

The second partial derivatives of u_0 and v_0 are computed by applying Beaudet's operators [1] to the components u_0 and v_0 of the displacement vector field estimate u_0 obtained in the course of the iterative solution approach. These operators are based on the assumption that the variation of $u_0(x)$ and $v_0(x)$ can be approximated within the operator window by a bivariate polynomial. As a consequence of this approach the value of $u_0(x)$ as well as $v_0(x)$ averaged over the window is not identical to the value of u_0 or v_0 , respectively, at the window center. Since the correction Du will eventually become smaller than the difference between the average and the center value, the center value at each pixel location is replaced by the average value obtained by application of Beaudet's averaging operator to the field u_0 within the operator window.

D. Interframe Average for Derivatives

Nagel [14] has shown that one may use the interframe average for the computation of first and second partial derivatives rather than using the derivatives obtained from frame 1 alone. Although this ensues a slightly greater number of iterations, it extends the range wherein the approach catches on. This can be understood, for example, in the case of the edge line between the rear cab window and the car top. Between frame 1 and frame 2 in Fig. 1, the rear window area moves across the fairly constant bright area of the car top. Since the gradient in this area of frame 1 is small, one initially obtains large displacement estimates. They are a consequence of the attempt to explain the large gray value difference between bright car top and dark rear window on the basis of too small a gray value slope. The gray value slope averaged between the areas to be compared gives a better displacement estimate. Of course, the opposite effect happens for example at the transition between rear window and trunk. There the estimate resulting from this approach will be somewhat larger than in the case where only the gradient from frame 1 had been used. The overall effect of this modification, however, appears to be positive. Once the iteration converges, the two windows to be compared will exhibit essentially the same gray value variation and hence the averaging process does not bias the final result.

E. The Strength of the Smoothness Constraint as a Function of the Iteration Count

During experiments with the weight matrix $W = F$, the value of α^2 has been handadjusted during the iteration in order to keep the maximum contribution from the smoothness correction below a heuristic limit.

Deviating from the steps outlined towards the end of Section III-A, we start immediately with $u_0(x) = 0$ and omit steps 1)-3). In order to prevent the coefficient matrix of Du to become ill-conditioned in image areas with almost constant gray values, α^2 is not set to zero but to a small nonzero value. Fig. 3 presents our current choice of α^2 as a function of the iteration count.

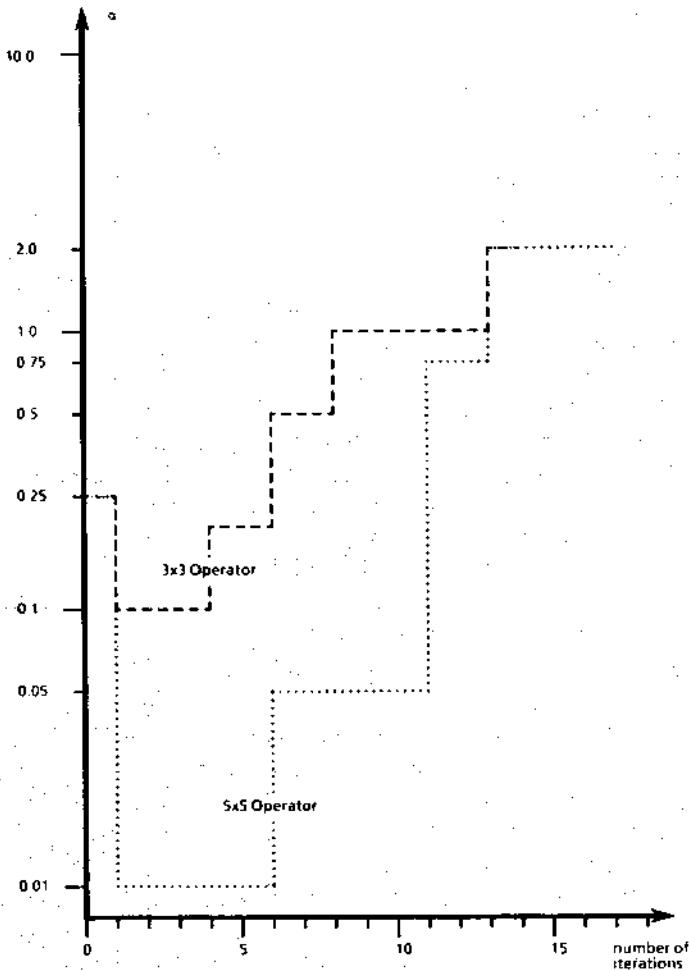


Fig. 3. The value of the relative weight α^2 of the smoothness constraint as a function of the iteration count in case where the weight matrix $W = F$ has been used.

During experiments with $W = F/\text{trace}(F)$, the value of α^2 has been doubled as soon as the maximum correction dropped below 0.1 pixels per frame.

VI. RESULTS

In order to facilitate a comparison of the approaches investigated here with earlier ones, the approach of Horn and Schunck [11] has been reimplemented and applied to our data. Since our approach incorporates various modifications in addition to the "oriented smoothness" constraint, we include a series of experimental results which demonstrate the gradual transition from the Horn and Schunck approach to the ones discussed in previous sections.

A. The Approach of Horn and Schunck with Modifications

Fig. 4(a) presents the result of applying the approach of Horn and Schunck [11] to the window shown in Fig. 1(e) and the corresponding window from the subsequent frame. Fig. 4(b) presents the result of applying to the same data a general smoothness approach according to Horn and Schunck—modified, however, by using interframe

displaced gray value differences. The following formula is employed to estimate the correction Du :

$$Du = -\frac{(g_2(x) - g_1(x - u_0))}{g_1(x - u_0)_x + g_1(x - u_0)_y + \alpha^2} \begin{pmatrix} g_1(x - u_0)_x \\ g_1(x - u_0)_y \end{pmatrix} \quad (46)$$

It can be seen that this alone provides a considerable improvement compared to the method described by Horn and Schunck [11]. Fig. 4(c) demonstrates to what extend the use of interframe averages for the first derivatives as explained in Section V-D modifies the results in comparison with Fig. 4(b). The correction term has been computed in this case according to the following formula:

$$Du = -\frac{(g_2(x) - g_1(x - u_0))}{g_x^2 + g_y^2 + \alpha^2} \begin{pmatrix} g_x \\ g_y \end{pmatrix} \quad (47a)$$

with

$$g_x = \frac{g_{2x}(x) + g_{1x}(x - u_0)}{2} \quad (47b)$$

and

$$g_y = \frac{g_{2y}(x) + g_{1y}(x - u_0)}{2} \quad (47c)$$

3×3 Beaudet operators have been used to compute the derivatives. The data for $g_1(x - u_0)$ has been obtained by bilinear interpolation according to the current value of u_0 . A value of $\alpha^2 = 50$ has been used in these experiments. Figs. 5–7 present analogous results after 3, 10, and 30 iterations.

B. "Oriented Smoothness" with Weight Matrix $W = F$

Fig. 8(a) presents the result of the "oriented smoothness" approach outlined in Section IV-B, i.e., with just F as the weight matrix, after the first iteration for a 3×3 pixel operator window. This result may be compared directly to the smoothness approach. One can see that the estimates are different from zero in image areas where significant gray value variations have been displaced between frame 11 and frame 12. Fig. 8(b) shows analogous results for a 5×5 pixel operator window. It can be seen that the 3×3 window is somewhat more susceptible to digitization noise because only nine measurements are available to estimate the six unknowns required to describe the Taylor expansion up to the second order. A 5×5 window provides 25 measurements to estimate the same coefficients. Within a 3×3 operator window, moreover, the curvature estimate of the gray value distribution is somewhat larger. This has to be taken into account by an appropriate choice of the parameters. Fig. 9 shows how these estimates have improved after the third iteration. The iteration process has been stopped manually after the maximal magnitude of the correction term

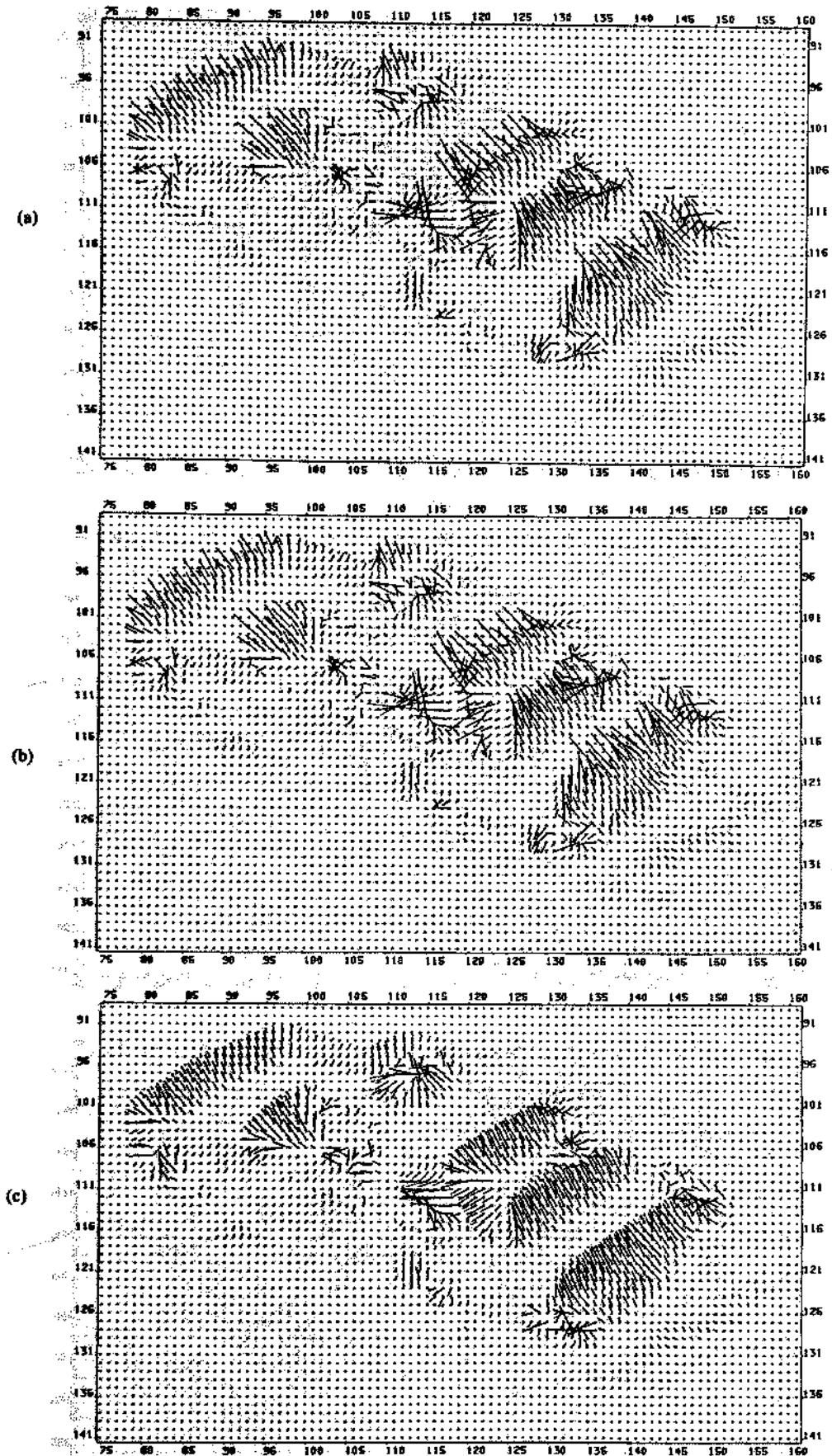
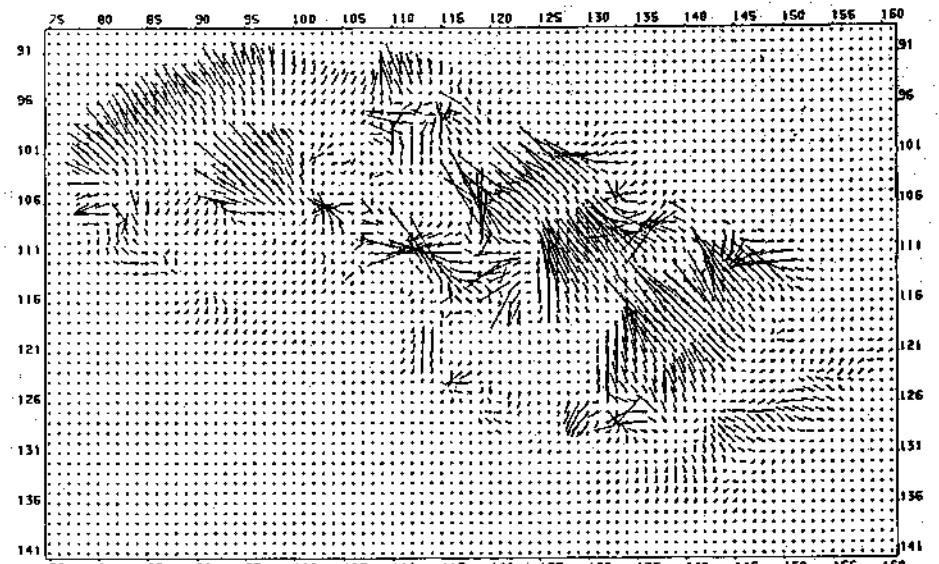
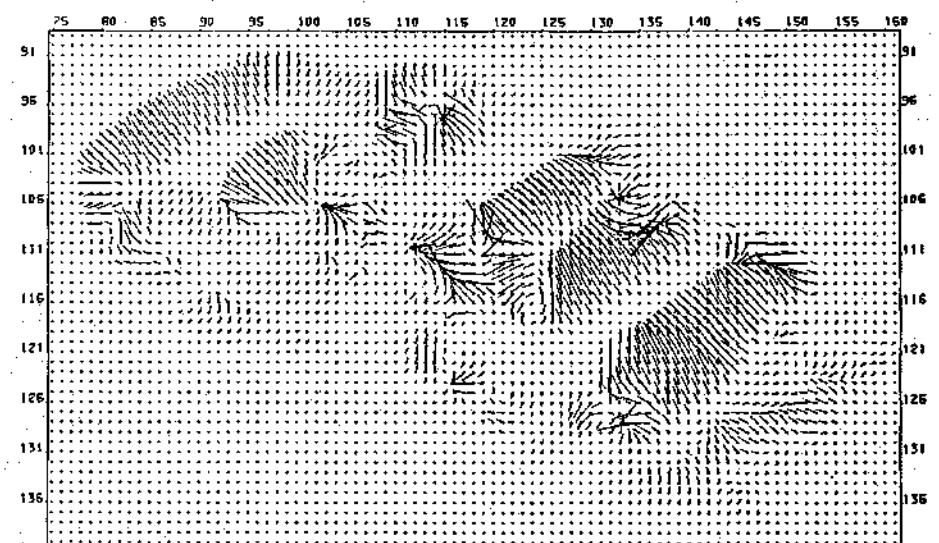


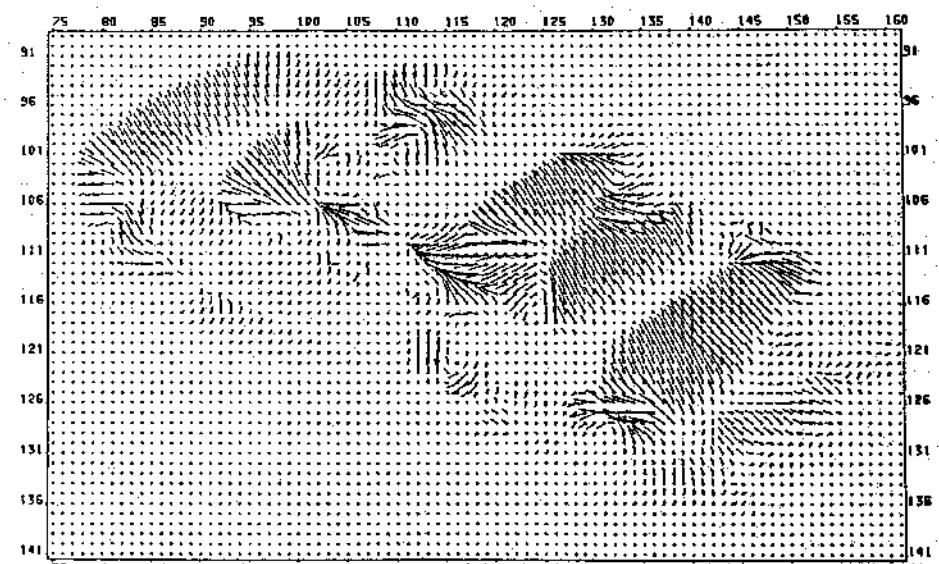
Fig. 4. Result after iteration 1 for the approach of Horn and Schunck [11], applied to the window shown in Figure 1e and the corresponding window from the subsequent frame: (a) as described by Horn and Schunck [11]; (b) analogous to (a), but using the displaced frame difference according to (46) in order to compute the correction term; (c) analogous to (b), but including in addition the interframe average of gradients according to (47).



(a)

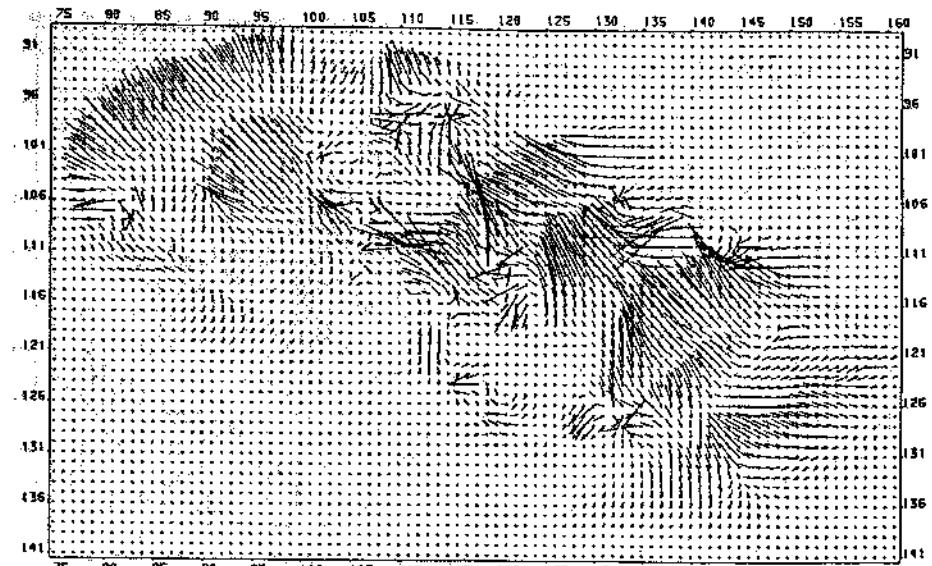


(b)

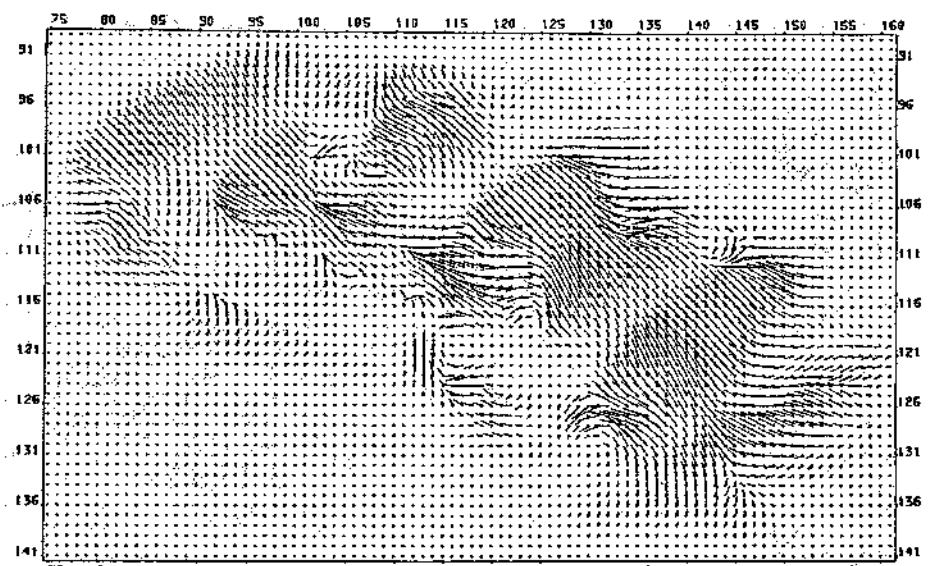


(c)

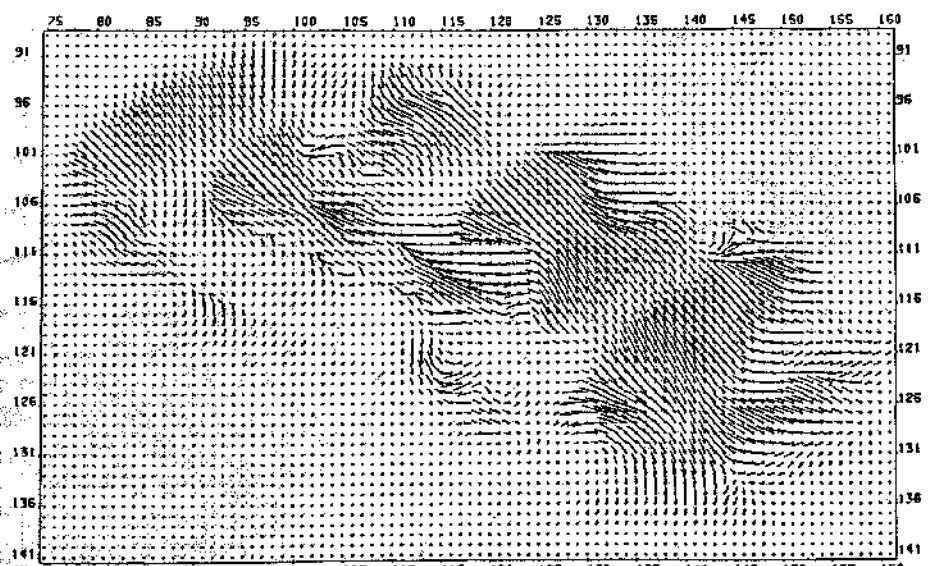
Fig. 5. Analogous to Fig. 4, but after iteration 3.



(a)



(b)



(c)

Fig. 6. Analogous to Fig. 4, but after iteration 10.

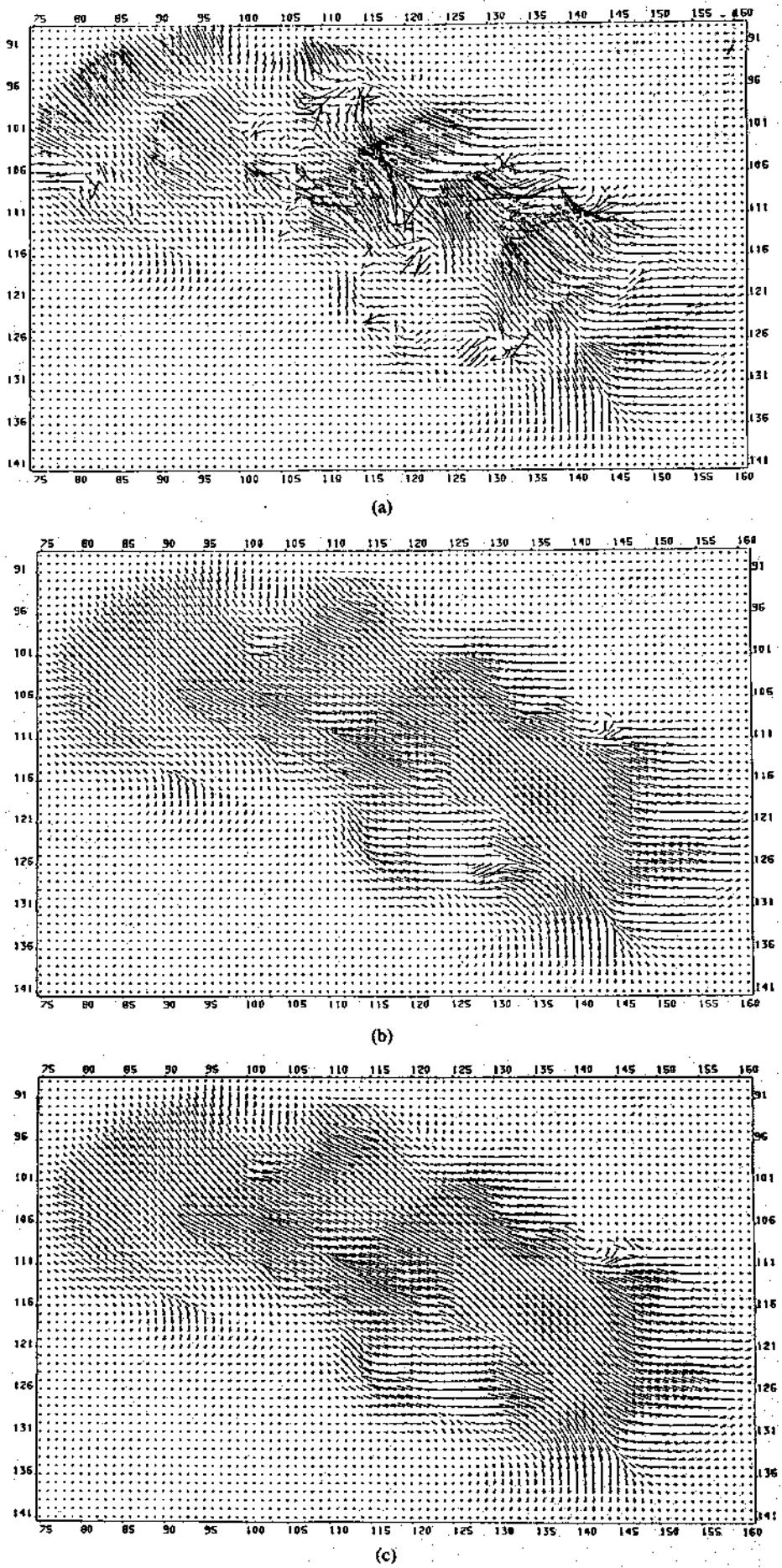


Fig. 7. Analogous to Fig. 4, but after iteration 30.

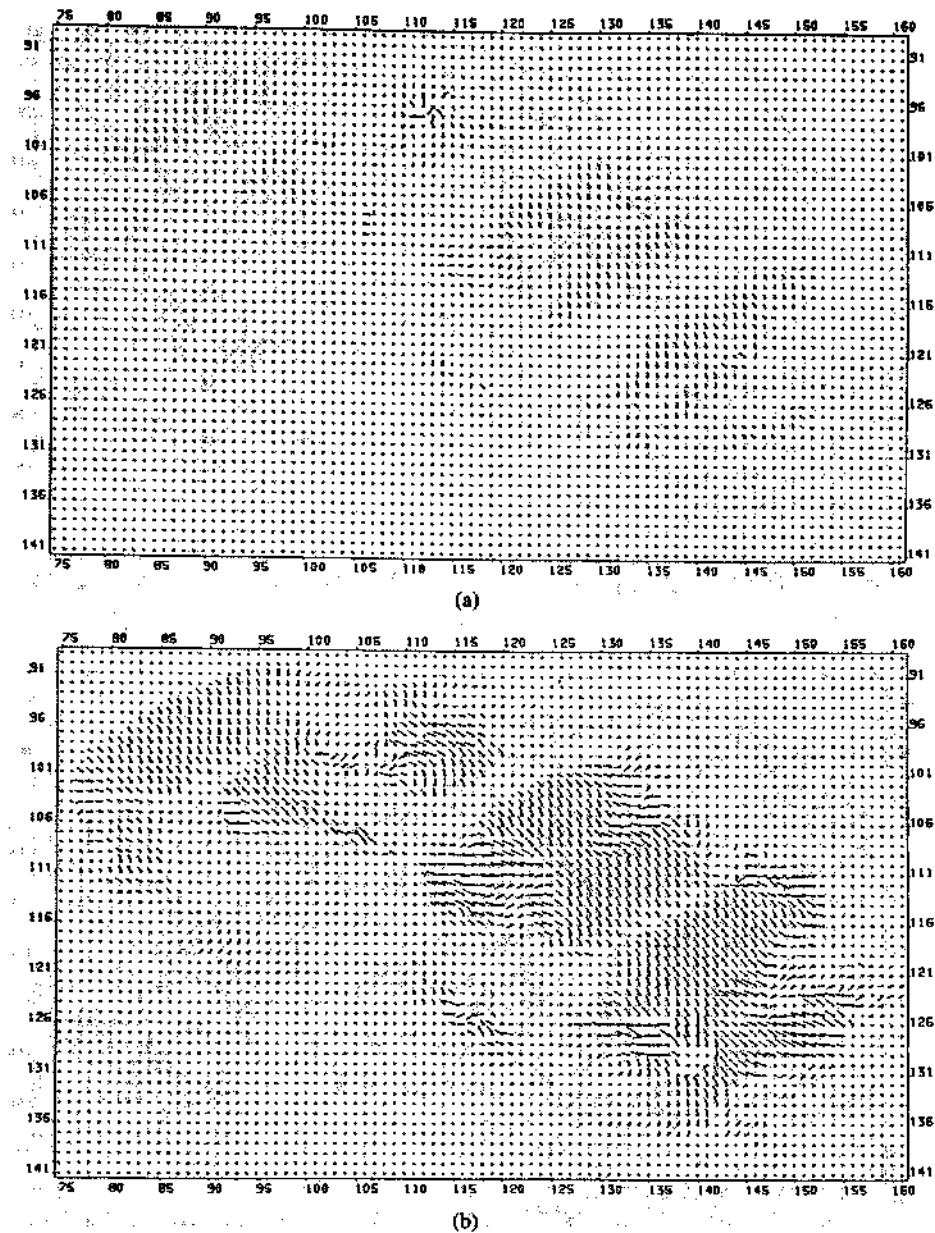


Fig. 8. Result after iteration 1 of estimating the displacement vector field using the "oriented smoothness" constraint with weight matrix $W = F$ as discussed in Section IV-B for the window indicated by Fig. 1(e) from frames 11 and 12. The initial value has been $u_0(x) = 0$: (a) for 3×3 pixel operator window with $\alpha^2 = 0.25$; $c = 8$; $\gamma = 12.5$; (b) for 5×5 pixel operator window with $\alpha^2 = 0.25$; $c = 8$; $\gamma = 5$.

dropped below $\frac{1}{5}$ of a pixel per frame in the 13th and 17th iteration, respectively. The final result is shown in Fig. 10.

Attention is drawn to the following aspects:

1) Apart from the moving shadow area behind the taxicab—which is, moreover, partially occluded by branches of the tree in the foreground—the displacement vector field is fairly smooth inside the area covered by the image of the moving taxicab. This displacement vector field is plausible.

2) The displacement vector field exhibits the expected (near) discontinuities, both along contours parallel to the displacement as well as along contours perpendicular to the displacement.

3) The displacement vector field does not exhibit sizable discontinuities across strong gray value transitions within the image area of the moving car.

One problem observed in these experiments is connected with the choice for the relative weight of the smoothness term, i.e., α^2 . Small values for α^2 will yield reasonable results around gray value corners where F takes on large values. In rather homogeneous image areas like those for the roof or the trunk of the taxicab, a large number of iterations is required in order to spread the displacement vector estimates from areas with greater gray value variations where both components of the displacement vector field can be estimated. If one gets impatient and increases the weight of the smoothness term in order to

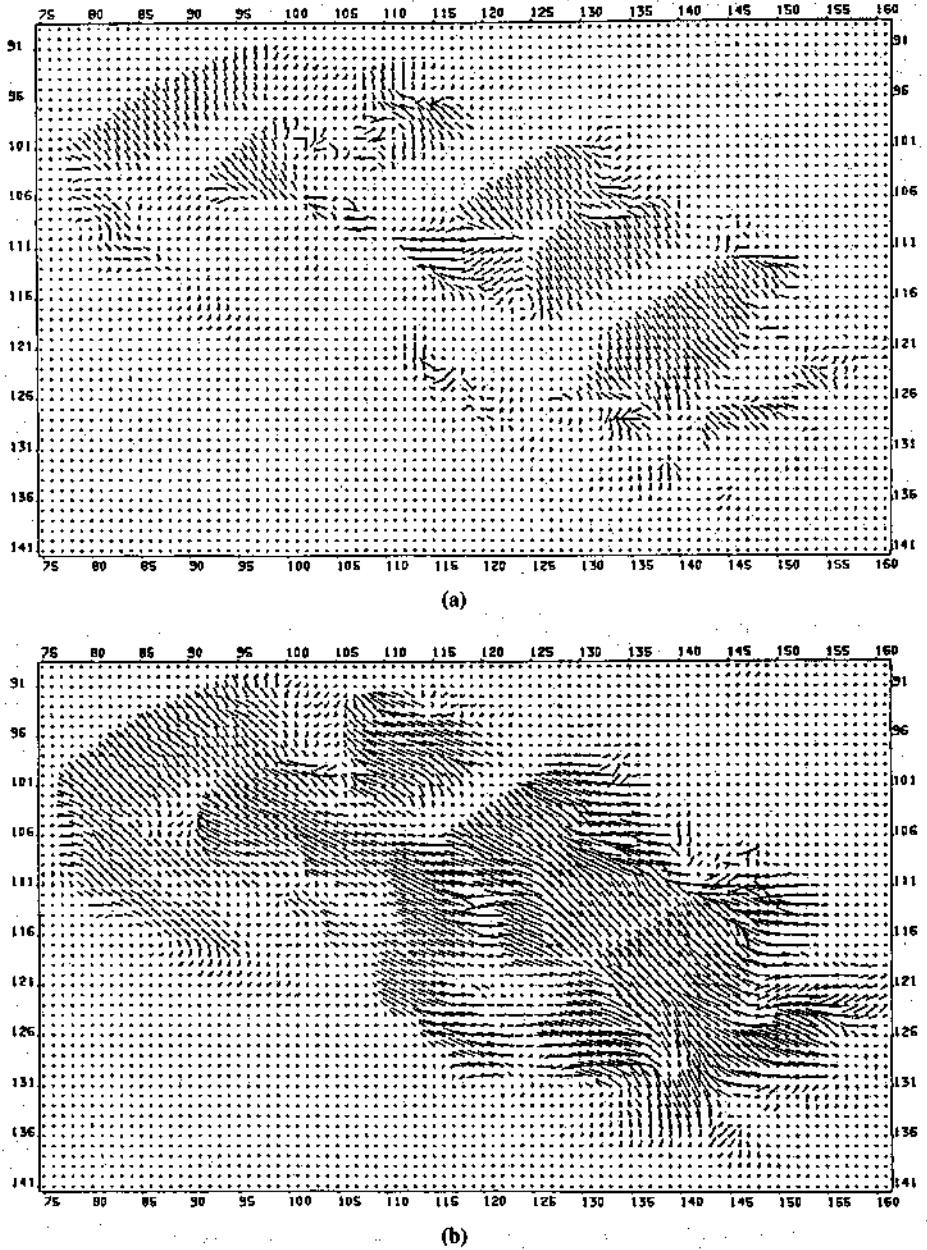


Fig. 9. Result after iteration 3 of estimating the displacement vector field using the "oriented smoothness" constraint with weight matrix $W = F$ as discussed in Section IV-B for the window indicated by Fig. 1(e) from frames 11 and 12: (a) for 3×3 pixel operator window with $\alpha^2 = 0.1$; $c = 8$; $\gamma = 12.5$; (b) for 5×5 pixel operator window with $\alpha^2 = 0.01$; $c = 8$; $\gamma = 5$.

spread displacement estimates faster into such areas, then the second partial derivatives $\nabla\nabla u_0$ and $\nabla\nabla v_0$ must already be sufficiently small around gray value corners or along strong edges in order to avoid too large correction terms.

C. "Oriented Smoothness" with Weight Matrix $W = F/\text{trace}(F)$

Fig. 11(a) presents the result for the first iteration starting from $u_0 = 0$ and using the weight matrix $W = F/\text{trace}(F)$ as discussed in Section IV-A. A 3×3 pixel operator window has been used. The analogous result for a 5×5 pixel operator window is shown in Fig. 11(b).

The result for iterations 3, 10, and 30 are given in Figs. 12–14.

D. Using Previous Results to Predict Starting Values

Such a large number of iterations appears to be only necessary if no knowledge about the displacement vector field is available. In most cases, the displacement vector field has to be calculated for an image sequence where the results from preceding frames can be used as a starting value rather than the uniformly vanishing starting field employed for the first frame pair. Fig. 15(a) shows the result obtained after 12 iterations for a window around the bright taxicab in the center of Fig. 1(c) as given in Fig.

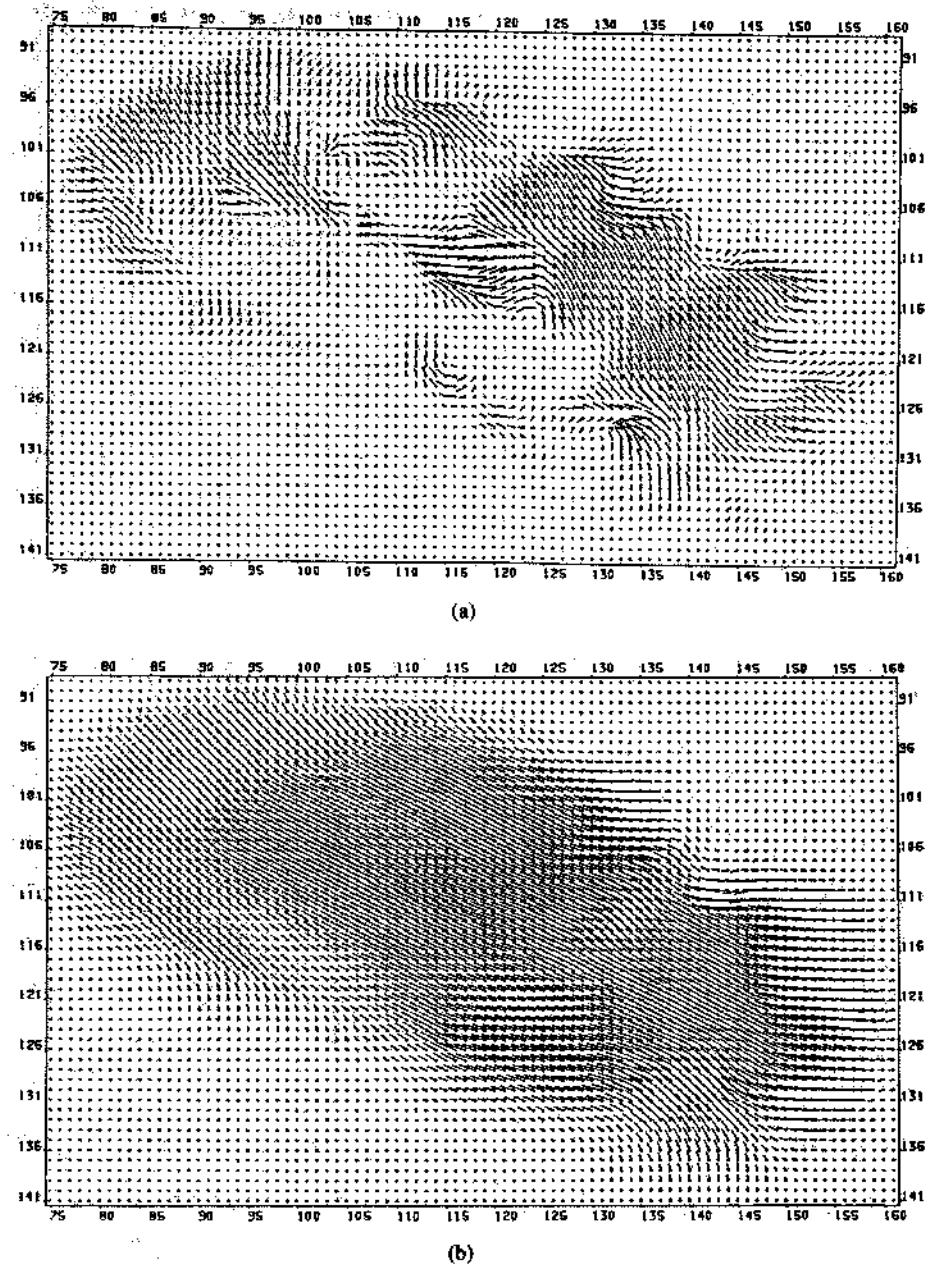
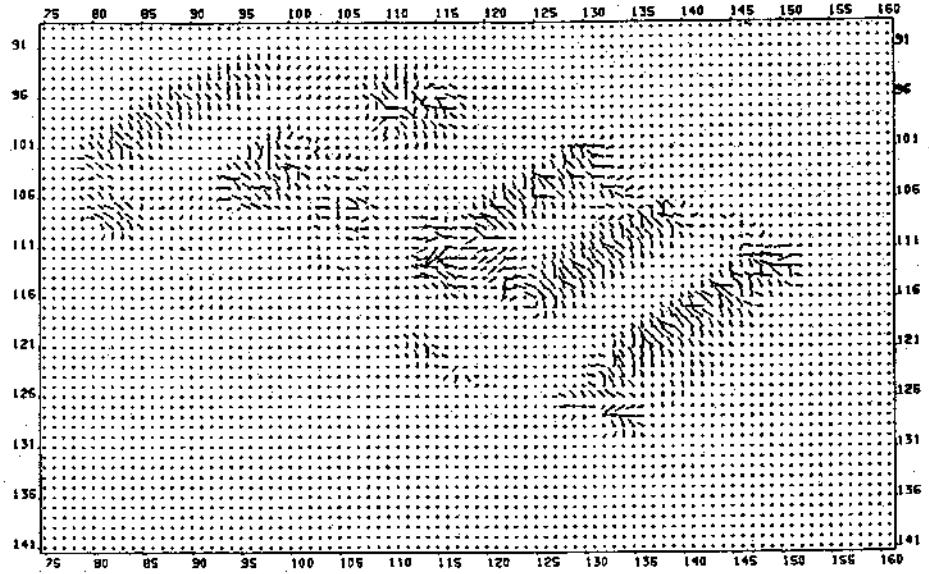


Fig. 10. Analogous to Fig. 9, but after the corrections for the displacement vector have been everywhere smaller than $\frac{1}{2}$ of a pixel per frame:
(a) for 3×3 pixel operator window with $\alpha^2 = 1$; $c = 8$; $\gamma = 12.5$; iteration 13; (b) for 5×5 pixel operator window with $\alpha^2 = 2$; $c = 8$; $\gamma = 5$; iteration 17.

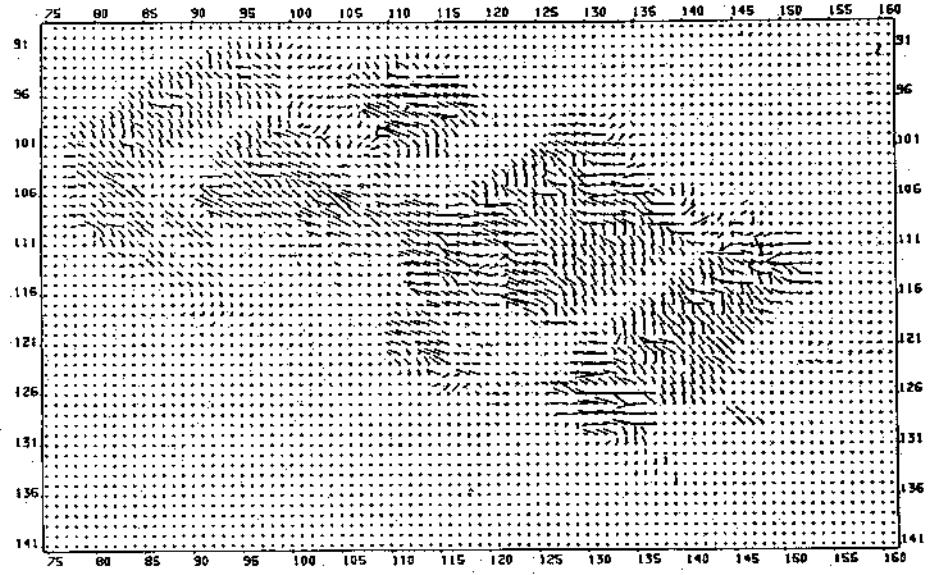
1(f) and the corresponding window from the subsequent frame, i.e., frames 20 and 21, again with the uniformly vanishing starting field $u_0 = 0$. Fig. 15(b) presents the result after four additional iterations for windows from the subsequent frame pair 21 and 22, initialized with the results shown in Fig. 15(a). Similarly, Fig. 15(c) presents the analogous results for frame pair 22 and 23 after two further iterations.

This approach has been pursued for each frame pair between frames 21 and 30. The window from the last frame of this series, frame 30, is shown in Fig. 1(g). The result for the displacement vector field in this window obtained

after two iterations starting from the result of the preceding frame pair (28 and 29) as initialization is shown in Fig. 16(a). For comparison, the result for the same windows, but obtained with $u_0 = 0$ as the starting value is shown in Fig. 16(b). The difference between these two results, i.e., the dependency upon the initialization, is shown in Fig. 16(c). The differences in Fig. 16(c) are partially due to the fact that the fairly homogeneous gray value area behind the moving taxicab can be matched with either starting value. In one case this area is assumed to be stationary, whereas in the other case this area is assumed to move with the car. This latter hypothesis is ac-



(a)



(b)

Fig. 11. Result after iteration 1 of estimating the displacement vector field using the "oriented smoothness" constraint with weight matrix $W = F/\text{trace}(F)$ as discussed in Section IV-A for the window indicated by Fig. 1(e) from frames 11 and 12. The initial value has been $u_0(x) = 0$: (a) for 3×3 pixel operator window with $\alpha^2 = 50$; $c = 2$; $\gamma = 12.5$; (b) for 5×5 pixel operator window with $\alpha^2 = 10$; $c = 1$; $\gamma = 12.5$.

ceptable to the extent that the gray values are due to the shadow moving together with the car. Discrimination between these two hypotheses requires additional assumptions about what caused the observed gray values. It will have to be studied to which extend conceptually inappropriate starting values can be balanced by more iterations in image areas with fairly homogeneous gray values and gradual gray value transitions.

It can be seen that a few iterations will be sufficient for image areas with sizable gray value variations if appropriate starting values are available as in the case of an image sequence.

It should be noticed that another object with different displacement intrudes into the window. The estimates seem to be reliable enough to warrant segmentation approaches based on such displacement vector fields.

In conclusion, these observations are taken to be evidence that the concept of an "oriented smoothness" constraint as implemented appears to be a good step towards the solution of the problem to estimate displacement vector fields from image sequences. The analysis has been pushed to the point where closed form solutions for the correction term can be discussed, based on fairly general assumptions about the operators employed to compute the

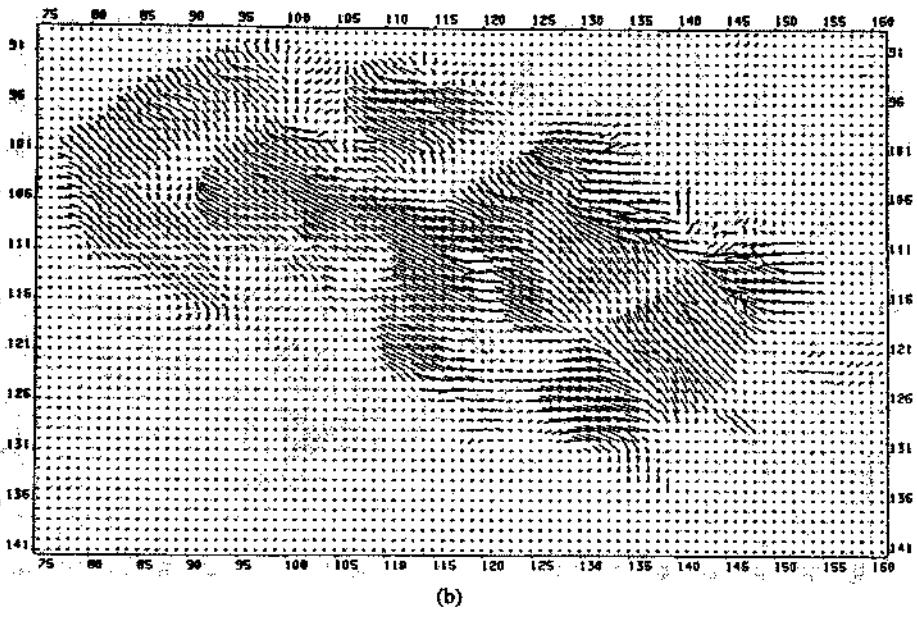
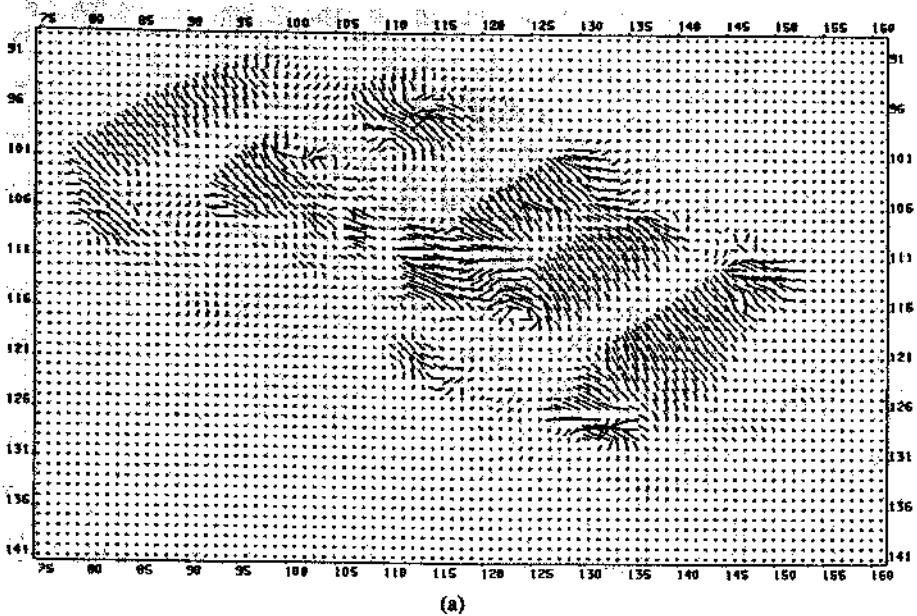


Fig. 12. Analogous to Fig. 11, but after iteration 3: (a) for 3×3 pixel operator window with $\alpha^2 = 50$; $c = 2$; $\gamma = 12.5$; (b) for 5×5 pixel operator window with $\alpha^2 = 10$; $c = 1$; $\gamma = 12.5$.

partial derivatives. It has been possible to study the dependency on the window size for 3×3 and 5×5 operator windows without any modification to the formulas.

As has been mentioned in previous sections, there remain a number of problems which we currently investigate. We have drawn attention to these problems which we have identified so far in order to enable the reader to form his own judgement about the merits of this approach.

APPENDIX 1 DERIVATION OF (36)

We start from (35). In order to simplify the notation, the argument x for the approximate displacement vector field $u_0(x)$ as well as for the correction vector field $Du(x)$ will be omitted henceforth.

Since Du is considered to be small, we retain only first order terms in components of Du :

$$\begin{aligned} g1(x - u) &= g1(x - u_0 - Du) \\ &\approx g1(x - u_0) - (\nabla g1(x - u_0))^T Du \end{aligned} \quad (A1-1)$$

and

$$\frac{\partial g1(x - u)}{\partial u} \approx \frac{\partial g1(x - u_0)}{\partial u} = -\frac{\partial g1(x - u_0)}{\partial x} \quad (A1-2a)$$

$$\frac{\partial g1(x - u)}{\partial v} \approx \frac{\partial g1(x - u_0)}{\partial v} = -\frac{\partial g1(x - u_0)}{\partial y} \quad (A1-2b)$$

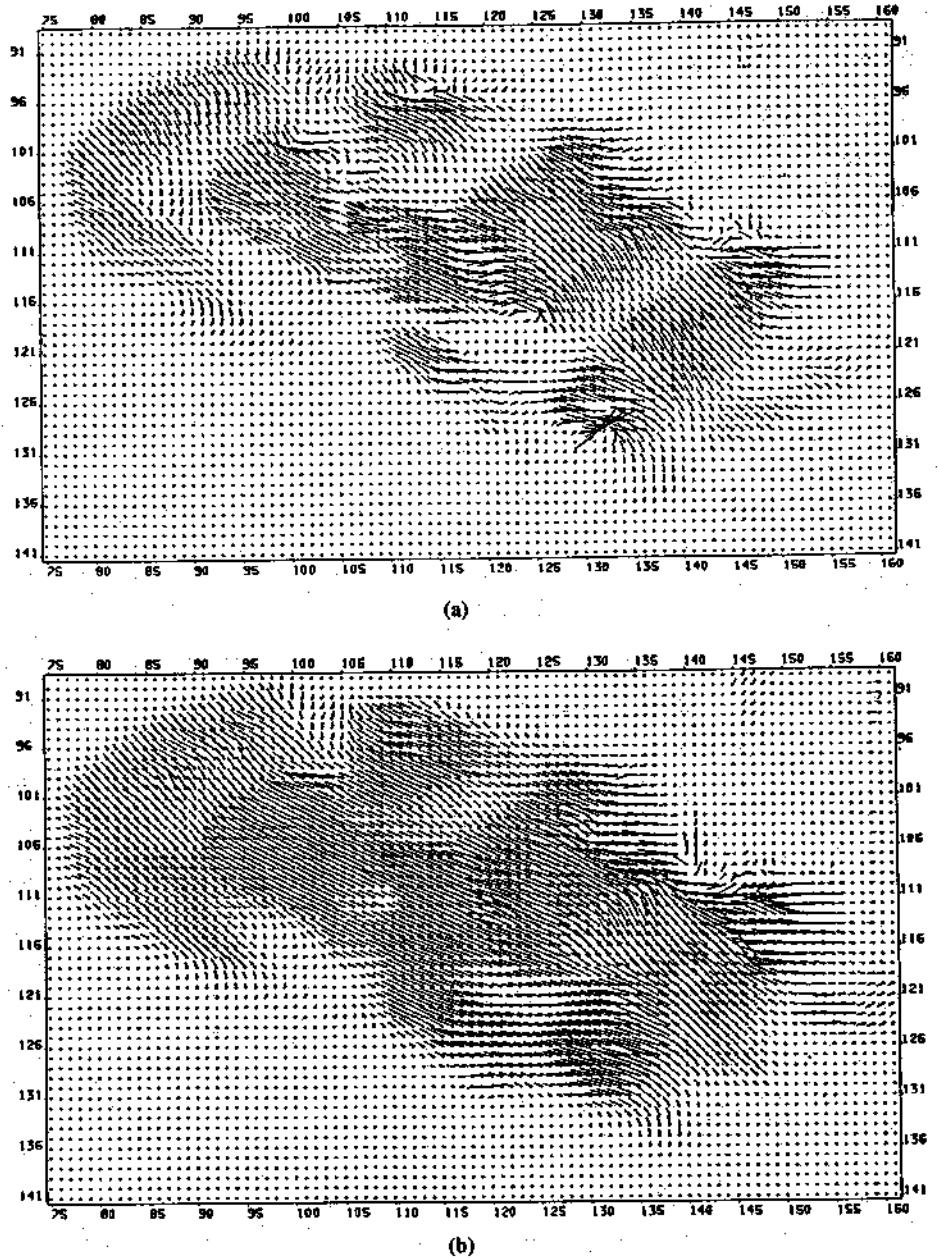


Fig. 13. Analogous to Fig. 11, but after iteration 10: (a) for 3×3 pixel operator window with $\alpha^2 = 50$; $c = 2$; $\gamma = 12.5$; (b) for 5×5 pixel operator window with $\alpha^2 = 10$; $c = 1$; $\gamma = 12.5$.

Introduction of these approximations into (35) yields

$$-[g2(x) - g1(x - u_0) + (\nabla g1(x - u_0))^T Du] \\ \cdot \frac{\partial g1(x - u_0)}{\partial x} + \alpha^2 \operatorname{trace} \left(C^{-1} \begin{pmatrix} u_{0,xx} & u_{0,xy} \\ u_{0,yx} & u_{0,yy} \end{pmatrix} \right) = 0 \quad (\text{A1-3a})$$

$$-[g2(x) - g1(x - u_0) + (\nabla g1(x - u_0))^T Du] \\ \cdot \frac{\partial g1(x - u_0)}{\partial y} + \alpha^2 \operatorname{trace} \left(C^{-1} \begin{pmatrix} v_{0,xx} & v_{0,xy} \\ v_{0,yx} & v_{0,yy} \end{pmatrix} \right) = 0. \quad (\text{A1-3b})$$

This system of equations can be rewritten as a matrix equation where for simplicity of notation the argument $x - u_0$ is suppressed for all derivatives $g1(x - u_0)$:

$$(\nabla g1)(\nabla g1)^T Du = -[g2(x) - g1(x - u_0)]\nabla g1 \\ + \alpha^2 \left[\operatorname{trace} \left(C^{-1} \begin{pmatrix} u_{0,xx} & u_{0,xy} \\ u_{0,yx} & u_{0,yy} \end{pmatrix} \right) \right. \\ \left. + \alpha^2 \operatorname{trace} \left(C^{-1} \begin{pmatrix} v_{0,xx} & v_{0,xy} \\ v_{0,yx} & v_{0,yy} \end{pmatrix} \right) \right] \quad (\text{A1-4})$$

It is not possible to solve (A1-4) immediately for Du , because the coefficient matrix of Du consists of the outer

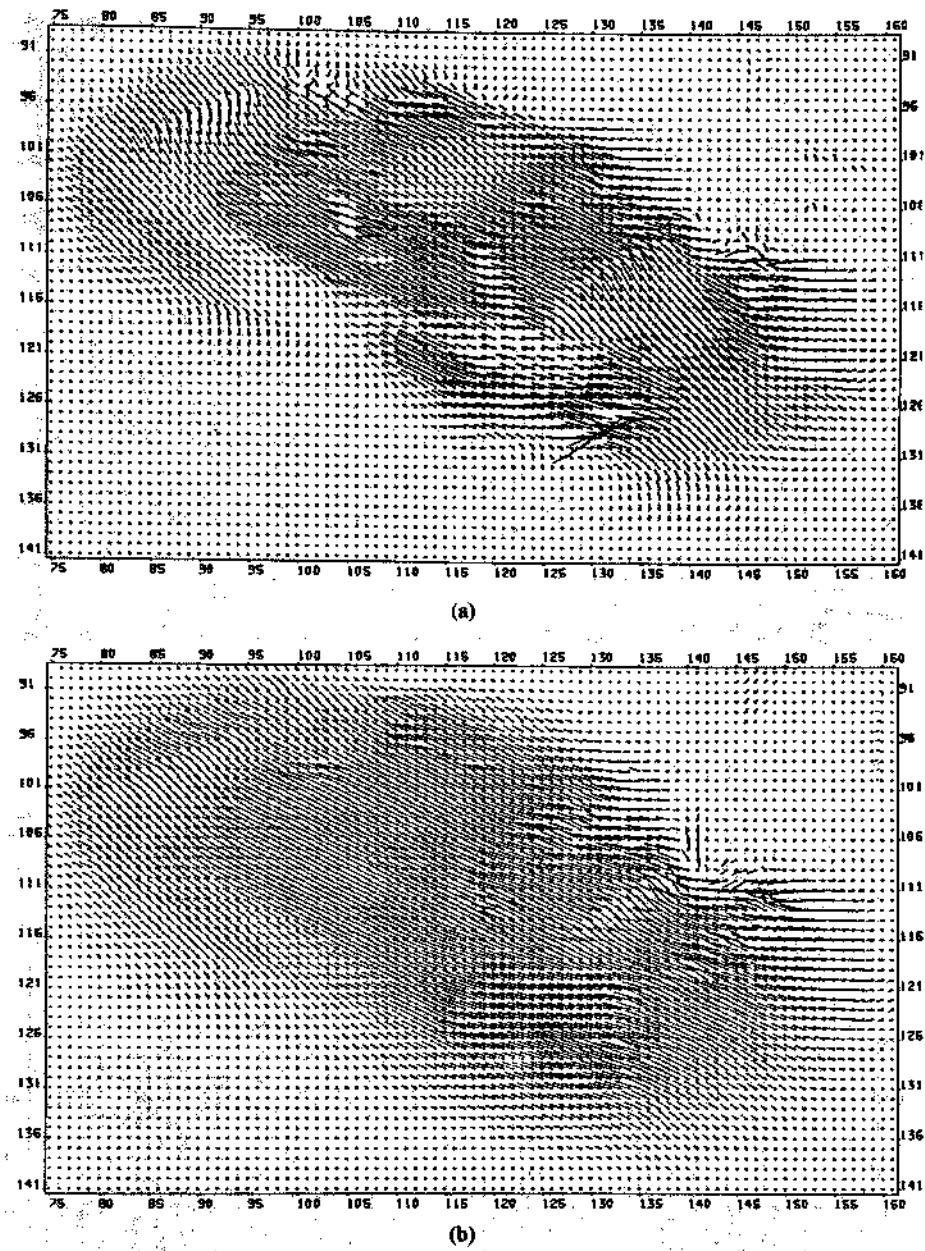


Fig. 14. Analogous to Fig. 11, but after iteration 30: (a) for 3×3 pixel operator window with $\alpha^2 = 50$; $c = 2$; $\gamma = 12.5$; (b) for 5×5 pixel operator window with $\alpha^2 = 40$; $c = 1$; $\gamma = 12.5$.

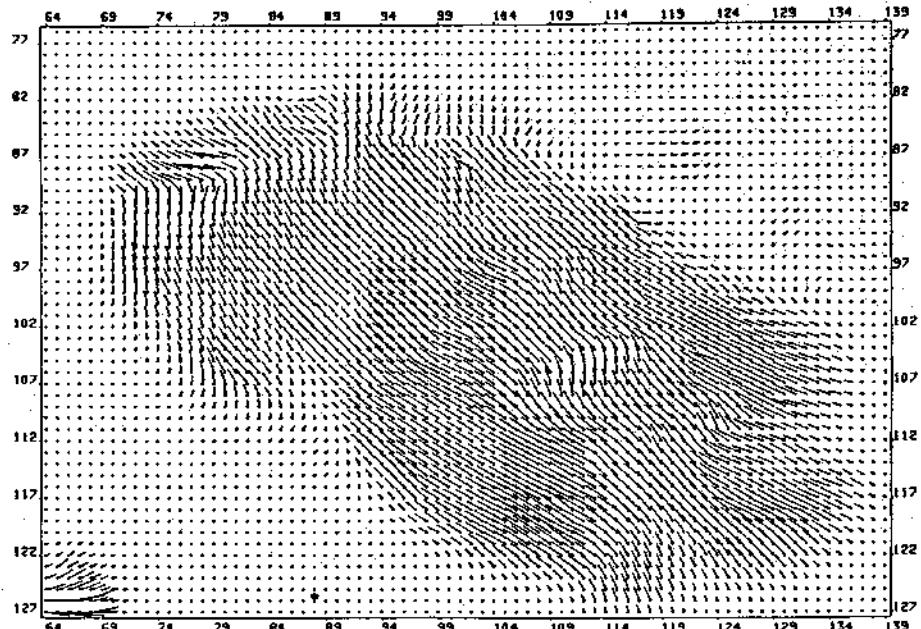
product of a vector $\nabla g1$ with itself and, therefore, has only rank one. At this point, the discussion of the matrix $B^T W_G B$ in Section II-A can be of help. It essentially represents a weighted product of gradients averaged over the image window under consideration. Since the derivatives of $g1$ as well as of u or v have eventually to be determined by applying some digital derivative operator to an environment around the position $x - u_0$, it appears plausible to average the coefficient matrix $(\nabla g1)(\nabla g1)^T$ over this environment. Assuming that this environment is small enough so that the variation of the gradient may be taken into account by its first order Taylor expansion, we obtain:

$$\nabla g1(x - u_0 + \xi) = \nabla g1(x - u_0) + (\nabla \nabla g1(x - u_0))\xi \quad (A1-5)$$

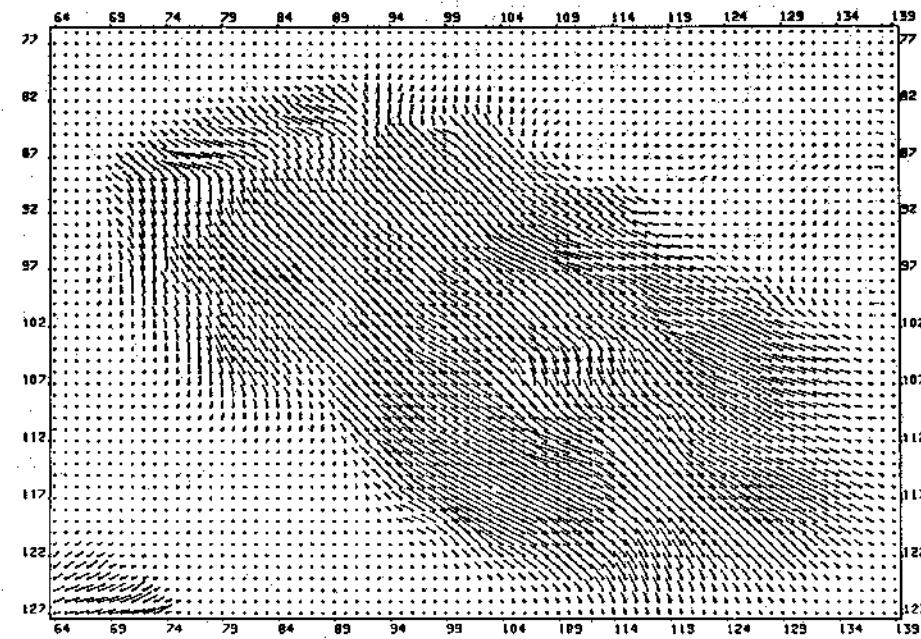
$$\begin{aligned} & (\nabla g1(x - u_0 + \xi))(\nabla g1(x - u_0 + \xi))^T \\ &= (\nabla g1)(\nabla g1)^T + \overline{\xi^2}(\nabla \nabla g1)(\nabla \nabla g1)^T \end{aligned} \quad (A1-6)$$

where the overbar indicates the averaging over all positions ξ in the environment around $x - u_0$. If we equate the weight factor b^2 in the matrix C of (30) with the average value of ξ^2 , the averaging process transforms the outer product matrix $(\nabla g1)(\nabla g1)^T$ into the matrix C . Equation (A1-4) could thus be written as

$$\begin{aligned} CDu = & -[\overline{g2(x)} - g1(x - u_0)]\nabla g1 \\ & + \alpha^2 \left[\frac{\text{trace}(C^{-1}\nabla \nabla u_0)}{\text{trace}(C^{-1}\nabla \nabla v_0)} \right]. \end{aligned} \quad (A1-7)$$



(a)



(b)

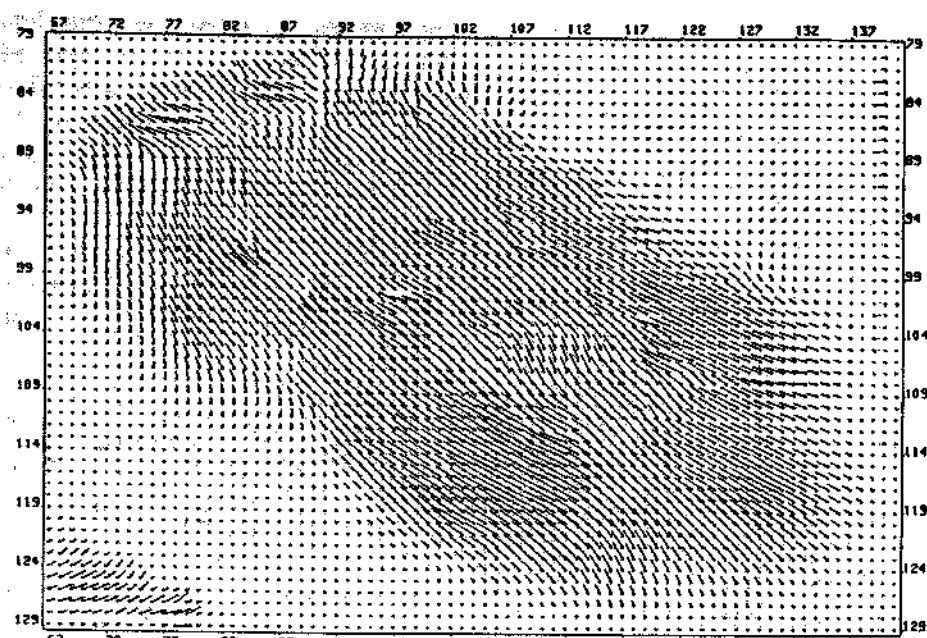
Fig. 15. (a) Result after 12 iterations for a window indicated by Fig. 1(f) from frames 20 and 21 with a uniformly vanishing starting value $u_0(x) = 0$; (b) result after four additional iterations for the corresponding windows from frames 21 and 22 with the result from (a) as a starting value; (c) analogously for frames 22 and 23 after two additional iterations with the result from (b) as starting value.

The calculation of ∇u and ∇v is accomplished by application of the operators described by Beaudet [1] to the digital representation of u and v . This implies some averaging across the operator window. The overbar has been omitted, therefore, atop the second term on the right-hand side of (A1-7) because the averaging process is essentially implied by the calculation of this term. The matrix C is supposed to be essentially constant across the

window needed to determine the partial derivatives of g_1 and u as well as v at location $x = u_0$. ■

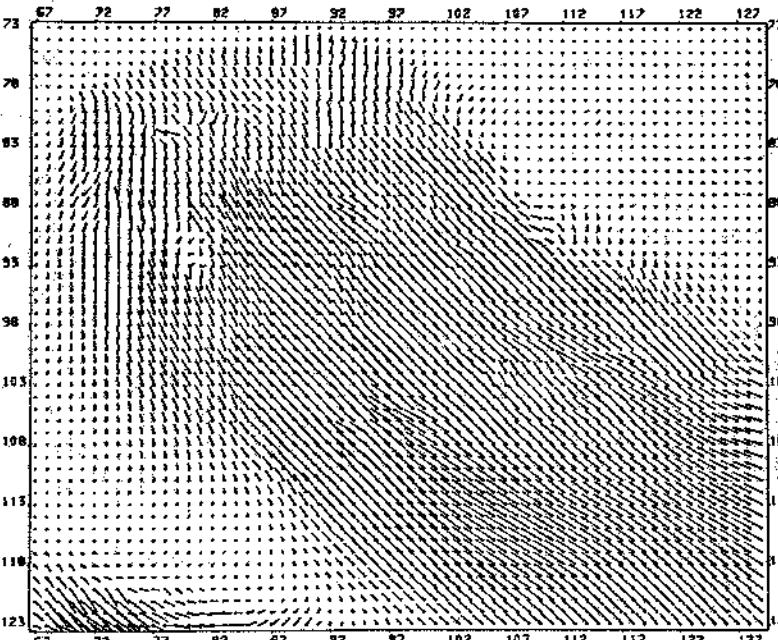
APPENDIX 2 DERIVATION OF (38)

In order to study the influence of Du on the estimation of partial derivatives of the displacement vector field $u(x)$, the following example shows the operators employed for



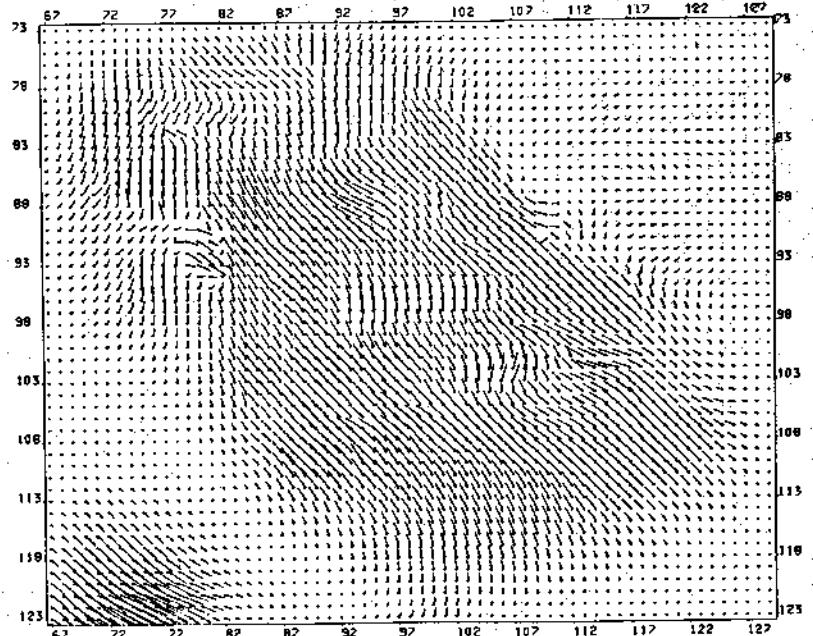
(c)

Fig. 15. (Continued.)

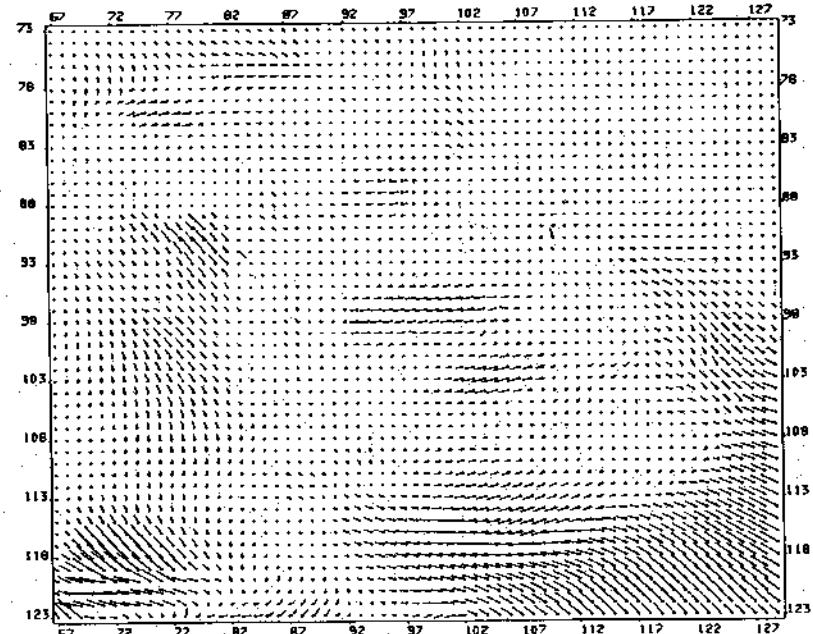


(a)

Fig. 16. (a) The result after two iterations for the window indicated by Fig. 1(g) from frame 30 and the preceding one, frame 29, taking the result of a propagation of starting values from frame 22 and 23 as given in Fig. 15(c) through frames 28 and 29 as initial values for the displacement vector field; (b) result after 30 iterations for the same window as in (a), but using $u_0(x) = 0$ as the starting value; (c) difference between the results shown in (a) and (b) in order to illustrate the dependency on the initial displacement vector field. Since the shadowy area behind the car is of fairly homogeneous gray value, both the vanishing displacement estimate as well as the displacement estimate of the moving car are solutions which are acceptable in principle. A gradual fading of the displacement estimate with diminishing contributions from the moving shadow seems to offer the intuitively most plausible solution.



(b)



(c)

Fig. 16. (Continued.)

a 5×5 pixel window. These operators are given by Beaudet [1] in (row, column)-format whereas they are represented here in a regular (x, y) -format:

$$\left(\frac{\partial^2}{\partial x^2} \right) = \left(\frac{\partial^2}{\partial y^2} \right)^T = \frac{1}{35} = \begin{pmatrix} 2 & -1 & -2 & -1 & 2 \\ 2 & -1 & -2 & -1 & 2 \\ 2 & -1 & -2 & -1 & 2 \\ 2 & -1 & -2 & -1 & 2 \\ 2 & -1 & -2 & -1 & 2 \end{pmatrix}$$

(A2-1a)

$$\left(\frac{\partial^2}{\partial x \partial y} \right) = \frac{1}{100} = \begin{pmatrix} -4 & -2 & 0 & 2 & 4 \\ -2 & -1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & -1 & -2 \\ 4 & 2 & 0 & -2 & -4 \end{pmatrix}$$

(A2-1b)

It is seen that the central pixel contributes to the second partial derivatives with respect to x as well as to y with a weight factor $-2/35$. The central pixel does not contrib-

ute to the mixed second partial derivative. As can be seen, for example, from the Appendix of [15], this latter aspect is a general property for all odd mask sizes of operators derived based on the following assumption: the functional variation within the operator mask can be approximated by a bivariate second-order polynomial in the image plane coordinates.

Exploiting these properties of operators for the computation of second order partial derivatives, the substitution of $u_0 + Du$ for u at the center pixel has the following effect:

$$\text{trace} \left(C^{-1} \begin{pmatrix} u_{xx} & u_{xy} \\ u_{xy} & u_{yy} \end{pmatrix} \right) = \text{trace} \left(C^{-1} \begin{pmatrix} u_{0xx} & u_{0xy} \\ u_{0xy} & u_{0yy} \end{pmatrix} \right) - mDu \text{trace} C^{-1} \quad (\text{A2-2a})$$

$$\text{trace} \left(C^{-1} \begin{pmatrix} v_{xx} & v_{xy} \\ v_{xy} & v_{yy} \end{pmatrix} \right) \approx \text{trace} \left(C^{-1} \begin{pmatrix} v_{0xx} & v_{0xy} \\ v_{0xy} & v_{0yy} \end{pmatrix} \right) - mDv \text{trace} C^{-1} \quad (\text{A2-2b})$$

where m stands for the magnitude value of the weight factor at the center pixel position in the operator mask for $\partial^2/\partial x^2$. In the case of a 5×5 operator mask, this results in the value $m = 2/35$.

Taking into account these effects of the approximation implied by the substitution of $u_0 + Du$ for u into (35), the derivation follows closely the one discussed in Appendix 1. In analogy to (A1-3) we obtain

$$-[g2(x) - g1(x - u_0) + (\nabla g1)^T Du] \frac{\partial g1}{\partial x} + \alpha^2 [\text{trace}(C^{-1} \nabla \nabla u_0) - mDu \text{trace} C^{-1}] = 0 \quad (\text{A2-3a})$$

$$-[g2(x) - g1(x - u_0) + (\nabla g1)^T Du] \frac{\partial g1}{\partial y} + \alpha^2 [\text{trace}(C^{-1} \nabla \nabla v_0) - mDv \text{trace} C^{-1}] = 0. \quad (\text{A2-3b})$$

This can be transformed into the following matrix equation:

$$\begin{bmatrix} (\nabla g1)(\nabla g1)^T + \alpha^2 m \text{trace} C^{-1} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \text{trace}(C^{-1} \nabla \nabla u_0) \\ \text{trace}(C^{-1} \nabla \nabla v_0) \end{bmatrix} Du = -[g2(x) - g1(x - u_0)] \nabla g1 \quad (\text{A2-4})$$

The averaging process will then yield—using the derivation expressed by (A1-5), (A1-6)—with I denoting a 2×2 -unit matrix:

$$\begin{aligned} & [(\nabla g1)(\nabla g1)^T + b^2 (\nabla \nabla g1)(\nabla \nabla g1)^T \\ & + \alpha^2 m \text{trace} C^{-1} I] Du \\ & = -[g2(x) - g1(x - u_0)] \nabla g1 \\ & + \alpha^2 \begin{bmatrix} \text{trace}(C^{-1} \nabla \nabla u_0) \\ \text{trace}(C^{-1} \nabla \nabla v_0) \end{bmatrix} \end{aligned} \quad (\text{A2-5})$$

APPENDIX 3

DERIVATION OF (40)

We may assume that the local coordinate system is aligned with the principal curvature directions of the gray value distribution within this window, i.e., $g1_{xy} = 0$. We further assume that $u_0(x)$ is still zero throughout the image window under consideration. Using these conventions, (38) reduces to

$$\begin{aligned} & [C + I\alpha^2 m \text{trace} C^{-1}] Du \\ & = -[g2(x) - g1(x - u_0)] \nabla g1 \end{aligned} \quad (\text{A3-1})$$

because $u_0(x) = 0$ implies $\nabla \nabla u_0 = \nabla \nabla v_0 = 0$.

The explicit solution of this system of two equations with the two unknowns Du and Dv can be written in a more compact form if the following identity is used which can be easily verified:

$$\begin{aligned} & [C + I\alpha^2 m \text{trace} C^{-1}]^{-1} \\ & = \frac{C^{-1} \det C + I\alpha^2 m \text{trace} C^{-1}}{\det(C + I\alpha^2 m \text{trace} C^{-1})} \end{aligned} \quad (\text{A3-1})$$

In order to evaluate the right-hand side of (39) we proceed in analogy to the transition between (A1-5) and (A1-6) (see also [15]). The image window is assumed to be square with sides parallel to the axes of the local coordinate system ξ centered in the window. We use (A1-5) to make explicit how the gradient $\nabla g1(x - u_0 + \xi)$ depends on ξ . We thus obtain

$$\begin{aligned} & [g2(x + \xi) - g1(x - u_0 + \xi)] \nabla g1(x - u_0 + \xi) \\ & = [g2(x + \xi) - g1(x - u_0 + \xi)] \nabla g1(x - u_0) \\ & + [g2(x + \xi) - g1(x - u_0 + \xi)] \\ & \cdot (\nabla \nabla g1(x - u_0)) \xi. \end{aligned} \quad (\text{A3-2})$$

If this equation is averaged over the image window, the first term on the right-hand side will simply result in

$$[\overline{g2(x)} - \overline{g1(x - u_0 + \xi)}] \nabla g1.$$

The second term has to be treated in more detail. We develop the square bracket of the second term up to first order in components of ξ :

$$\begin{aligned} & [g2(x + \xi) - g1(x - u_0 + \xi)] (\nabla \nabla g1) \xi = [\overline{g2(x)} \\ & + (\nabla g2)^T \xi - \overline{g1(x - u_0)} - (\nabla g1)^T \xi] (\nabla \nabla g1) \xi. \end{aligned} \quad (\text{A3-3})$$

Since the window is symmetric with respect to the components of ξ over which we have to average, all terms containing odd powers of components of ξ vanish in this process. We retain

$$\begin{aligned} & [g2(x) - g1(x - u_0)] \nabla g1(x - u_0) \\ &= [\overline{g2(x)} - \overline{g1(x - u_0)}] \nabla g1 + \overline{\xi^2} \left(\begin{array}{l} (g2_x - g1_x) g1_{xx} \\ (g2_y - g1_y) g1_{yy} \end{array} \right) \quad (A3-4) \end{aligned}$$

where the derivatives of $g2$ have been taken at location x and those of $g1$ at location $x - u_0$. Combining (39), (A3-1), and (A3-4) yields

$$\begin{aligned} Du &= \frac{-1}{\det(C + I\alpha^2 m \operatorname{trace} C^{-1})} \cdot (C^{-1} \det C + I\alpha^2 m \operatorname{trace} C^{-1}) \\ &\quad \cdot \left(\begin{array}{l} [\overline{g2(x)} - \overline{g1(x - u_0)}] g1_x + \overline{\xi^2} (g2_x - g1_x) g1_{xx} \\ [\overline{g2(x)} - \overline{g1(x - u_0)}] g1_y + \overline{\xi^2} (g2_y - g1_y) g1_{yy} \end{array} \right) \quad (A3-5) \end{aligned}$$

It should be remembered that due to the alignment of the local coordinate system with the principal curvature directions of $g1$, the mixed second partial derivative $g1_{xy}$ vanishes.

We may now specialize (A3-5) to the case of a gray value corner in frame 1 at location $x - u_0$. Then, according to (1) we have:

$$\begin{aligned} g1_x &= \text{maximum} \neq 0 & g1_{xx} &= 0 \\ g1_y &= 0 & g1_{yy} &= \text{maximum} \neq 0. \end{aligned}$$

Insertion of these values into equation (A3-5) yields:

$$C^{-1} = \frac{1}{b^2 g1_x^2 g1_y^2} \begin{pmatrix} b^2 g1_y^2 & 0 \\ 0 & g1_x^2 \end{pmatrix} \quad (A3-6a)$$

$$\operatorname{trace} C^{-1} = \frac{g1_x^2 + b^2 g1_y^2}{b^2 g1_x^2 g1_y^2} \quad (A3-6b)$$

$$[C^{-1} \det C + I\alpha^2 m \operatorname{trace} C^{-1}] = \begin{pmatrix} b^2 g1_y^2 + \alpha^2 m \frac{g1_x^2 + b^2 g1_y^2}{b^2 g1_x^2 g1_y^2} & 0 \\ 0 & g1_x^2 + \alpha^2 m \frac{g1_x^2 + b^2 g1_y^2}{b^2 g1_x^2 g1_y^2} \end{pmatrix} \quad (A3-6c)$$

$$\det(C + I\alpha^2 m \operatorname{trace} C^{-1}) = \det(C^{-1} \det C + I\alpha^2 m \operatorname{trace} C^{-1}) \quad (A3-6d)$$

We thus obtain by setting

$$b^2 = \overline{\xi^2}$$

[see discussion following (A1-6)]:

$$Du = -\frac{\overline{g2(x)} - \overline{g1(x - u_0)}}{g1_x} \cdot \frac{1}{1 + \alpha^2 m \frac{1 + b^2 g1_y^2 / g1_x^2}{b^2 g1_x^2 g1_y^2}} \quad (A3-7a)$$

$$Dv = -\frac{g2_y}{g1_{yy}} \cdot \frac{1}{1 + \alpha^2 m \frac{1 + g1_x^2 / (b^2 g1_y^2)}{b^2 g1_x^2 g1_y^2}}. \quad (A3-7b)$$

APPENDIX 4 EVALUATION OF THE WEIGHT MATRIX NORMALIZED BY TRACE (F)

Instead of (31) we investigate the minimization of

$$\iint dxdy \left\{ [g2(x) - g1(x - u)]^2 + \alpha^2 \operatorname{trace} \left[(\nabla u)^T \frac{F}{\operatorname{trace} F} (\nabla u) \right] \right\} \Rightarrow \min. \quad (41)$$

The analogy to the derivation resulting in (A2-2a) now yields

$$\operatorname{trace} \left[\frac{F}{\operatorname{trace} F} (\nabla \nabla u) \right] = \operatorname{trace} \left[\frac{F}{\operatorname{trace} F} (\nabla \nabla u_0) \right] - mDu \frac{\operatorname{trace} F}{\operatorname{trace} F} \quad (A4-1)$$

and analogously for (A2-2b). We thus obtain instead of (A2-4):

$$\begin{aligned} & [(\nabla g1)(\nabla g1)^T + \alpha^2 m I] Du \\ & = -[g2(x) - g1(x - u_0)] \nabla g1 \\ & + \frac{\alpha^2}{\text{trace } F} \left[\frac{\text{trace } (\nabla \nabla u_0)}{\text{trace } (\nabla \nabla v_0)} \right]. \end{aligned} \quad (A4-2)$$

In analogy to (A2-5) or (38) we obtain

$$\begin{aligned} [C + \alpha^2 m I] Du & = -[g2(x) - g1(x - u_0)] \nabla g1 \\ & + \frac{\alpha^2}{\text{trace } F} \left[\frac{\text{trace } (\nabla \nabla u_0)}{\text{trace } (\nabla \nabla v_0)} \right] \end{aligned} \quad (A4-3)$$

and in analogy to (A3-7) or (40):

$$Du = \frac{g2(x) - g1(x - u_0)}{g1_x} \cdot \frac{1}{1 + \frac{\alpha^2 m}{g1_x^2}} \quad (A4-4a)$$

$$Du = \frac{g2_y}{g1_{yy}} \cdot \frac{1}{1 + \frac{\alpha^2 m}{b^2 g1_{yy}^2}} \quad (A4-4b)$$

In analogy to the discussion following (40) we "unbend" the gray value corner. With typical values along an almost straight line gray value transition— $g1_x$ around 30 and $b \cdot g1_{yy}$ around 1—(A4-4) yields a correction factor of $1/(1 + \alpha^2 m/900)$ for Du and $1/(1 + \alpha^2 m)$ for Dv .

With such values for the correction factors, the estimate for Du in (A4-4a) essentially reflects the contribution of the first term on the right hand side of equation (A4-3), namely the well known displacement estimate parallel to the gradient, usually written in the form $-g_x/g_x$. The estimate for Dv in (A4-4b) depends under these conditions more strongly on the value of α^2 . Since this value of α^2 also influences the contribution due to nonzero second derivatives of u_0 and v_0 for other than the initial iteration where $u_0 = 0$ and thus $\nabla \nabla u_0 = \nabla \nabla v_0 = 0$, the choice of α^2 has to be studied with great care.

If we take into account the second term on the right-hand side of (A4-3), we obtain instead of (A4-4) the following result for a gray value corner:

$$\begin{aligned} Du & = \frac{g2(x) - g1(x - u_0)}{g2_x} \cdot \frac{1}{1 + \frac{\alpha^2 m}{g1_x^2}} \\ & + \alpha^2 \frac{b^2 g1_{yy}^2 u_{0x} + g1_x^2 u_{0y}}{(g1_x^2 + b^2 g1_{yy}^2)(g1_x^2 + \alpha^2 m)} \end{aligned} \quad (A4-5a)$$

$$\begin{aligned} Du & = \frac{g2_y}{g1_{yy}} \cdot \frac{1}{1 + \frac{\alpha^2 m}{b^2 g1_{yy}^2}} \\ & + \alpha^2 \frac{b^2 g1_x^2 v_{0x} + g1_x^2 v_{0y}}{(g1_x^2 + b^2 g1_{yy}^2)(b^2 g1_{yy}^2 + \alpha^2 m)} \end{aligned} \quad (A4-5b)$$

ACKNOWLEDGMENT

We thank B. Radig for his help in sustaining our long-distance cooperation and the Deutsche Forschungsgemeinschaft for partial support of these investigations. The idea to compare the results from different initializations as presented in Figs. 16(b) and (c) was suggested by S. Ullman during a discussion about the "oriented smoothness" constraint. We are grateful to R. Kories, T. Krämer, and the referees for suggestions to improve the style of this paper.

REFERENCES

- [1] P. R. Beaudet, "Rotationally invariant image operators," in *Proc. Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, 1978, pp. 579-583.
- [2] N. Cornelius and T. Kanade, "Adapting optical flow to measure object motion in reflectance and X-ray image sequences," in *Proc. ACM SIGGRAPH/SIGART Interdisciplinary Workshop on Motion: Representation and Perception*, Toronto, Ont., Canada, Apr. 4-6, 1983, pp. 50-58.
- [3] L. S. Davis, Z. Wu, and H. Sun, "Contour-based motion estimation," *Comput. Vision, Graphics, Image Processing*, vol. 23, pp. 313-326, 1983.
- [4] L. Dreschner and H.-H. Nagel, "Volumetric model and 3D trajectory of a moving car derived from monocular TV frame sequences of a street scene," in *Proc. Int. Joint Conf. Artificial Intell.*, Vancouver, B.C., Canada, Aug. 1981, pp. 692-697.
- [5] —, "Volumetric model and 3D trajectory of a moving car derived from monocular TV frame sequences of a street scene," *Comput. Graphics Image Processing*, vol. 20, pp. 199-228, 1982 (complemented and extended version of the contribution to IJCAI-81).
- [6] —, "On the selection of critical points and local curvature extrema of region boundaries for interframe matching," in *Proc. Int. Conf. Pattern Recognition*, Munich, 1982, pp. 542-544; also in T. S. Huang, Ed., *Image Sequence Processing and Dynamic Scene Analysis*, NATO ASI Series F2, Berlin: Springer-Verlag, 1983, pp. 457-470.
- [7] E. C. Hildreth, "The integration of motion information along contours," in *Proc. Workshop Comput. Vision: Representation and Control*, Rindge, NH, Aug. 23-25, 1982, pp. 83-91.
- [8] —, "Computing the velocity field along contours," in *Proc. ACM SIGGRAPH/SIGART Interdisciplinary Workshop on Motion: Representation and Perception*, Toronto, Ont., Canada, Apr. 4-6, 1983, pp. 26-32.
- [9] —, "The measurement of visual motion," Ph.D. dissertation, Dep. Elec. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, Aug. 1983.
- [10] —, "Computations underlying the measurement of visual motion," *Artificial Intell.*, vol. 23, pp. 309-354, 1984.
- [11] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intell.*, vol. 17, pp. 185-203, 1981.
- [12] L. Kitchen and A. Rosenfeld, "Gray-level corner detection," Comput. Sci. Center, Univ. Maryland, College Park, Tech. Rep. 887, Apr. 1980.
- [13] —, "Gray-level corner detection," *Pattern Recognition Lett.*, vol. 1, pp. 95-102, 1982.
- [14] H.-H. Nagel, "On change detection and displacement vector estimation in image sequences," *Pattern Recognition Lett.*, vol. 1, pp. 55-59, 1982.
- [15] —, "Displacement vectors derived from second order intensity variations in image sequences," *Comput. Vision, Graphics, Image Processing*, vol. 21, pp. 85-117, 1983.
- [16] —, "On the estimation of dense displacement vector fields from image sequences," in *Proc. ACM SIGGRAPH/SIGART Interdisciplinary Workshop on Motion: Representation and Perception*, Toronto, Ont., Canada, Apr. 4-6, 1983, pp. 59-65.
- [17] —, "Constraints for the estimation of displacement vector fields from image sequences," in *Proc. Int. Joint Conf. Artificial Intell.*, Karlsruhe, West Germany, 1983, pp. 945-951.
- [18] —, "Spatio-temporal modeling based on image sequences," in *Proc. Int. Symp. Image Processing and Its Applications*, Univ. Tokyo, Inst. Industrial Sci., Jan. 18-21, 1984; also in *Progress in Image Processing*, M. Onoe, Ed., Tokyo: Univ. Tokyo, pp. 222-252.
- [19] —, "Recent advances in image sequence analysis," in *Proc. Pre-*

- mier Colloque Image—Traitement, Synthèse, Technologie et Applications*, Biarritz, France, May 21–25, 1984, pp. 545–558.
- [20] H.-H. Nagel and W. Enkelmann, "Investigations of second order gray value variations to estimate corner point displacements," in *Proc. Int. Conf. Pattern Recognition*, Munich, West Germany, Oct. 19–22, 1982, pp. 768–773.
- [21] —, "Iterative estimation of displacement vector fields from TV-frame sequences," in *Proc. Second European Signal Processing Conf. (EUSIPCO-83)*, H. W. Schüssler, Ed., Erlangen, West Germany, Sept. 12–16, 1983, pp. 299–302.
- [22] —, "Towards the estimation of displacement vector fields by 'oriented smoothness' constraints," in *Proc. Int. Conf. Pattern Recognition*, Montreal, P.Q., Canada, July 30–Aug. 2, 1984, pp. 6–8.
- [23] —, "Berechnung von Verschiebungsvektorfeldern in Bildbereichen mit linienhaften oder partiell homogenen Grauwertverteilungen," in *Proc. Deutsche Arbeitsgemeinschaft für Mustererkennung*, W. Kropatsch, Ed., Graz, Austria, Oct. 2–4, 1984; also *Informatik-Fachberichte 87*. Berlin: Springer-Verlag, 1984, pp. 154–160.
- [24] Z. Wu, H. Sun, and L. S. Davis, "Determining velocities by propagation," in *Proc. Int. Conf. Pattern Recognition*, Munich, West Germany, Oct. 19–22, 1982, pp. 1147–1149.
- [25] M. Yachida, "Determining velocity by spatio-temporal neighborhoods from image sequences," *Comput. Vision, Graphics, Image Processing*, vol. 21, pp. 262–279, 1983; also in *Proc. IJCAI-81*, pp. 716–718.
- [26] O. A. Zuniga and R. M. Haralick, "Corner detection using the facet model," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, 1983, pp. 30–37.

Director of the Fraunhofer-Institut für Informations- und Datenverarbeitung at Karlsruhe in a joint appointment as Full Professor (pattern recognition and digital image processing) at the Fakultät für Informatik der Universität Karlsruhe (TH). Since 1971, he has been interested in the evaluation of image sequences, especially TV-frame sequences. In addition, his interests include the implementation and use of higher level programming languages for the realization of image analysis systems.

Dr. Nagel is Associate Editor of *Computer Vision, Graphics, and Image Processing*, and a member of the Editorial Board of *Artificial Intelligence Journal*, as well as of *Pattern Recognition Letters*. He currently serves on the Advisory Board of *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* and *Future Generation Computer Systems*.

Hans-Helmut Nagel (M'77) received the Doctor degree in physics from the Universität Bonn, Bonn, West Germany, in 1964.

Subsequently, he spent 18 months at M.I.T. working on the automatic analysis of bubble chamber film, continuing this work at the Deutsche Elektronen-Synchrotron at Hamburg as well as at the Physikalische Institut der Universität Bonn from 1966 through 1971. In Fall 1971 he became Full Professor of Informatik (computer science) at the Universität Hamburg. Since 1983 he has been

Wilfried Enkelmann received the Dipl.Inform. degree and the Ph.D. degree in informatik (computer science) from the University of Hamburg, Hamburg, West Germany, in 1982 and 1985, respectively.

Since mid-1985 he has been with the Fraunhofer-Institut für Informations- und Datenverarbeitung in Karlsruhe, West Germany. His field of interest is computer vision and image sequence analysis.

Disparity Analysis of Images

STEPHEN T. BARNARD, MEMBER, IEEE, AND WILLIAM B. THOMPSON, MEMBER, IEEE

Abstract—An algorithm for matching images of real world scenes is presented. The matching is a specification of the geometrical disparity between the images and may be used to partially reconstruct the three-dimensional structure of the scene. Sets of candidate matching points are selected independently in each image. These points are the locations of small, distinct features which are likely to be detectable in both images. An initial network of possible matches between the two sets of candidates is constructed. Each possible match specifies a possible disparity of a candidate point in a selected reference image. An initial estimate of the probability of each possible disparity is made, based on the similarity of subimages surrounding the points. These estimates are iteratively improved by a relaxation labeling technique making use of the local continuity property of disparity that is a consequence of the continuity of real world surfaces. The algorithm is effective for binocular parallax, motion parallax, and object motion. It quickly converges to good estimates of disparity, which reflect the spatial organization of the scene.

Index Terms—Disparity, matching, motion, relaxation labeling, scene analysis, stereo.

I. INTRODUCTION

DIFFERENCES in images of real world scenes may be induced by the relative motion of the camera and the scene, by the relative displacement of two cameras, or by the motion of objects in the scene. The differences are important because they encode information that often allows a partial reconstruction of the three-dimensional structure of the scene from two-dimensional projections. When such differences occur between two images we say that there is a *disparity* between the two images, which we represent as a vector field mapping one image into the other. The determination of disparity has been called the *correspondence* problem [1], [2]. A contingent problem is the *interpretation* of disparity into meaningful statements about the scene, such as specifications of depth, velocity, and shape.

There is much evidence that disparity is important in human vision. Gibson discussed the nature of visual perception in a dynamic environment [3]. He argued that the visual stimulus is inherently dynamic and that the patterns of change in the stimulus are important sources for the perception of the spatial environment. Gibson described the patterns of optical flow that occur when the observer moves. He argued that binocular disparity and retinal motion are highly informative stimulus variables for spatial perception, and that along with other important visual phenomena, such as texture gradient and linear perspective, they interact with the kinesthetic "body

Manuscript received March 5, 1979; revised September 14, 1979. This work was supported in part by the National Science Foundation under Grant MCS-78-20780.

S. T. Barnard was with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455. He is now with SRI International, Menlo Park, CA 94025.

W. B. Thompson is with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

senses" in the perception of a stable, upright, three-dimensional world.

Julesz's experiments with random dot stereograms [4] support the contention that the human visual system is able to process differences between images. A stereogram is a pair of images that are recorded simultaneously by laterally separated sensors. A random dot stereogram is artificially constructed by shifting parts of an image of random dots to the left or right to form a second image. Both images consist of uncorrelated random dots, but there is a disparity relationship between the two which specifies the apparent relative depth of the shifted parts. People with normal stereo vision can easily achieve a binocular fusion of the two images and perceive the apparent relative depth of the various "surfaces." Even though each separate image of a random dot stereogram contains no depth information, people are able to perceive depth in the pair of images by measuring the disparity between the shifted areas.

Disparity analysis may be broadly defined as the determination of the geometric differences between two or more images of the same or similar scenes. The differences may be the result of binocular parallax, motion parallax, object motion, or some combination of these modes. The goal of the analysis is to assign disparities, which are represented as two-dimensional vectors in the image plane, to a collection of points in one of the images. Disparity analysis is useful for image understanding in several ways. There is information in a disparate pair of images that is difficult to find or even absent in any single image. This point is convincingly made by Julesz's random dot stereogram experiments. Disparity provides a way, independent of high-level knowledge, of determining the spatial relationships between points and surfaces in a scene. The objects in the scene may be completely unfamiliar, but their observed disparities will conform to precise rules that depend only on the location and velocity of objects in three-dimensional space. Disparity is therefore a very general property of images which may be used in a variety of situations. The measurement of depth and velocity will certainly be useful in many applications, but there is a more fundamental requirement in image understanding. A system for understanding dynamic scenes can use observed disparity to establish conceptual relationships between images that are invariant over several observations. Visual invariants can be used to predict future observations, to eliminate noise in any one observation, and in general to link several observations into one perceptually coherent whole.

II. MATCHING

Matching is a natural way to approach disparity analysis. Assigning disparity classifications to points in a sequence of images is equivalent to finding a matching between sets of

points from each image. Let $S_1 = \langle s_1^x, s_1^y \rangle$ and $S_2 = \langle s_2^x, s_2^y \rangle$ be points in images 1 and 2, respectively. These two points should be matched if and only if they are image plane projections of the same real world surface point. Matching S_1 with S_2 is the same as assigning to S_1 a disparity with respect to image 2 of $D_1^k = \langle s_1^x - s_2^x, s_1^y - s_2^y \rangle$, where $D_j^k(S)$ is a vector function defined on points of image j , which specifies the disparity of point S in image j with respect to image k .

A matching approach to disparity analysis must solve two problems. First, how are points selected for matching? It is clear that not all points can be matched with equal confidence because some are located in regions which lack identifying detail. Some points may not be matched at all because they may be visible in one image but not in another. To avoid ambiguous matches, it is advantageous to attempt to match only points which are easily distinguished from their neighbors. It is important to select only those points which are projections of distinct, precisely positioned local features on real world surfaces, such as spots, corners, and other small local forms. Interest operators sensitive to local variance, edges, and other properties of subimages can be used to choose potentially matchable points [5]-[9]. An alternative to this strategy is to partially segment each static image independently and then use properties of the segments, such as shape and color, as similarity variables [10], [11]. This is an attempt to match entire surfaces, not points.

The second problem in matching is to determine the basis for deciding which matches are correct. The matched points should have similar properties, of course, because they are both projections of the same surface point, but in many cases there will be ambiguity. Many studies have used cross correlation or mean-square difference as a measure of similarity. Typically, a small region in image 1 surrounding an interesting point is used as a template, and a search is made for the region of maximum similarity in image 2. Two problems with this approach are that the correct match may not be the one of maximum similarity due to noise or distortion (there are ambiguous matches), and the cost of searching a two-dimensional image is high. One way to avoid ambiguity is to increase the sensitivity of the similarity measurement. Levine, O'Handley, and Yagi [7] use an adaptive correlation window, the size of which varies inversely with the variance of the region surrounding the point. Mori, Kidode, and Asada [12] use a Gaussian-weighted correlation window to minimize the errors due to distortion of the extremities. They also vary the window size with ambiguity of the match and use a prediction/correction algorithm, modifying one image to fit the other according to a predicted matching and iteratively using the error of fit to improve the prediction. Stochastic matched filtering can reduce ambiguity by improving similarity detection in the presence of noise [13].

Several strategies have been used to limit search in a cross correlation approach. Studies of stereopsis use a fixed camera model to constrain the search to one dimension [5], [7], [8], [12]. Nevatia uses a series of progressive views to constrain disparity to small values [8]. This also reduces the chance of ambiguous matches and increases the sensitivity of the similarity measurement by minimizing distortion. Another strategy

is to use a coarse search to approximately locate the matching points, followed by a fine search to more accurately locate them [7], [14], [15]. Sequential similarity detection can be used for more efficient matching [16].

Many studies have used heuristics based on real world models to limit search and to resolve or avoid ambiguity. Julesz [4] and Gibson [3] observed that disparity varies continuously across unoccluded surfaces and discontinuously only at occluding edges. Marr and Poggio [17] used this property, which they call the adjacency constraint, in an iterative cooperative matching algorithm which fuses random dot stereograms. Levine *et al.* [7] use it to limit the range of search of proximate points, and Mori *et al.* [12] use it to avoid ambiguity by matching "well-contrasting" regions with high confidence and favoring these disparities for nearby points.

In the studies discussed above we can identify three properties of image pairs which can strongly influence disparity classification. The first, which we call discreteness, is a property of individual points. Discreteness is a measurement of the individual distinctness of a point, and is important for selecting good candidates for matching. The interest operators described above are good examples of discreteness measures. The second property, similarity, is a measurement of how closely two points resemble one another. Such measures are usually simple functions of small areas surrounding the points. The third property, consistency, is a measurement of how well a particular match (that is, a particular disparity classification) conforms to nearby matches. The three-dimensional spatial continuity of real world surfaces constrains the two-dimensional spatial distribution of disparity in the image plane. Disparity is discontinuous only at occluding edges. The continuity of disparity over most of the image can be used to avoid false matches based on similarity alone, by suppressing matches in the absence of supporting local evidence.

III. A LOCALLY PARALLEL MODEL FOR MATCHING

This section describes a computational model for analyzing disparity in a variety of real world situations. Object motion, stereo, and motion parallax modes of disparity are treated uniformly. Because the model has the locally parallel, globally sequential structure of a relaxation labeling algorithm, the notation in this section follows Rosenfeld, Hummel, and Zucker [18]. The general approach is similar to matching models proposed by Julesz [4] and Marr and Poggio [17] for the fusion of random dot stereograms and to a matching model proposed by Ullman [2] for retinal motion.

The theory behind the locally parallel model is that the discreteness, similarity, and consistency properties can interact to enhance the overall performance and rigor of a matching algorithm. The use of multiple properties reduces the chance of error by minimizing the dependence on any one property. The consistency property allows the most obvious classifications to improve the analysis of the more difficult ones. The discreteness property is used to minimize expensive searching. Sets of candidate matching points are selected from each image, and searching is done between these two relatively sparse sets instead of between the two gray-valued images. The emphasis throughout the formulation of the model has

been on simplicity. We will demonstrate the effectiveness of this matching approach in even a minimal system.

The first step is to find the points in the two images which will be candidates for matching. Matchable points should locate small discrete local features such as spots and corners which are likely to be detectable in both images. They should be the centers of highly variable areas, and furthermore the variance should be high in all directions. (If the variance is high in general but low along one direction, as would be the case with a straight line, for example, the point will not be easily distinguishable from its neighbors along that direction). A very simple interest operator described by Moravec [19] is effective in selecting these matchable points. The sums of the squares of the differences of pixels in four directions (horizontal, vertical, and the two diagonals) are computed over a small area (5×5 areas are used in all examples presented here). The initial value of the interest operator for the center point is the minimum of these variances. A point will have a high initial value only if there is a high variance in all four directions. The final values of the interest operator are obtained by suppressing (i.e., setting to zero) the initial values of all but the local maxima. Any point with a nonzero final interest value must be located in an area with a high variance in all four directions, and the initial value of the point must be greater than that of any of the point's neighbors.

The interest operator is applied independently to each image. Points with locally maximal but very small interest values are rejected by thresholding. The selection of the threshold is not critical. In the examples shown in this paper it was set to give a reasonable number of points, expressed as a percentage of the total number of pixels in each image.

After the two sets of candidate points are found the next step is to construct a set of possible matches. Ideally, we would like to match each candidate point from image 1 with exactly one candidate from image 2 such that the two points are image plane projections of the same real world point. However, we can realistically expect to find valid matches for only some of the points of image 1 because the interest operator does not perform perfectly, and because some points may be occluded, shadowed, or not visible in image 2 for some reason. An initial set of possible matches is constructed by pairing each candidate point from image 1 with every candidate from image 2 within some maximum distance of the (x, y) location of the point in image 1. This distance r is the maximum detectable disparity in the x or y direction. The set of possible matches is organized as a collection of "nodes" $\{a_i\}$, one node for each candidate point from image 1. Associated with each node a_i is a tuple (x_i, y_i) which is the location of the point in image 1, and a set of labels L_i which represents possible disparities that may be assigned to the point. Each label in L_i is either a disparity vector (l_x, l_y) , where l_x and l_y are integers in $[-r, r]$, or it is a distinguished label l^* denoting "undefined disparity." A node a_i has undefined disparity if point (x_i, y_i) in image 1 does not correspond to any candidate point in image 2. Every label set L_i must initially contain the element l^* . The point (x_i, y_i) in image 1 is tentatively matched to a point at (x'_i, y'_i) in image 2 by entering a label $l = (x'_i - x_i, y'_i - y_i)$ into L_i . Note that not every vector with integral coordinates in the square of side $2r + 1$ need be represented in the label set of all possible matches for a node, but that the undefined disparity is always represented.

For every node a_i we want to associate with every label $l = (l_x, l_y)$ in L_i a number $p_i(l)$ which we can interpret as an estimate of the probability that point (x_i, y_i) in image 1 has disparity l . This requires that $p_i(l)$ be in $[0, 1]$ and $\sum_l p_i(l) = 1$. These probability estimates will be successively improved by applying the consistency property. If relatively many nearby points have a high probability of having disparity l , then $p_i(l)$ will increase. Otherwise, $p_i(l)$ will decrease.

The initial probabilities $p_i^0(l)$ are based on the sum of the squares of the differences between a small window from image 1 centered on (x_i, y_i) and a window from image 2 centered on $(x_i + l_x, y_i + l_y)$. (The window sizes in all examples presented here are 5×5 .) Let this sum be $s_i(l)$ for $l \neq l^*$. When $s_i(l)$ is small $p_i^0(l)$ should be large, and vice versa.

Let

$$w_i(l) = \frac{1}{1 + c * s_i(l)}, \quad l \neq l^* \quad (1)$$

be the "weight" associated with the label l of node a_i for some positive constant c . ($c = 10$ for all examples presented in this paper.) A disparity label which associates highly similar pairs of regions will have a large weight value. Note that $w_i(l)$ is in the interval $[0, 1]$ and is inversely related to $s_i(l)$, but in general the sum of the weights is not 1 and $w_i(l^*)$ is undefined, so we cannot use these weights as our probability estimates. Nevertheless, $w_i(l)$ has the proper qualitative behavior for probability estimates.

We first estimate the initial probability of the undefined disparity, $p_i^0(l^*)$, by observing that in many cases the label of maximum weight is the correct one. If no label has a high weight then there is probably no valid match. It may often be the case that the correct label is not the one of maximum weight, but if there is a significant correlation between maximum weight and ground truth the relationship will prove an adequate initial estimate of the probability that each node is matchable.

Using this relationship, let

$$p_i^0(l^*) = 1 - \max_{l \neq l^*} (w_i(l)) \quad (2)$$

be the initial estimate of the probability that the point (x_i, y_i) in image 1 corresponds to no point in image 2.

We can apply Bayes' rule to obtain an initial estimate of the probability that a_i should be label l for labels other than l^* . Let

$$p_i^0(l) = p_i(l|i) * (1 - p_i^0(l^*)), \quad l \neq l^* \quad (3)$$

where $p_i(l|i)$ is the conditional probability that a_i has label l given that a_i is matchable, and $(1 - p_i^0(l^*))$ is the probability that a_i is matchable. We can estimate $p_i(l|i)$ with

$$p_i(l|i) = \frac{w_i(l)}{\sum_{l' \neq l^*} w_i(l')}. \quad (4)$$

Equations (2), (3), and (4) can be used to calculate the initial probabilities for every label of every node.

The initial probabilities, which depend only on the similarity of neighborhoods of candidate matching points, can be improved by using the consistency property. We want a rule for updating $p_i^k(l)$ which has the following property. The new probability $p_i^{k+1}(l)$ should tend to increase when nodes with highly probable labels consistent with l are found near node a_i . Labels are considered consistent if they represent nearly the same disparities,

$$\|l - l'\| \leq \Theta$$

for an appropriate threshold. For the examples used in this paper the condition for consistency is

$$\|l - l'\| = \max(|l_x - l'_x|, |l_y - l'_y|) \leq 1.$$

A node a_j may be considered near a_i if

$$\max(|x_i - x_j|, |y_i - y_j|) \leq R.$$

That is, the points corresponding to a_i and a_j in image 1 are no more than R rows or columns apart. ($R = 15$ in all examples presented here.)

The degree to which the label l' of a_j reinforces $p_i(l)$ should be related to the estimated likelihood that l' is correct. To compute the new probabilities $p_i^{k+1}(l)$ for all l in L_i we examine each node in an area surrounding a_i , but not including a_i . Let

$$q_i^k(l) = \sum_{\substack{j \in a_i \\ \text{near } a_i \\ j \neq i}} \left[\sum_{\substack{l' \in L_j \\ \|l - l'\| < \Theta}} p_j^k(l') \right], \quad l \neq l^*. \quad (5)$$

In all cases $q_i^k(l) \geq 0$. This quantity is zero if and only if no nodes surrounding a_i have possible matches with disparity labels similar to l . It will be large when there are several nodes with highly probable matches surrounding a_i which have disparity labels similar to l . In general, $q_i(l)$ varies according to the consistency of l with the current estimate of the disparities in the local neighborhood of a_i . All nodes surrounding a_i are treated uniformly.

We can use the following rule to update the label probabilities of node a_i using q_i . Let

$$\hat{p}_i^{k+1}(l) = p_i^k(l) * (A + B * q_i^k(l)), \quad l \neq l^* \quad (6)$$

and

$$\hat{p}_i^{k+1}(l^*) = p_i^k(l^*). \quad (7)$$

We must normalize the \hat{p} 's to obtain the new probabilities,

$$p_i^{k+1}(l) = \frac{\hat{p}_i^{k+1}(l)}{\sum_{l' \in L_i} \hat{p}_i^{k+1}(l')}. \quad (8)$$

Parameters A and B are positive constants which influence the convergence characteristics of the model ($A = 0.3$ and $B = 3$ in all examples presented here). The role of A is to delay the total suppression of unlikely labels. Even if $q_i(l)$ is zero, the positive A ensures that the new probability does not become zero. This is desirable because information may propagate to a_i from other nodes which will eventually cause

$p_i(l)$ to increase. The role of B is to determine the rate of convergence. The larger B is relative to A the faster will be the convergence of the disparity assignments. A and B may be interpreted as damping and gain parameters. There is an effective constant correlation for label pairs which are similar, and a negative correlation for all others. (For a more general discussion of updating rules in relaxation labeling algorithms see [18].)

The probability of the label l^* (undefined disparity) is affected only by the normalization step (8). If the net contribution to $\hat{p}_i(l)$, $l \neq l^*$, is such that

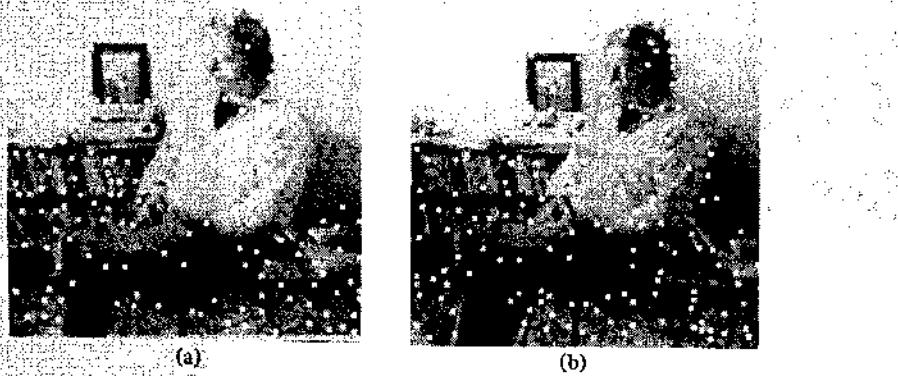
$$\sum_{l \neq l^*} \hat{p}_i^{k+1}(l) < \sum_{l \neq l^*} p_i^k(l), \quad (9)$$

then the probability that a_i has disparity l^* increases, but if this is not the case the probability decreases or perhaps remains the same.

The complete procedure to estimate the most likely disparities for each potentially matchable point in image 1 can be summarized as follows. First, a network of nodes corresponding to possible matches is constructed. Each possible match is assigned an initial likelihood using (1), (2), (3), and (4). These likelihood estimates are iteratively refined using (5), (6), (7), and (8). We have observed that after a few iterations most of the possible matches have very low probability. To increase the efficiency of the algorithm it is effective to purge these from the set of possible matches after each iteration. (In the examples presented here we purge a tentative match when its probability falls below 0.01.) If a node a_i loses all its possible matches, then (x_i, y_i) is classified as "unmatchable." That is, $p_i(l^*) = 1$. This procedure may be repeated until the network reaches a steady state, but in practice we arbitrarily stop it at ten iterations. Those nodes having a disparity with an estimated likelihood of 0.7 or greater are considered to be matched. Some nodes may remain ambiguous, with several potential matches retaining nonzero probabilities.

IV. RESULTS

An interesting and important property of this matching algorithm is that it works for any mode of disparity and does not require precise information about camera orientation, position, and distortion. Disparity in multiple images of the same scene may be due to translation or rotation of the sensor or due to motion of objects in the scene. At least three specific cases are commonly found in real world situations. The disparity is strictly temporal when a single, stationary camera records a sequence of images. Temporal disparity specifies the motion of objects in the image plane. The disparity is stereoptic when two horizontally displaced cameras simultaneously record the same scene. Often the focal axes are also rotated about a fixation point which has an arbitrary disparity of $(0, 0)$. (If the focal axes are parallel the fixation point is at infinity.) Another salient mode is forward sensor motion. While the motion is commonly along the focal axis, camera rotations and/or off-axis motion may also occur. The computational model described in the previous section was tested on examples of each of these cases. A precise photogrammetric model could translate these results into quantitative



(a)

(b)

Fig. 1. Stereogram.

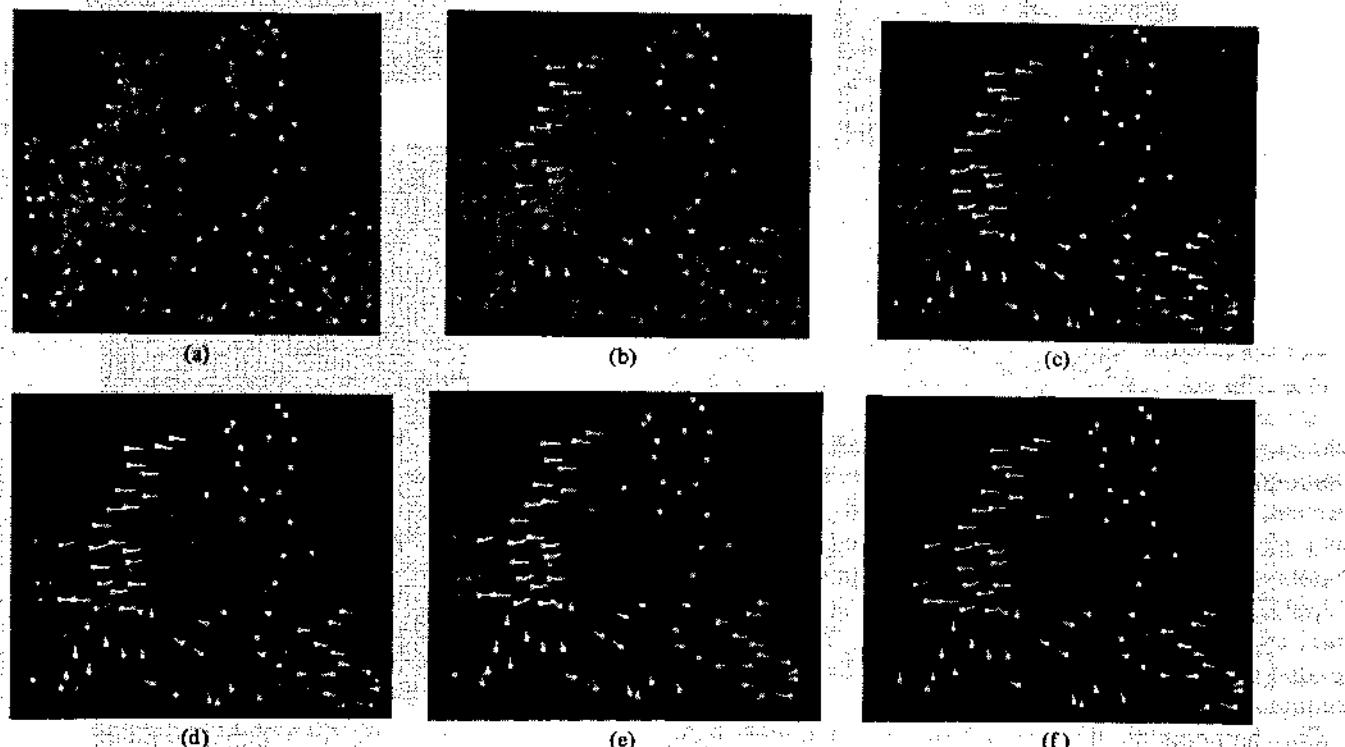


Fig. 2. (a) Initial probability assignments for Fig. 1. (b)-(f) Iterations 2, 4, 6, 8, and 10.

measurements of velocity and depth, but we shall only discuss their qualitative significance.

The first case is a stereogram (Fig. 1). The candidate matching points selected by the interest operator are superimposed on the images. Fig. 2 shows the performance of the model over ten iterations, displaying every second iteration in addition to the initial state. Each point represents a node and each line a possible match. Each line is in the direction of the corresponding disparity vector, and the length is proportional to the length of the vector. A match of small disparity appears only as a point because the line is too small to be visible. The brightness of each point is proportional to the probability that the point is matchable, $(1 - p_i^k(l^*))$, and the brightness of each line is proportional to the probability of the disparity assignment, $p_i^k(l)$. Initially, there are many possible matches with relatively low probability, but by iteration 4 [Fig. 2(c)] almost

all of them have been discarded. On iteration 10 [Fig. 2(f)] only those classifications with probability greater than 0.7 are shown. The cameras are fixated on the person's head, which has a disparity of $(0, 0)$. More distant points have disparities with positive x components and nearer points have disparities with negative x components. Observe that from the person's head to his left knee and then to his left foot disparity varies smoothly, following the smooth transition from far to near and back to far [Fig. 2(f)]. Between the person and the background, however, there is a step change in depth at the occluding edge, and here the assigned disparities exhibit an abrupt transition from near to far. The nonzero y components of some of the closer points is due to motion of the subject between exposures (only one camera was used to record this stereogram).

The next example illustrates temporal disparity. Fig. 3



(a)

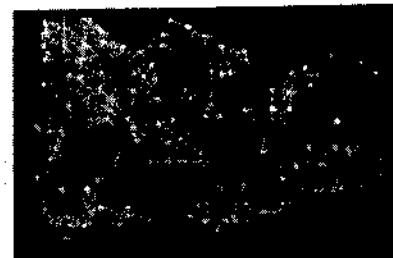


(b)

Fig. 3. Object motion.

shows a sequence in which the truck moves approximately seven pixels to the right while the camera, and hence the background, remains stationary. The truck is easily distinguishable as a cluster of points with disparity of about $\langle 7, 0 \rangle$ and the background as a cluster of points with disparity of about $\langle 0, 0 \rangle$ [Fig. 4(f)]. An error occurs at a point slightly above the roof of the truck. The point is actually part of the background but its estimated disparity is characteristic of the truck surface.

The third example simulates a view from an airplane flying over a city (Fig. 5). Actually, the subject was a scale model of downtown Minneapolis which was photographed with an ordinary camera and tripod. As the camera "flies" over the "city" it rotates downward in the frontal plane to fixate on a point near the center of the image (Fig. 6). The final disparity vectors diverge from the fixation point which has disparity $\langle 0, 0 \rangle$ [Fig. 7(f)]. Two distinctive features in Fig. 7(f) are the cluster of points on the large building in the near foreground, which has a large disparity because it is very close, and the cluster of points in the far background (upper left corner), which has mainly vertical disparity because of the rotation of the camera. The algorithm is susceptible to an aliasing effect when it encounters a dense cluster of similar points, such as would occur in a high frequency periodic subimage. An example of this may be seen at the middle of the right border of Fig. 7(f), where a number of points have been misclassified. One way to avoid this problem would be to enlarge the area in which interest values must be locally maximal to produce feature points, thereby constraining the density of feature points to a relatively small value.



(a)



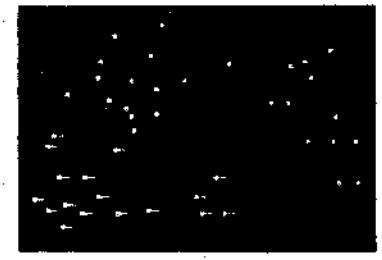
(b)



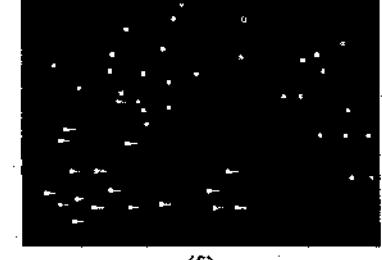
(c)



(d)



(e)



(f)

Fig. 4. (a) Initial probability assignments for Fig. 3. (b)-(f) Iterations 2, 4, 6, 8, and 10.

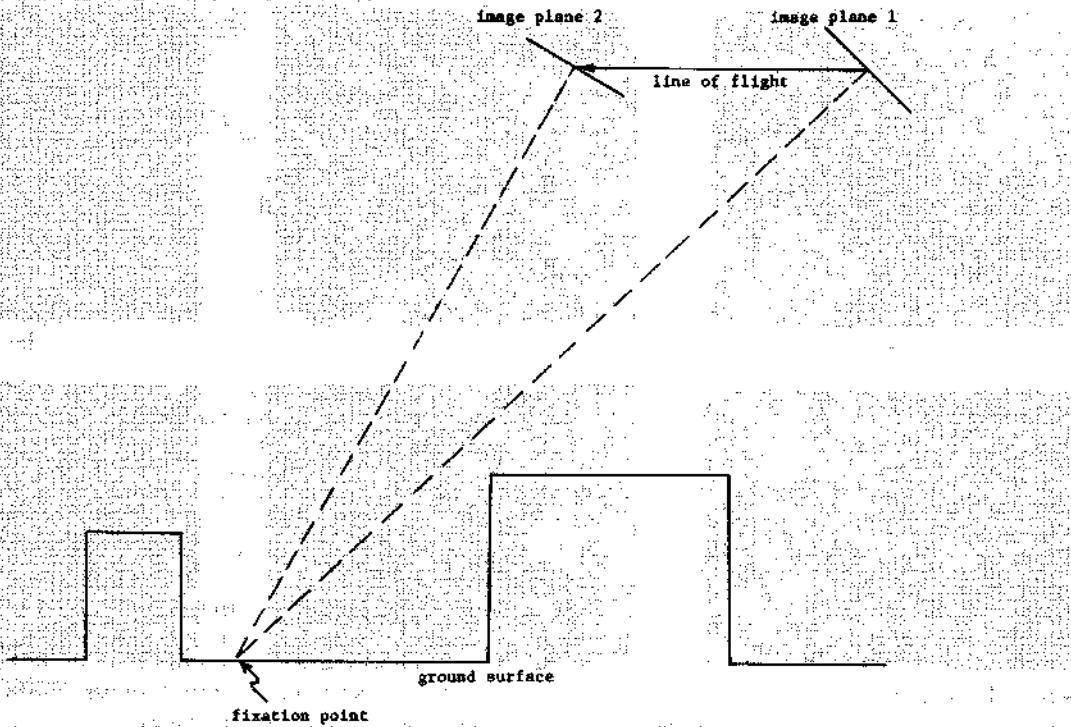


Fig. 5. Schematic of third example.

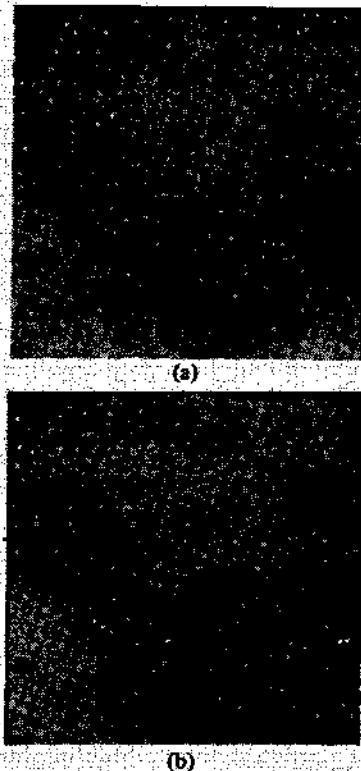


Fig. 6. Motion parallax.

V. SUMMARY AND CONCLUSIONS

Disparity relationships between images convey important information for spatial perception. The conventional cross correlation approach to matching for disparity analysis is prone to errors caused by distortion in the imaging process and the unavoidable occurrence of ambiguous matches. Increasing the sophistication of similarity detection can reduce

the problem of ambiguity, but the necessary adaptive windowing or extensive preprocessing can also reduce the efficiency and reliability of the matching process. Geometric distortion may prevent accurate similarity measurement by cross correlation. Constraining the search with a precisely known camera geometry restricts the process to special cases.

Another way to attack the ambiguity problem is to use information other than similarity to resolve ambiguous matches. The continuity of real world surfaces, a very general property which is independent of camera geometry, constrains disparity to values which are consistent with the local matching context. The consistency property can be used to iteratively refine a probabilistic estimate of disparity classification which allows ambiguous matches. The initial estimate may be obtained from a simple measure of similarity. Expensive searching is minimized by applying a simple interest operator to both images to find two sets of discrete feature points, resulting in a greatly reduced search space.

In practice the procedure is effective in a wide variety of cases, converging quickly to an unambiguous disparity classification which accurately reflects the large-scale spatial characteristics of real world scenes. It is robust in that it is not very sensitive to noise and distortion and does not require careful registration of images. It does not require complicated, carefully tuned updating functions, nor very precise initial probability estimates. The success of the method comes in large part from its ability to integrate effectively different sources of information into a simple procedure.

ACKNOWLEDGMENT

We would like to thank R. Hummel, C. Lemche, H. Nagel, S. Sahni, and A. Yonas for useful suggestions and criticism. We would also like to thank W. Franta and the staff of the

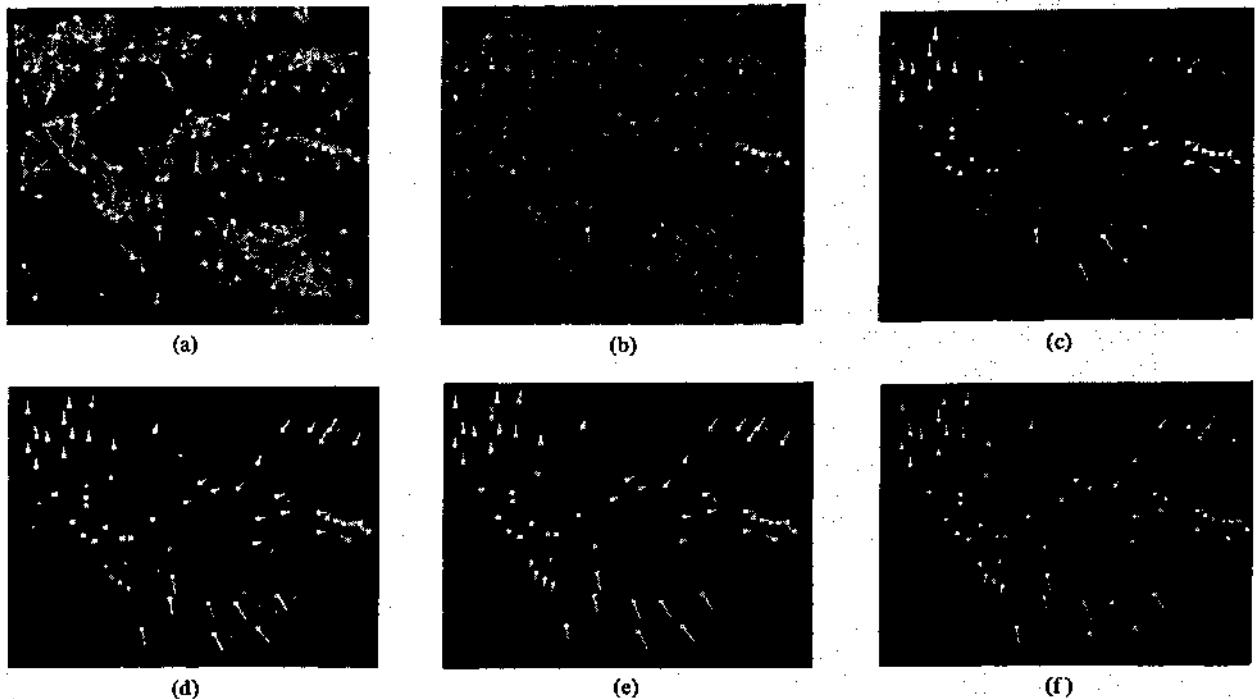


Fig. 7. (a) Initial probability assignments for Fig. 6. (b)-(f) Iterations 2, 4, 6, 8, and 10.

Special Interactive Computing Laboratory, University of Minnesota, for their helpful assistance and for the use of their excellent facilities.

REFERENCES

- [1] R. O. Duda and P. E. Hart, *Pattern Recognition and Scene Analysis*. New York: Wiley, 1973.
- [2] S. Ullman, "The interpretation of visual motion," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, MA, May 1977.
- [3] J. J. Gibson, *The Perception of the Visual World*. Cambridge, MA: Riverside, 1950.
- [4] B. Julesz, *Foundations of Cyclopean Perception*. Chicago, IL: Univ. Chicago Press, 1971.
- [5] M. J. Hannah, "Computer matching of areas in stereo images," Stanford A.I. Memo. 239, July 1974.
- [6] L. H. Quam, "Computer comparison of pictures," Stanford A.I. Memo, AIM-144, May 1971.
- [7] M. D. Levine, D. A. O'Handley, and G. M. Yagi, "Computer determination of depth maps," *Comput. Graphics Image Processing*, vol. 2, pp. 131-150, 1973.
- [8] R. Nevatia, "Depth measurement by motion stereo," *Comput. Graphics Image Processing*, vol. 5, pp. 203-214, 1976.
- [9] D. J. Kahl, A. Rosenfeld, and A. Danker, "Some experiments in point pattern matching," Univ. Maryland, Tech. Rep. TR-690, Sept. 1978.
- [10] K. Price and D. R. Reddy, "Change detection and analysis in multispectral images," in *Proc. Int. Joint Conf. Artificial Intell.*, pp. 619-625, 1977.
- [11] H. Nagel, "Formation of an object concept by analysis of systematic time variations in the optically perceptible environment," *Comput. Graphics Image Processing*, vol. 7, pp. 149-194, 1978.
- [12] K. Mori, M. Kidode, and H. Asada, "An iterative prediction and correction method for automatic stereocomparison," *Comput. Graphics Image Processing*, vol. 2, pp. 393-401, 1973.
- [13] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1978.
- [14] D. B. Gennery, "A stereo vision system for an autonomous vehicle," in *Proc. 5th Int. Joint Conf. Artificial Intell.*, Cambridge, MA, Aug. 1977, pp. 576-582.
- [15] S. L. Tanimoto, "A comparison of some image searching methods," in *Proc. IEEE Comput. Soc Conf. Pattern Recognition Image Processing*, Chicago, IL, June 1978, pp. 280-286.
- [16] D. I. Barnea and H. F. Silverman, "A class of algorithms for fast

image registration," *IEEE Trans. Comput.*, vol. C-21, pp. 179-186, Feb. 1972.

- [17] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 194, pp. 283-287, Oct. 15, 1976.
- [18] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, June 1976.
- [19] H. P. Moravec, "Towards automatic visual obstacle avoidance," in *Proc. 5th Int. Joint Conf. Artificial Intell.*, Cambridge, MA, Aug. 1977, p. 584.



Stephen T. Barnard (S'78-M'79) received the B.S. degree in mechanical engineering from Case Western Reserve University, Cleveland, OH, and the M.S. and Ph.D. degrees in computer science from the University of Minnesota, Minneapolis.

He has worked for Sperry Univac, the 3M Corporation, and Honeywell, Inc. He is currently with the Artificial Intelligence Center, SRI International, Menlo Park, CA, where he is working on image understanding and automated visual inspection.



William B. Thompson (S'72-M'75) was born in Santa Monica, CA, in August 1948. He received the Sc.B. degree in physics from Brown University, Providence, RI in 1970, and the M.S. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, in 1972 and 1975, respectively.

From 1972 to 1975 he was a member of the Image Processing Institute at the University of Southern California. Since September 1975, he has been an Assistant Professor in the Department of Computer Science, University of Minnesota, Minneapolis. His interests are in artificial intelligence and scene analysis.

Dr. Thompson is a member of Sigma Xi, Eta Kappa Nu, and the Association for Computing Machinery.

Finding Trajectories of Feature Points in a Monocular Image Sequence

ISHWAR K. SETHI, MEMBER, IEEE, AND RAMESH JAIN, SENIOR MEMBER, IEEE

Abstract—Identifying the same physical point in more than one image, the correspondence problem, is vital in motion analysis. Most research for establishing correspondence uses only two frames of a sequence to solve this problem. By using a sequence of frames, it is possible to exploit the fact that due to inertia the motion of an object cannot change instantaneously. By using *smoothness of motion*, it is possible to solve the correspondence problem for arbitrary motion of several nonrigid objects in a scene. We formulate the correspondence problem as an optimization problem and propose an iterative algorithm to find trajectories of points in a monocular image sequence. A modified form of this algorithm is useful in case of occlusion also. We demonstrate the efficacy of this approach considering synthetic, laboratory, and real scenes.

Index Terms—Correspondence, motion object tracking, path coherence, smoothness of motion, structure from motion.

I. INTRODUCTION

THE last few years have seen increasing interest in dynamic scene analysis. The input to a dynamic scene analysis system is a sequence of images. As is well known, an image represents a 2-D projection of a 3-D scene at a time instant. A major problem in a computer vision system is to recover the information about objects in a scene from images. This problem cannot be solved without some assumptions about the world. A sequence of frames allows one additional dimension to recover the information about the 3-D world that is lost in the projection process. Multiple views of a moving object acquired using a stationary camera may allow recovery of the structure of the object [4], [36], [32]–[34], [31], [24]. A mobile camera may be used to acquire information about the structure of the stationary objects in a scene using optical flow [6], [22], axial motion stereo [20], and other methods [16], [14].

Many researchers in psychology of vision support the recovery of information from image sequences, rather than an image, representing a scene [19], [6], [16]. Gibson [16] argued in support of active information pickup by the observer in an environment. Johansson [16] demonstrated the efficacy of only motion information in recognition of

objects using moving light displays. Neisser [19] proposed a model according to which the perceptual processes continually interact with the incoming information to verify anticipations formed on the basis of available information until a given time instant. In computer vision systems, the efficacy of even noise-sensitive approaches, such as difference and accumulative difference pictures, was demonstrated by using hypothesize-and-test mechanisms to analyze complex real-world scenes [13]. Although many researchers are addressing the problem of recovering information in dynamic scenes, it appears that due to the legacy of static scenes most researchers are approaching the recovery problem using just two or three frames of a sequence. This self-imposed restriction results in approaches suitable for *quasi-dynamic* scene analysis, rather than dynamic scene analysis. Since the information recovery process requires constraints about the scene, the analysis based on a minimal number of frames rests on assumptions that ignore the most important information in dynamic scenes—the motion of objects.

Structure from motion has attracted significant research efforts recently from researchers working in the field of dynamic scene analysis [27], [31], [32], [34], [36], [38]. Ullman popularized the *rigidity assumption* in computer vision. This assumption states that any set of elements undergoing a 2-D transformation which has a unique interpretation as a rigid body moving in space should be so interpreted. The rigidity assumption allows recovery of the structure of objects, under certain conditions, in three frames.

Another popular approach for the recovery of the structure and motion is to use optical flow fields [16], [22], [14], [18], [26], while others try to recover the same information using points in frames. The optical flow is the field of retinal velocities. In computer vision, it is considered the velocity field for all image points. It has been shown that the optical flow contains information about the motion of the observer and the environment. Approaches for the computation [10] and for the recovery of structure [37] have been proposed. Considering the difficulties in computing optical flow of acceptable quality, some efforts are being made to recover the structure using the characteristics of optical flow, but without computing it [11], [20].

Recently, the trajectory-based recovery has attracted some attention [31], [28], [38], [24], [15]. It has been

Manuscript received April 18, 1985; revised February 6, 1986. Recommended for acceptance by W. B. Thompson. This work was supported in part by the NSF under Grant DCR-8500717.

I. K. Sethi is with the Department of Computer Science, Wayne State University, Detroit, MI 48202.

R. Jain is with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 8609395.

shown that the human visual system requires an extended frame sequence to recover the structure of moving patterns [31], [24] and that the noise sensitivity of the system improves with an increase in the number of frames [3]. We believe that the trajectory-based approach to the recovery of structure is more suitable for complex scenes. This approach will allow successive refinement of the structure of objects as more frames are observed. Using the first few frames, an initial, although tentative, structure of an object might be hypothesized, and then this structure could be refined based on later observations. A major advantage of such an approach would be to free ourselves from assumptions of rigidity and rely more on natural assumptions of motion characteristics.

A major step in the recovery using a token-based approach, both in quasi-dynamic and dynamic approaches, is identifying images of a physical point in several frames, usually called the correspondence problem. This paper addresses the problem of establishing correspondence for tokens in a sequence of images. Contrary to most other approaches, we do not try to solve the correspondence problem using just two frames. In two frames, one can use only the location of the tokens and has to make assumptions about the nature of the objects or about the maximum velocity of objects. A longer sequence of frames allows the use of velocity information in solving the problem [35]. Jenkin [15] proposed an approach for establishing correspondence in a binocular image sequence. He suggested the use of the smoothness of velocity, and his approach is influenced by the Gestalt rules. We also believe that the smoothness of velocity is more general and more powerful compared to the rigidity assumption. As shown in a latter section, our approach is similar to Jenkin's in using smoothness, but is very different from his approach in many other respects.

The smoothness constraint has been used by some other researchers, also in dynamic scenes [10], [8]. The smoothness used by them, however, is different from the smoothness we are using. Horn and Schunck use the smoothness of velocity of neighboring points on a surface to propagate velocity vectors computed at some points of the surface to the other points of the surface in the same frame. Hildreth also uses the smoothness of the velocity field to compute the optical flow for rigid objects. We use the smoothness of motion of a point over several frames.

We propose an optimization approach to the correspondence problem and propose an iterative optimization algorithm to find optimal trajectories. This computational approach allows solution of the correspondence problem incrementally. As new frames are acquired, our iterative optimization approach establishes the correspondence. Thus, solution of the correspondence problem also gives us trajectories of points, which, in turn, may be used to obtain structure. Our approach is able to handle limited occlusion and disocclusion also. A limitation of the proposed approach, similar to Jenkin's, is that it assumes uniform motion. If for some reason there is a sudden change in motion, this approach will not be able to detect it and may result in wrong correspondence.

Section II discusses approaches used for correspondence, with emphasis on Jenkin's approach because of its similarity to our approach. Section III gives possible applications of the proposed approach. Path coherence and smoothness of motion, as used in our approach, are discussed in Section IV, and the optimization approach to the correspondence is presented in Section V. Section VI gives the greedy exchange algorithm for establishing correspondence. The issues related to the formulation of the path coherence function are discussed in Section VII, and results of the application of the algorithm are given in Section VIII. The basic algorithm has some problems. These problems and modifications to handle occlusion are discussed in Section IX. Results for laboratory sequences and a real movie (*Superman*) sequence are presented to show the efficacy of the algorithm.

II. CORRESPONDENCE PROBLEM

In the recovery of structure from motion, the establishment of the token correspondence has been a major hurdle. Ullman [36] proposed a minimal mapping approach to this problem. His approach was based on the 2-D features of the tokens and the 2-D distance between tokens. His elegant approach is computationally very attractive because of the possibility of implementation using a network of processors. The major problem with the approach is, however, that it fails to consider the fact that most realistic motion analysis must be performed by considering an extended region of space-time. Correspondence based on 2-D distances alone may lead to erroneous results in scenes containing several objects moving in assorted directions and in case of nonrigid motion. In fact, psychological experiments conducted to study the feasibility of this approach consider only a few points in artificial motion situations; Todd [31] shows that most human observers require several frames to infer the structure of moving objects, even in simple experimental situations.

Barnard and Thompson [1] and Prager and Arbib [23] proposed relaxation-based approaches to the solution of this problem. In relaxation-based methods, all tokens in frame 2 that are within a spatial neighborhood of a token in frame 1 are assigned a weight indicating the match strength between the token in frame 1 and the tokens in frame 2. It is assumed that only one of these matches is correct. The correct match is obtained by using relaxation. The relaxation process tries to find the most consistent matches based on the disparities. The most consistent match implies uniform motion and hence *rigid* objects. These approaches also use 2-D displacements in two frames as a major factor in establishing the correspondence.

These approaches to the solution of the correspondence problem try to solve the problem using tokens in two frames only. In fact, most research in dynamic scene analysis, or time-varying image analysis, has been concerned with only two or three frames, *not with dynamic scenes*. It is not clear why most researchers in this field have been concerned with the recovery of information

considering a minimal number of points in a minimal number of frames, rather than with the recovery of reliable information (see Brady's comment on Todd's paper in [2]). It appears that for real-world problems, like many other problems in dynamic scene analysis, the correspondence problem may be solved more reliably using more than two frames. Due to the self-imposed limitation of working with only two frames, some form of the rigidity assumption is essential for solving the problem. In two frames, the most powerful constraint is the spatial uniformity of displacements. The rigidity assumption, with orthographic projections, is very appealing because it guarantees the uniformity of displacements and is domain independent. If we consider more than two frames in establishing correspondence, however, we may admit nonrigid objects in our analysis. Thus, assumptions about the nature of motion can be used to relax those about the nature of objects. In its simplest form, the smoothness of motion is used to relax the rigidity of objects.

This can be done using *path coherence* and assuming *smoothness of motion*. In this paper, we introduce the notion of *path coherence* and use it for solving the correspondence problem as an optimization problem.

Jenkin [15] proposed a novel approach for combining the solution of correspondence problems for stereo and motion. He argues in support of the smoothness assumption. He is interested, like us, in an approach that is based not on assumptions like rigidity [36], planarity [9], and rotation and translation [38], but on the smoothness of motion. His notion of smoothness is related to 3-D motion. It is based on the following.

- 1) The location of a given point would be relatively unchanged from one frame to the next.
- 2) The scalar velocity, or speed of a given point, would be relatively unchanged from one frame to the next.
- 3) The direction of motion of a given point would be relatively unchanged from one frame to the next.

To solve the correspondence problem, he assumed that at all instants all points were visible in both images of the binocular frame sequence and that the 3-D position and velocity of the points were known at some initial time. As will be shown in the next section, our approach uses smoothness of image motion and uses a monocular sequence. Moreover, we have extended the algorithm to cope with limited occlusion; severe occlusion requires characteristics of objects. Since we are concerned only with points, our domain in this paper may be considered to be similar to that of Rashid [25], O'Rourke and Badler [21], and Jenkin [15]. In this paper, we do not address the problem of occlusion analysis in general cases using other than simple motion characteristics.

III. APPLICATIONS OF THE TRACKING

The tracking approach proposed here will play an important role in many different applications. A complete object may be assumed to be a point, and then the proposed algorithms may be applied to those points. This approach will be useful in the analysis of scenes containing

several moving objects. Some possible applications are traffic analysis, cell motion analysis, and others. A more important application of the proposed algorithm may be in the segmentation of dynamic scenes by grouping points with similar motion in one object [5]. A nonrigid or jointed object can be segmented by tracking individual points and then grouping these points based on common motion characteristics, as is done by humans as demonstrated by Johansson [16]. This segmentation will give an idea of the general structure of an object. For rigid objects, by tracking several points on the object, one may use the structure from motion approaches using trajectories [28], [31].

IV. PATH COHERENCE AND SMOOTHNESS OF MOTION

We exploit the fact that, due to inertia, the motion of a physical entity cannot change instantaneously. If a frame sequence is acquired at a rate such that no dramatic changes take place between two consecutive frames, then for most physical objects no abrupt change in the motion can be observed. Thus, a very reasonable assumption for the analysis of real-world dynamic scenes is: *the motion of an object at any time instant cannot change abruptly*. This is *path coherence*.

Note that this assumption will be valid for all moving objects, rigid and nonrigid. If the objects collide, then some high-level process may be required to analyze the motion after the collision. In most other cases, if the sampling rate is high enough, then the changes in the motion will be gradual.

The projection of a smooth 3-D trajectory will be a smooth 2-D trajectory in both orthographic and perspective projections. This can be easily verified by considering the projection equations and analyzing first and second derivatives of the projected points. It can be shown that for any continuous variations in the velocity of a point in space, the velocity of the projected point will also be continuous. This fact allows us to study the correspondence problem using path coherence in images, rather than in 3-D space as in [15].

Let us consider a trajectory T_i for a point P_i in an image sequence. The coordinates of the point are given by a vector X_i in a frame. The coordinates in the k th frame for this point are denoted by X_{ik} . Let us represent a trajectory T_i as

$$T_i = \langle X_{i1}, X_{i2}, \dots, X_{in} \rangle \quad (1)$$

where X_{ik} is the point in the k th frame participating in the i th trajectory. A set of points in the k th frame will be denoted by X^k .

Now let us consider the deviation d_i^k in the path of the point in the k th frame. We denote a measure of deviation in the path, and hence a measure of path coherence, as

$$d_i^k = \Psi(X_{ik-1}X_{ik}, X_{ik}X_{ik+1}) \quad (2)$$

where Ψ is a path coherence function. This function may consider the direction or magnitude, or both, of the displacements of the point in an image sequence.

We may define the deviation for the trajectory as

$$D_i = \sum_{k=2}^{n-1} d_i^k. \quad (3)$$

Note that for translational motion, the difference in the direction will be a suitable measure of deviation; in more complex cases, one may need more rigorous measures. For uniform translational motion in an image plane, D_i will be zero for a correct trajectory. For an arbitrary trajectory, if the motion is smooth, the value of D_i will be very small.

Let us consider the projection of two points in frames of a sequence, as shown in Fig. 1. The points are moving in different directions. The three frames under consideration are at the crossover of the trajectories of the points. The rigidity assumption cannot be used in this situation. Moreover, if distances between points in two frames are the basis for finding the trajectory of the points, then one will obtain an abrupt change in the direction of motion from the second frame to the third frame. The path coherence will allow determination of the correct correspondences over three frames and will result in the correct trajectories, as shown in Fig. 2. If we are considering a trajectory, then the instantaneous change in the motion of a physical entity may be obtained by comparing the motion vectors from the previous frame and the current frame. Path coherence requires that there not be abrupt changes in the motion vectors obtained from consecutive frames.

In a scene, several objects may be undergoing random motions. The objects may be rigid, jointed, or arbitrary; the motion of each object, however, will be smooth. Thus, if a dynamic scene is sampled at a rate fast enough to capture all significant events in the scene, then the observed motion of all objects will be smooth trajectories. As suggested by Todd [31], more reliable motion perception may be achieved by considering the primitive units of motion perception to be the global trajectories in an optic array defined over an extended region of space-time.

Now, if there are m points in a sequence of n frames, resulting in m trajectories, then the deviation for all the trajectories should be considered. This deviation is given by

$$D(T_1, T_2, \dots, T_m) = \sum_{i=1}^m \sum_{k=2}^{n-1} d_i^k. \quad (4)$$

The problem we are facing is to determine these trajectories. Thus, our problem, in its simplest form, may be stated as: *given m points in each of the n frames of a sequence, determine m trajectories*. Since there are m^n total possible trajectories, only m of which are to be selected, we need a method to judiciously select these m trajectories. We know that each trajectory is smooth, and hence a reasonable approach is to try to maximize the smoothness of the set of trajectories that are selected as the correct set. This approach is consistent with those pro-



Fig. 1. In this figure, we show the trajectories of two points. We are concerned with the three points on the crossover of the trajectories. The points in the first, second, and third frames are, \square , x , and Δ , respectively.

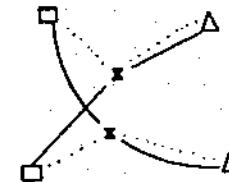


Fig. 2. The correspondence established using 2-D distance and the smoothness of motion are shown using broken and full lines, respectively.

posed by [15], [24], [31]. This approach of maximizing smoothness of motion may be stated as: *if a set of points undergoing a 2-D transformation has a unique interpretation as a set of points following smooth 3-D trajectories, then it should be so interpreted*.

Note that the above smoothness of motion conjecture is an extension of the path coherence assumption. The path coherence assumption for a point in a frame considers its motion from the previous frame to the current frame and from the current frame to the next frame; the smoothness of motion hypothesis extends this notion of *smooth velocity changes* to a longer sequence. It should be emphasized here that this hypothesis gives more importance to the continuity of motion, rather than displacements in just two frames, and considers individual point characteristics for establishing correspondence in a dynamic scene. Clearly, two frames are not enough to apply smoothness of motion, not even to use path coherence. We require at least three frames to compute the change in the motion. For verifying smoothness of motion, more frames will be required.

V. FINDING TRAJECTORIES

In a general but noise-free case, elements may be moving in assorted directions. Suppose that a feature detector, such as a moving corner detector [29], gives points in each frame. In the problem formulation, let us assume that the number of moving points in each frame is m . The problem now is to find trajectories of m points in n frames. We can make general assumptions based only on motion characteristics, not on the nature of objects. These assumptions are the following.

- 1) An element in a frame can only belong to one trajectory.
- 2) There should be m trajectories, each containing n points.

3) For each trajectory, the deviation should be minimal.

4) The sum of the deviations for trajectories should be minimal.

The first two assumptions above are to assign each point to only one unique trajectory. The path coherence assumption requires that the deviation in the path be minimal at every time instant for a point. Path coherence alone may lead to wrong trajectories near the points of intersection of trajectories. The smoothness of motion, which is presented in the form of the fourth assumption above, comes to the rescue in such situations.

In order to represent the correct correspondence of points on a trajectory, we define an n -dimensional array C to represent m valid trajectories. This array will have

$$C(i_1, i_2, \dots, i_n) = 1 \quad (5)$$

for a valid i th trajectory. All other entries will be zero. Thus, for a set of trajectories under consideration, m out of m^n entries in the array C will be 1, while the rest will be 0. In Fig. 3 we show points on four trajectories in five frames. For the trajectories shown in the figure, only the following elements of the C array will have nonzero values:

$$C(1, 2, 3, 4, 4) = 1$$

$$C(2, 1, 1, 2, 3) = 1$$

$$C(3, 4, 4, 3, 1) = 1$$

$$C(4, 3, 2, 1, 2) = 1.$$

Now we can formulate the trajectory-finding method as an optimization problem. The total number of all possible trajectories is m^n , however, only m of those are valid. One may consider exhaustively all possible combinations of trajectories to find the correct solution. The exhaustive approach will be computationally very expensive, however. Before we consider the computational approach, let us consider the solution of the problem. According to the hypothesis of the smoothness of motion, we want to select those trajectories from the set of all possible trajectories that maximize the smoothness of the trajectories. Thus, we want to select that set which minimizes the function $D(T_1, T_2, \dots, T_m)$, given by

$$D(T_1, T_2, \dots, T_m) = \sum_{X_1 \in X^1} \dots \sum_{X_n \in X^n} \left(\sum_{k=2}^{n-1} \Psi(\overline{X_{k-1} X_k}, \overline{X_k X_{k+1}}) \right) * C(i_1, i_2, \dots, i_n), \quad (6)$$

subject to the conditions

$$\sum_{X_i \in X^i} C(i_1, i_2, \dots, i_n) = 1$$

$$\sum_{X_i \in X^i} C(i_1, i_2, \dots, i_n) = 1 \quad (7)$$

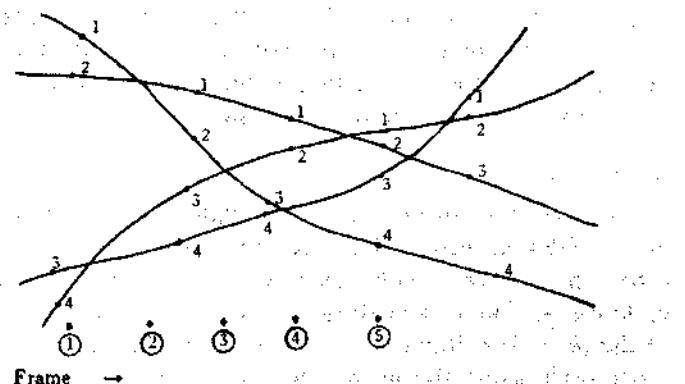


Fig. 3. The points on four trajectories are shown in five frames. This figure is a composite frame showing locations of points in different frames. The points are numbered, and the trajectories are shown by lines. The frame number is assumed increasing from left to right. The number shown next to a point indicates that in its frame it is the k th point.

and

$$C(i_1, i_2, \dots, i_n) \geq 0 \quad \text{for } X_{i_1} \in X^1, \dots, X_{i_n} \in X^n, \quad (8)$$

The set of trajectories T_1, T_2, \dots, T_m that minimizes this function should be interpreted as the correct set describing the motion of m points in n frames.

VI. OPTIMIZATION ALGORITHM

As pointed out in an earlier section, an exhaustive solution of the optimization problem to establish the correspondence is computationally a difficult problem. Although established optimization methods like branch and bound and dynamic programming can be applied to solve our problem, they are inadequate for two reasons. First, the computational requirements of these optimization methods are still quite high, and they do not exploit the redundancy of information available in dynamic scenes. Moreover, in our problem, it will be desirable to have an incremental algorithm that assigns points in the $(k + 1)$ th frame to trajectories that have already been obtained up to the k th frame.

We use an iterative optimization algorithm to assign points to proper trajectories. Suppose that initial trajectories are obtained using some criterion, such as closeness alone. Points from a new frame may be tentatively as-

signed to trajectories using a weak criterion, and then the resulting trajectories can be refined using a stepwise optimization procedure to minimize the overall deviation of the trajectories while establishing correspondence. Like hill-climbing procedures in general, stepwise optimization procedures guarantee only local optimization. Also, in these procedures, the initial correspondence plays an

important role in the final solution. The solution obtained is strongly influenced by the starting point. With these general limitations of stepwise approaches in mind, we propose an algorithm in the following.

A. Algorithm

The algorithm proposed here is called the *greedy exchange (GE) algorithm*. This algorithm will extend trajectories up to the $(k + 1)$ th frame, assuming the trajectories up to the k th frame have already been obtained. The points of the $(k + 1)$ th frame are assigned to be the established trajectories using the nearest neighbors. These tentative trajectories are then iteratively refined using the following method. The value of the criterion D for the tentative trajectories for the $(k + 1)$ th frame is

$$D = \sum_{p=1}^m D_p = \sum_{p=1}^m \sum_{q=2}^k d_p^q. \quad (9)$$

Now let us consider two trajectories, the i th and the j th, and rewrite the above equation as

$$\begin{aligned} D &= \sum_{p=1; p \neq i,j}^m D_p + \sum_{q=2}^k d_i^q + \sum_{q=2}^k d_j^q \\ &= \sum_{p=1; p \neq i,j}^m D_p + \sum_{q=2}^{k-1} d_i^q + \sum_{q=2}^{k-1} d_j^q + d_i^k + d_j^k. \end{aligned} \quad (10)$$

Suppose now we exchange the points from the $(k + 1)$ th frame on the i th and j th trajectories. Clearly, such an exchange would not affect the first three terms in the above equation. Now let d_i^k and d_j^k be the new path coherence measures for the i th and j th trajectories after the exchange. Then, for the above exchange to be profitable, we must have

$$d_i^k + d_j^k < d_i^k + d_j^k, \quad (11)$$

or

$$g_{ij}^k = d_i^k + d_j^k - (d_i^k + d_j^k) \quad (12)$$

should be positive.

1. *Initialization:* For each point in X^k , $k = 1, 2, \dots, n - 1$, determine the nearest neighbor in X^{k+1} . Using these nearest neighbors, initialize m trajectories, such that the point X_{ik+1} . In case of multiple nearest neighbors, the decisions are arbitrary, and a point in any frame is assigned to only one trajectory.

2. *Exchange Loop:* For each frame value $k = 2$ to $n - 1$:

- a. Compute the gain g_{ij}^k for $i = 1$ to $m - 1$ and $j = i + 1$ to m .
- b. Pick the $i-j$ pair providing the maximum gain.
- c. Exchange the points in the $(k + 1)$ th frame on the T_i and T_j trajectories, and set the exchange flag on.

3. *Termination:* Check the exchange flag at the end of exchange loop. If an exchange was made during a frame, go back to exchange loop; otherwise, stop.

Clearly, the complexity of the algorithm is of $O(nm^2)$. If we choose our path coherence measure such that the D cannot become negative, then the above procedure will terminate after a finite number of iterations.

VII. PATH COHERENCE FUNCTION

In selecting a function for path coherence, we used four guiding principles.

- 1) The function should not be negative.
- 2) The function should consider the amount of deviation in the direction of motion, not the sense (left or right) of the deviation. Thus, the sign of the angle of deviation should not factor in the computation of the function.
- 3) The function should respond equally to increases and decreases in speed.
- 4) If there is no change in the motion characteristics, then the function should be zero.

These four conditions help us in selecting a suitable function to represent path coherence at a point. We used the following function in our experiments:

$$\begin{aligned} \Psi(X_{ik-1}, X_{ik}, X_{ik+1}) &= w_1 \left(1 - \frac{\overline{X_{ik-1}X_{ik}} \cdot \overline{X_{ik}X_{ik+1}}}{\| \overline{X_{ik-1}X_{ik}} \| \cdot \| \overline{X_{ik}X_{ik+1}} \|} \right) \\ &\quad + w_2 \left[1 - \frac{2[\| \overline{X_{ik-1}X_{ik}} \| \cdot \| \overline{X_{ik}X_{ik+1}} \|]^{1/2}}{\| \overline{X_{ik-1}X_{ik}} \| + \| \overline{X_{ik}X_{ik+1}} \|} \right] \end{aligned} \quad (14)$$

where w_1 and w_2 are weights.

As can be easily verified, this function satisfies the above four conditions and takes into account both the direction and the magnitude of changes in the motion. In fact, the first term in the above expression can be considered *directional coherence* and the second term can be considered *speed coherence*. Note that the first term is the dot product of the displacement vectors, and the second considers the geometric and arithmetic mean of the mag-

If we are to select only one such exchange from all possible exchanges, then we should make a decision in favor of the exchange maximizing the gain, i.e., exchange i and j if

$$g_{ij}^k \geq g_{rs}^k \quad (13)$$

for all possible values of i, j, r , and s . This idea leads to the following GE algorithm.

nitude of these vectors. The weights w_1, w_2 are experimentally selected to be 0.1 and 0.9, respectively. It should be pointed out here that one may also consider acceleration in the path coherence. This will require extending the notion of the path coherence to more than three frames so that acceleration may be computed. In our current study, no effort was made to account for acceleration.

VIII. EXPERIMENTS

We studied the efficacy of the proposed approach considering several synthetic and real scenes. In the synthetic data, we considered assorted motions of points in several frames and added random noise to the position of points. In all cases, excellent results were obtained. Here we present results of two synthetic sequences to show how our algorithm works.

In the first experiment, we generated four points in space. These points were undergoing different rotational motions. The proposed algorithm tracked the images of the points and found the trajectories shown in Fig. 4.

In the next experiment with the synthetic data, we considered four points in ten frames. A composite frame showing the location of points in all frames is shown in Fig. 5. All points are moving left to right. As can be seen from the figure, the trajectories of points cross between the fourth and sixth frames. During this period, the location of points is such that a correspondence scheme based only on the location of points is likely to give wrong results. We applied our iterative algorithm to these data, using the path coherence function discussed above. The initialization phase resulted in the trajectories shown in Fig. 6(a). Note that the initialization phase considers only the location of points from frame to frame, and hence it is not surprising that the resulting trajectories are wrong. In fact, these trajectories are completely counterintuitive, when one considers the data. In the first iteration, four corrections are made. The trajectories after the first correction and after the first iteration are shown in Fig. 6(b) and (c), respectively. Note that the first correction removes the abrupt change in direction due to a wrong correspondence of the fifth frame points on the second and fourth trajectories (trajectories are numbered from top to bottom). After the first iteration, the trajectories are improved, but are far from being correct. In fact, not even one trajectory is correct. The trajectories after two, three, and four iterations are shown in Fig. 6(d)-(f), respectively. As can be seen, after each iteration the trajectories are refined. The fourth iteration results in correct trajectories, and the algorithm terminates.

We generated a real sequence in our laboratory to study the efficacy of our algorithm in real scenes. In the scene shown in Fig. 7, three blocks are moving such that their trajectories intersect. We manually obtained interesting points on these blocks. We selected several points on each block and applied our algorithm. In order to show the behavior of our algorithm, without unnecessary complications, we show results for the four points on three blocks.

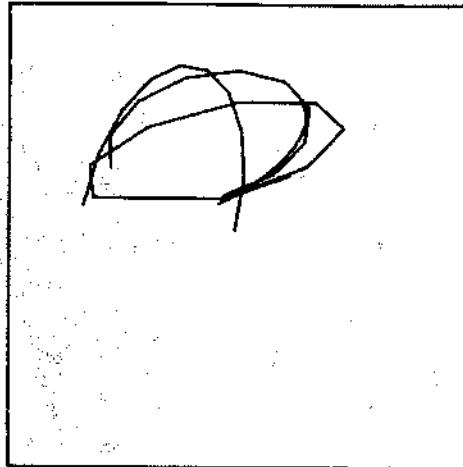


Fig. 4. The trajectories obtained by the algorithm for rotation of points in a sequence.

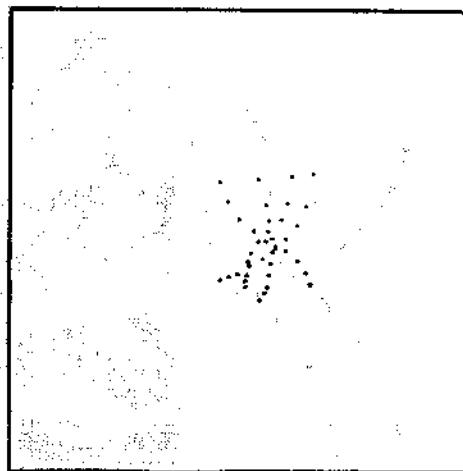


Fig. 5. A composite frame shows four points in ten frames. The points move from left to right in this figure.

The initial trajectories are shown in Fig. 8(a). The first iteration requires four corrections; the trajectories after each correction are shown in Fig. 8(b)-(e), respectively. The second iteration was enough to give the correct results, resulting in the trajectories shown in Fig. 8(f). The trajectories for all interesting points were successfully determined by our algorithm and are shown, for six frames, in Fig. 9. It required eight iterations for the convergence to final trajectories.

We applied this algorithm to a real sequence from a movie (*Superman*). The frames of the movie are shown in Fig. 10. We tracked manually selected points on the belt and head of three soldiers. These points are shown in Table I. Our algorithm converged after four iterations, giving the trajectories shown in Fig. 11. Although soldiers are moving in different directions, the motion is not really very complex, and this allowed the algorithm to converge very quickly for this case.

IX. MODIFIED GE ALGORITHM

A close look at the GE algorithm reveals that the correspondence for the first two frames is never altered as the

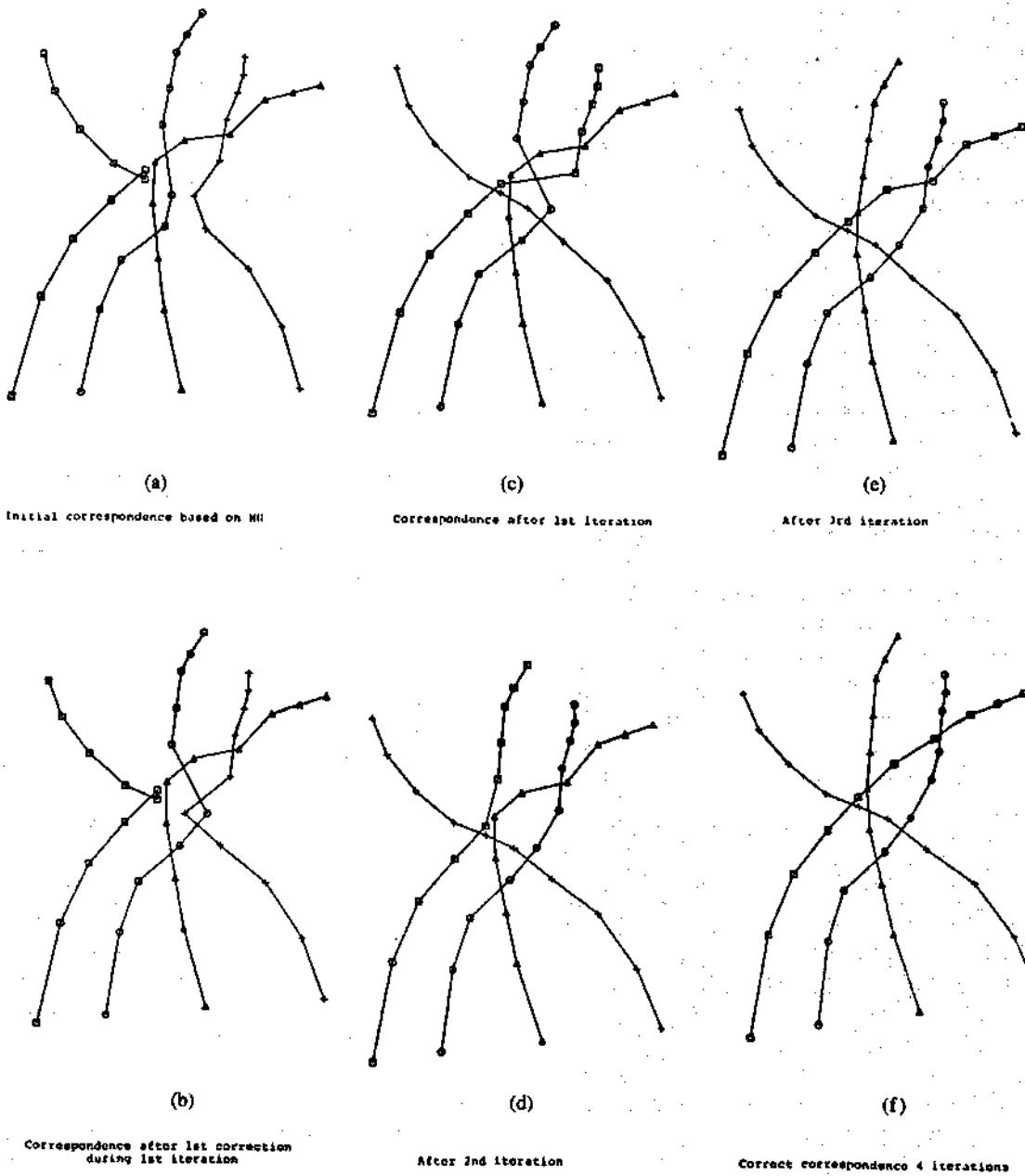


Fig. 6. The trajectories of the points after several iterations and corrections are shown in this figure.

exchange loop operates from the second frame onward. Thus, for the above algorithm to provide correct correspondence, it is essential that the correspondence for the first two frames be correct to start with. Since the initial correspondence is based purely on the closeness of feature points in successive frames, the GE algorithm was found to give poor results for the case where the displacement of the objects was comparable to the object size. Fig. 12 shows a sequence generated in our laboratory to study the efficacy of this approach. The objects and their motion were selected to confuse the algorithm. Fig. 13(a) shows the trajectories obtained by the GE algorithm for one such laboratory-generated sequence. Clearly, the source of error in this case is the initial correspondence between the first two frames, which is never altered. In order to remove this kind of error source, we modified the GE al-

gorithm. The modified greedy exchange algorithm (MGE) essentially allows the exchange loop to operate in both directions, i.e., forward and backward. By changing the termination criterion to have a cascade of exchange free loops in either direction, it becomes possible to alter the correspondence in any frame. The steps of the MGE algorithm are given below.

A. MGE Algorithm

- 1. Initialization:** Set the forward and backward iteration flags on. For each point in X^k , $k = 1, 2, \dots, n - 1$, determine the nearest neighbor in X^{k+1} using these nearest neighbors, and initialize m trajectories.
- 2. Forward Exchange Loop:** For each frame value $k = 2$ to $n - 1$:

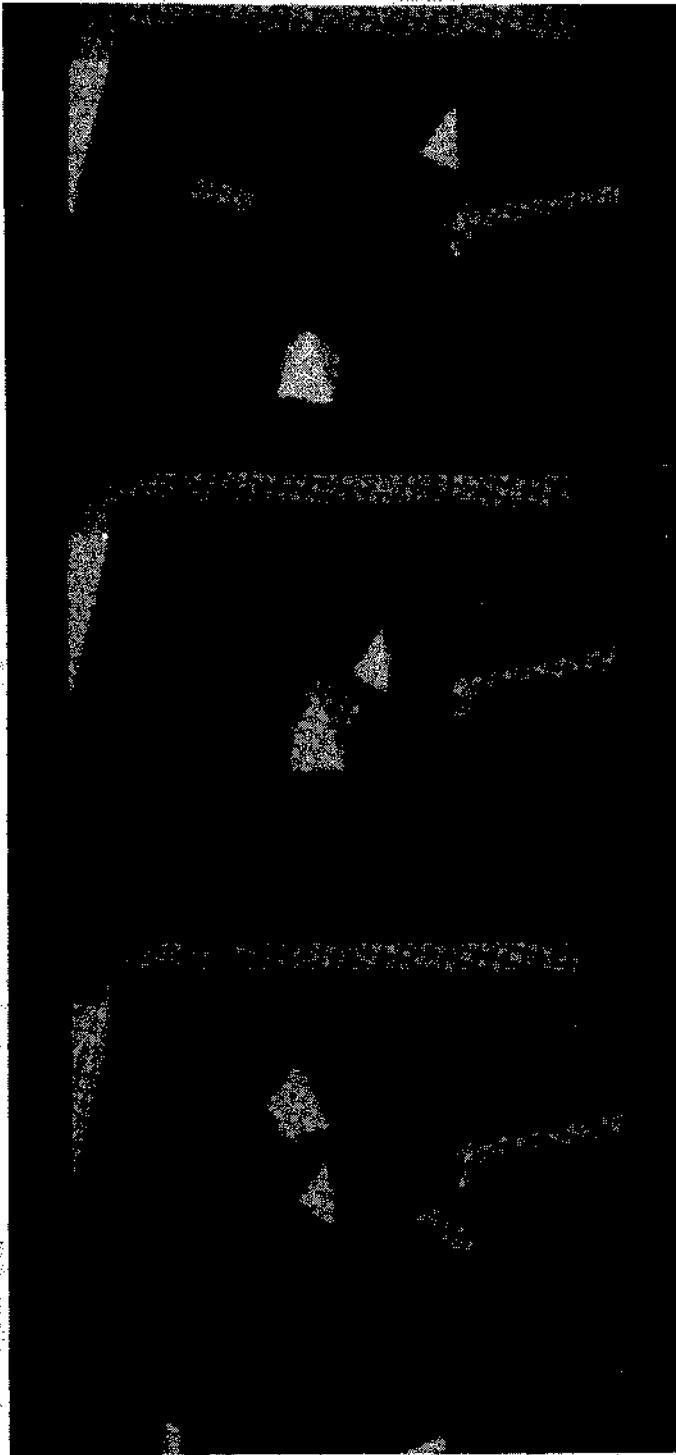


Fig. 7. Three frames of a block scene. The blocks were manually moved, and the interesting points were also manually obtained.

- a. Compute the gain g_{ij}^k for $i = 1$ to $m - 1$ and $j = i + 1$ to m .
- b. Pick the $i-j$ pair providing the maximum gain.
- c. Exchange the points in the $(k+1)$ th frame on the T_i and T_j trajectories, and set the exchange flag on.
- d. Check the exchange flag at the end of the loop. If any exchange was made during the entire pass, set the backward iteration flag on and go back to the beginning of the forward exchange loop.

Otherwise, clear the forward iteration flag and go to the termination check step.

3. Backward Exchange Loop: For each frame value $k = n - 1$ to 2:

- a. Go through the exchange loop similarly to above, with exchanges now being made in the $(k-1)$ th frame.
- b. Check the exchange flag at the end of the loop. If any exchange was made, then set the forward iteration flag on and go back to the beginning of

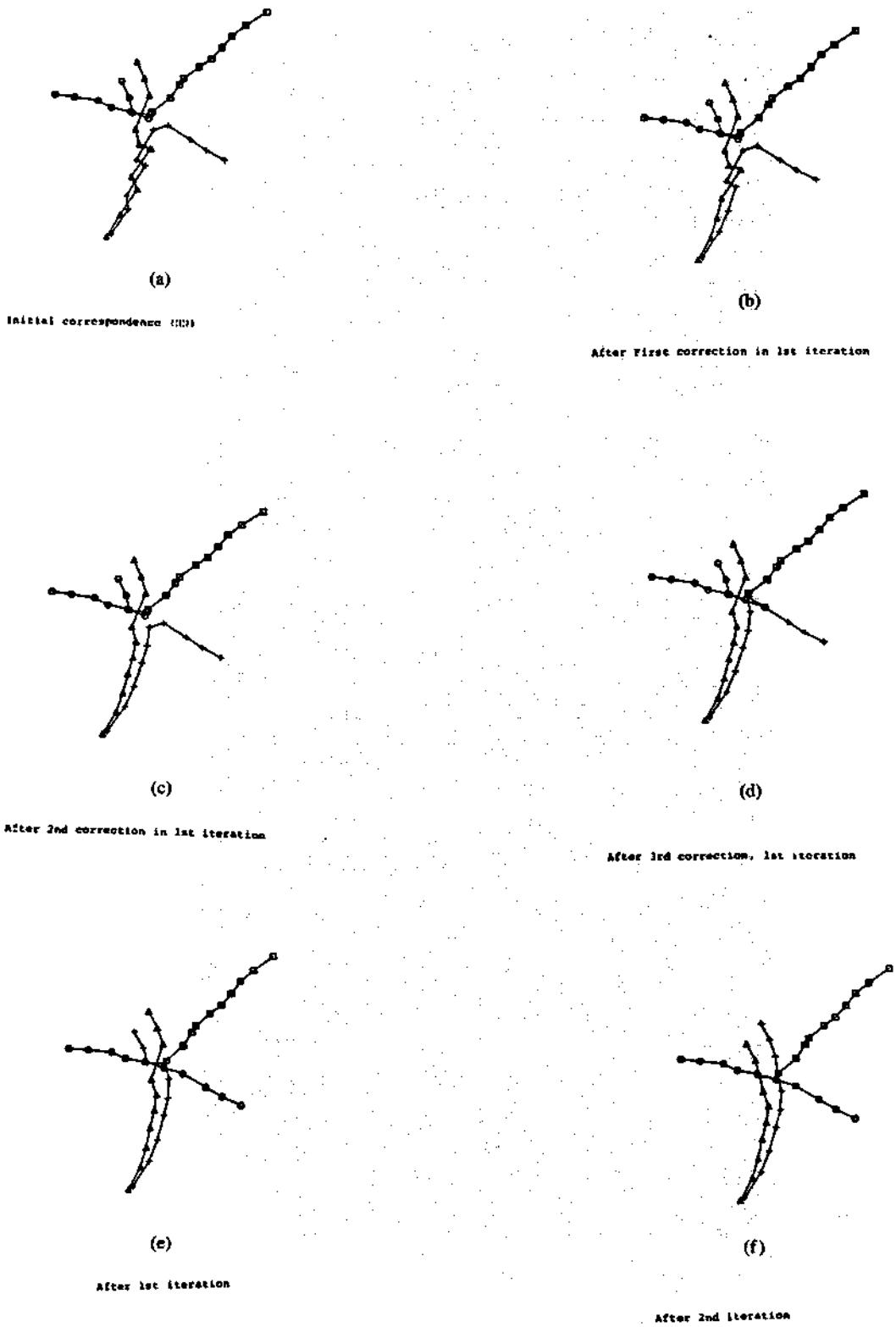


Fig. 8. The trajectories of four points in the block scene. We considered ten frames.

the loop. Otherwise, clear the backward iteration flag and go to the termination check step.

4. **Termination Check:** Check the forward and backward iteration flags. If both are off, then stop; otherwise, go to the (forward/backward exchange) loop for which the corresponding flag is on.

Fig. 13(a) shows the trajectories which were obtained after the iterations ended in the forward exchange loop. Fig. 13(b) shows the trajectories which were available at the end of iterations in the backward exchange loop. The annoying correspondence of the first two frames has clearly been changed. The final and correct trajectories

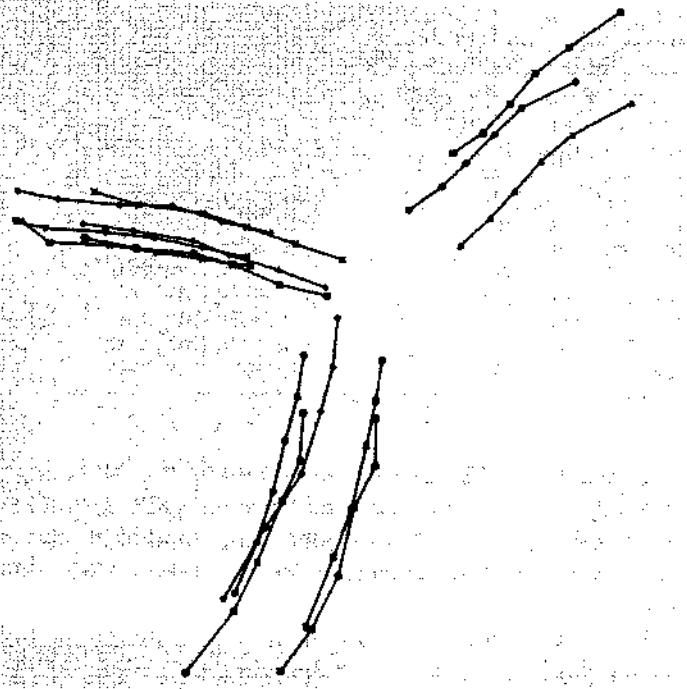


Fig. 9. Trajectories for all interesting points in six frames for the block scene.

are shown in Fig. 13(c), which required 22 iterations in the forward direction, 10 iterations in the backward direction, 8 iterations in the forward direction again, and 1 exchange-free iteration in the backward direction.

X. OCCLUSION

When working with a large sequence of frames, it is possible that some objects may disappear totally or partially. Similarly, some new objects may appear in the frame sequence from some intermediate frame onward. In order to handle the problem of occlusion in an effective way, it is necessary that the system have much more information than just the number and location of feature points. For example, consider the case of two rectangles moving linearly in different planes parallel to the optical axis of the camera. Suppose the positions of these two rectangles are as shown in Fig. 14(a) in one frame and in the next frame some of the different possibilities are as shown in Fig. 14(b)-(d).

For all the cases of the next frame shown above, a corner detector will detect eight feature points, and there is no way of telling whether an occlusion has occurred or not by simply looking at the number of feature points. In fact, until and unless some assumptions about the rigidity of the feature point configurations are made or the object structural information is available, it is difficult to say that an occlusion has occurred. Thus, it is our contention that for any method of correspondence analysis to be successful with respect to occlusion, much more information than just the number and location of feature points is needed.

Does this imply that the problem of occlusion cannot be taken care of when the only information available is the set of moving points? The answer is "no," as is evident from some psychological studies [30], [24]. It has



Fig. 10. Three frames from the *Superman* sequence. The points tracked by the algorithm are on the belt and head of the three soldiers that are running towards the camera.

TABLE I
POINTS FROM THE SUPERMAN SEQUENCE

Frame	Head-1; Belt-1	Head-2; Belt-2	Head-3; Belt-3
1	(73, 306); (72, 261)	(147, 305); (140, 240)	(295, 300); (290, 256)
2	(79, 302); (78, 261)	(154, 302); (147, 239)	(291, 296); (288, 254)
3	(86, 305); (83, 259)	(171, 301); (166, 234)	(278, 296); (276, 249)
4	(95, 312); (93, 261)	(189, 303); (185, 236)	(275, 304); (273, 253)
5	(95, 309); (95, 258)	(200, 303); (196, 238)	(266, 303); (263, 250)
6	(108, 307); (106, 256)	(229, 292); (222, 222)	(254, 303); (249, 244)
7	(120, 310); (119, 254)	(249, 300); (242, 228)	(251, 307); (243, 240)
8	(129, 313); (130, 252)	(256, 298); (257, 222)	(243, 307); (240, 236)
9	(138, 313); (137, 254)	(262, 301); (262, 222)	(250, 307); (242, 233)
10	(149, 307); (147, 245)	(268, 303); (271, 223)	(256, 305); (245, 229)



Fig. 11. The trajectories obtained by the algorithm after four iterations.

been shown that the human visual system is capable of filling in the missing points by relying on the integration of information over several frames. Shaw and Ramachandran [30] have determined that for the filling in to take place, it is necessary that at least two frames on either side of the frame with missing points be available. The availability of at least two frames on either side possibly allows the hypothesize-and-test situation where the missing point position is interpolated using the previous two frames and the interpolated value is tested against the subsequent two frames.

With respect to the concepts of path coherence and smoothness of motion, the findings of Shaw and Ramachandran are not surprising in the sense that with five frames the interpolated value of the feature point appears in all three local path coherence measures, which are integrated to verify the filling in by checking the overall smoothness of the motion. Thus, within the framework of feature point location alone, it is possible to hypothesize and test the occurrence of occlusion with the constraint that any change in the number of feature points signifies occlusion. We have found the following hypothesize-and-test occlusion algorithm useful in several experiments. The algorithm is discussed with respect to occlusion, but can be modified easily to take care of disocclusion.

A. Hypothesize-and-Test Occlusion Algorithm

Assumption: If the number of feature points is the same, then it implies no occlusion. There is only one frame in which some points disappear. For simplicity, we assume that, at least on one side of the frame with missing points, there are three frames available.

1) Determine the frame with some missing points. Let it be the p th frame, and let the number of points in this frame be $m_p (< m)$. Let A represent this set of points with individual points, denoted as $A_j, j = 1, 2, \dots, m_p$.

2) Establish the correspondence using the MGE algorithm up to $(p - 1)$ frames and from the $(p + 1)$ th frame to the n th frame. As per the stated assumption, it should be possible to obtain correspondence at least on one side, say, up to $(p - 1)$ th frame.

3) For each of the m trajectories up to the $(p - 1)$ th frame obtained through correspondence, predict the trajectory points for the p th frame using forward interpolation. Let this set of predicted points be denoted by P , and let P_i represent the predicted point for the i th trajectory.

4) For each $A_j, j = 1$ to m_p , determine the ordered list of nearest neighbors from the set P of predicted points.

5) From the ordered list of nearest neighbors for every A_j , construct the sets N_1, N_2, \dots, N_m , respectively, as the set of the first nearest neighbors, the set of up to the second nearest neighbor, and through the set of up to the $(m - 1)$ th nearest neighbors. Thus, if $P_i \in N_j$, then P_i is at least the j th nearest neighbor of one of the points from set A .

6) Count the number of distinct points from the sets N_j 's for $j = 1$ to m . Stop at the set when the count value is the same as m_p , say, at the j th set. The remaining $(m - m_p)$ predicted points not in the j th set are then selected as candidate points for filling in. However, if no N_j with count value is m_p , then determine the set N_j with count values such that $\text{count}(N_j) < m_p < \text{count}(N_{j+1})$ predicted points as candidate points for filling in.

7) Complete the correspondence for n frames by using in the p th frame the given m_p points A and the candidate points from step 6).

8) For the case of multiple candidate sets for filling in, retain that candidate set which yields the best filling in measured in terms of D .

We applied the above algorithm to the frame sequence of Fig. 12. Points on the third and fourth trajectories from the fifth frame were suppressed, thus creating occlusion. The correspondence up to the fourth frame was then determined, and the points for the fifth frame were then predicted. Table II shows the location of 10 actual points from frames and 12 predicted points for frame 5. Table III gives the ordered set of nearest neighbors, while the set N_1, N_2 , and N_3 is shown in Table IV.

The candidate points for filling in are then any two points from P_4, P_{11} , and P_{12} . Fig. 15(a)-(c) shows the

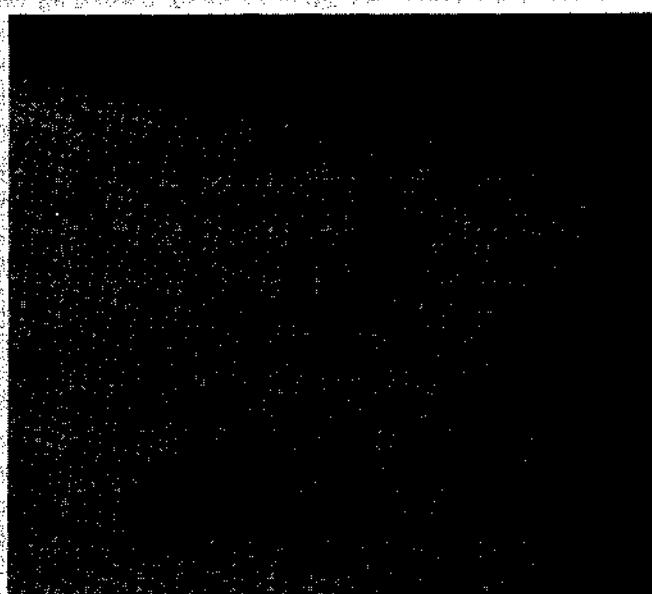
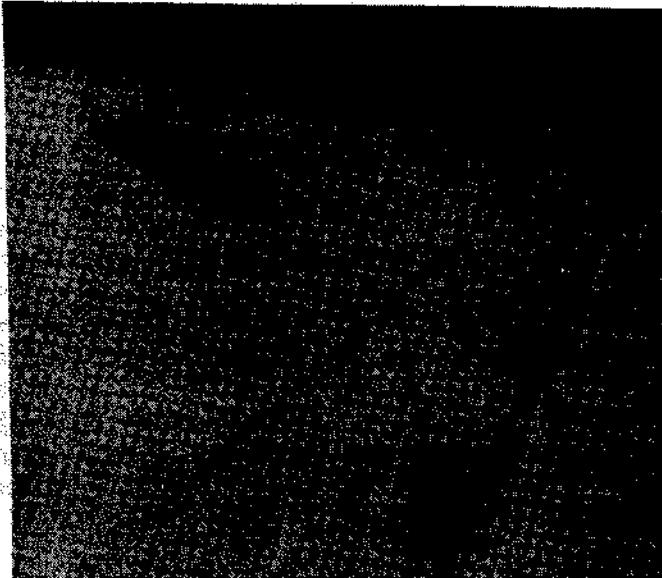
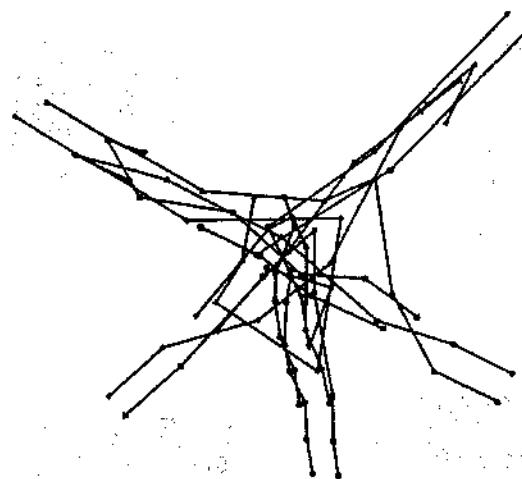
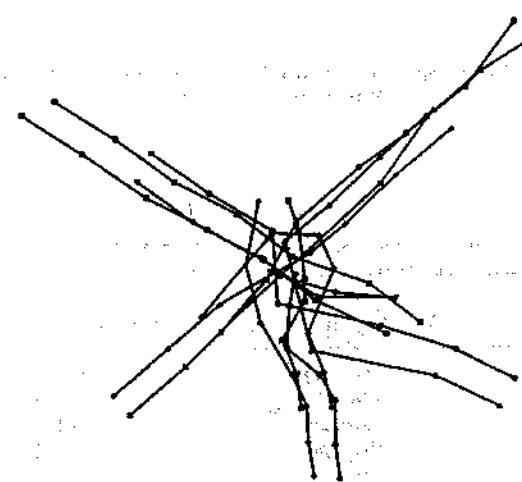


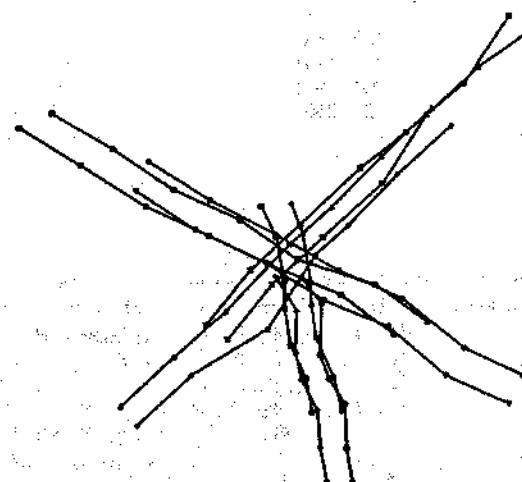
Fig. 12. Three frames of a laboratory sequence. In this sequence, the objects are of uniform color, and their shapes are also similar. At a point in the sequence, points of one object are not visible due to occlusion.



(a)



(b)



(c)

Fig. 13. The trajectories obtained by the modified algorithm. (a) The trajectory obtained after the first forward pass. (b) After the forward and backward pass. (c) After the forward, backward, and, again, forward passes.

trajectories obtained by using P_4-P_{12} , P_4-P_{11} , and $P_{11}-P_{12}$ as filling-in points, respectively.

Comparing these trajectories to the actual trajectory of Fig. 15(d), one important observation can be made, and

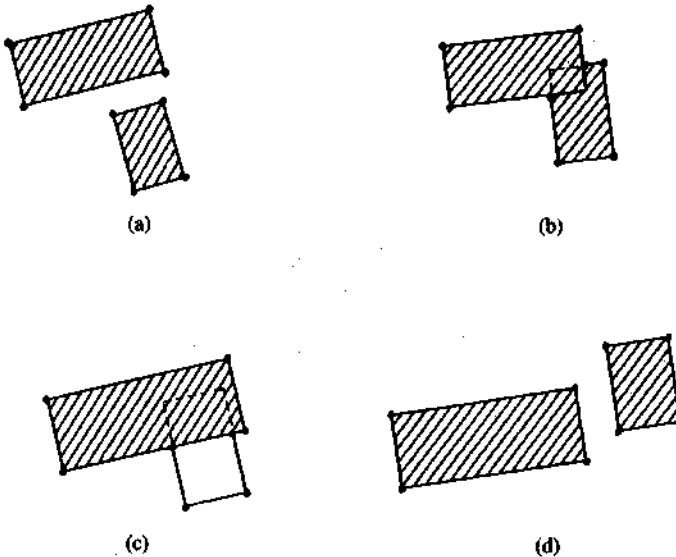


Fig. 14. A frame sequence in which there is occlusion, but the number of vertices remains same.

TABLE II
ACTUAL AND PREDICTED POSITION FOR FRAME

Trajectory Number	Actual Point Position	Predicted Point Position
1	(389, 195)	(386, 184)
2	(376, 171)	(360, 159)
3	missing	(313, 249)
4	missing	(276, 239)
5	(289, 214)	(292, 225)
6	(311, 212)	(318, 214)
7	(296, 151)	(275, 150)
8	(317, 149)	(298, 133)
9	(237, 189)	(256, 225)
10	(273, 173)	(299, 227)
11	(304, 262)	(329, 288)
12	(323, 250)	(342, 221)

TABLE III
ORDERED SETS OF NEAREST NEIGHBORS FOR A_i 'S FROM P_j 'S

A_1	$\{P_1, P_2, P_{12}, P_6, P_3, P_{10}, P_5, P_8, P_{11}, P_4, P_7, P_9\}$
A_2	$\{P_1, P_2, P_{12}, P_6, P_8, P_{10}, P_5, P_3, P_7, P_4, P_{11}, P_9\}$
A_3	$\{P_3, P_{10}, P_4, P_6, P_9, P_3, P_{12}, P_7, P_8, P_{11}, P_2, P_1\}$
A_4	$\{P_6, P_{10}, P_5, P_{12}, P_3, P_4, P_9, P_7, P_2, P_{11}, P_1, P_8\}$
A_5	$\{P_8, P_7, P_2, P_6, P_5, P_{10}, P_{12}, P_9, P_4, P_1, P_3, P_{11}\}$
A_6	$\{P_8, P_7, P_2, P_6, P_{12}, P_1, P_5, P_{10}, P_9, P_4, P_3, P_{11}\}$
A_7	$\{P_9, P_7, P_4, P_5, P_{10}, P_8, P_6, P_3, P_{12}, P_2, P_{11}, P_1\}$
A_8	$\{P_7, P_8, P_9, P_5, P_{10}, P_6, P_4, P_{12}, P_3, P_2, P_1, P_{11}\}$
A_9	$\{P_3, P_{10}, P_{11}, P_4, P_5, P_6, P_{12}, P_9, P_1, P_7, P_2, P_3\}$
A_{10}	$\{P_3, P_{10}, P_{12}, P_6, P_{11}, P_5, P_4, P_9, P_1, P_2, P_7, P_8\}$
A_{11}	$\{P_3, P_{10}, P_{12}, P_6, P_{11}, P_5, P_4, P_9, P_1, P_2, P_7, P_8\}$
A_{12}	$\{P_3, P_{10}, P_{12}, P_6, P_{11}, P_5, P_4, P_9, P_1, P_2, P_7, P_8\}$

TABLE IV

N_1	$\{P_1, P_3, P_5, P_6, P_7, P_8, P_9\}$	Count = 7
N_2	$\{P_1, P_2, P_3, P_5, P_6, P_7, P_8, P_9, P_{10}\}$	Count = 9
N_3	$\{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}\}$	Count = 12

that is, irrespective of the candidate points for filling in, that the overall perception of the trajectories in limited occlusion is correct.

XI. OBJECT TRACKING VIA CENTROIDS

We mentioned earlier that the present approach for correspondence can be used to track objects by representing each moving object as a point, possibly the centroid. Since this object representation is very coarse, it is not necessary to perform a precise segmentation. It should be enough if every moving object can be located approximately in a frame. Difference pictures offer one such convenient way of roughly locating moving objects [13], [40]. Let the image sequence be represented as $F_1(x, y), F_2(x, y), \dots, F_N(x, y)$. A difference picture $D_{ik}(x, y)$ is then obtained as

$$D_{ik}(x, y) = F_{i+k}(x, y) - F_i(x, y).$$

If the objects are darker than the background, a simple thresholding such as

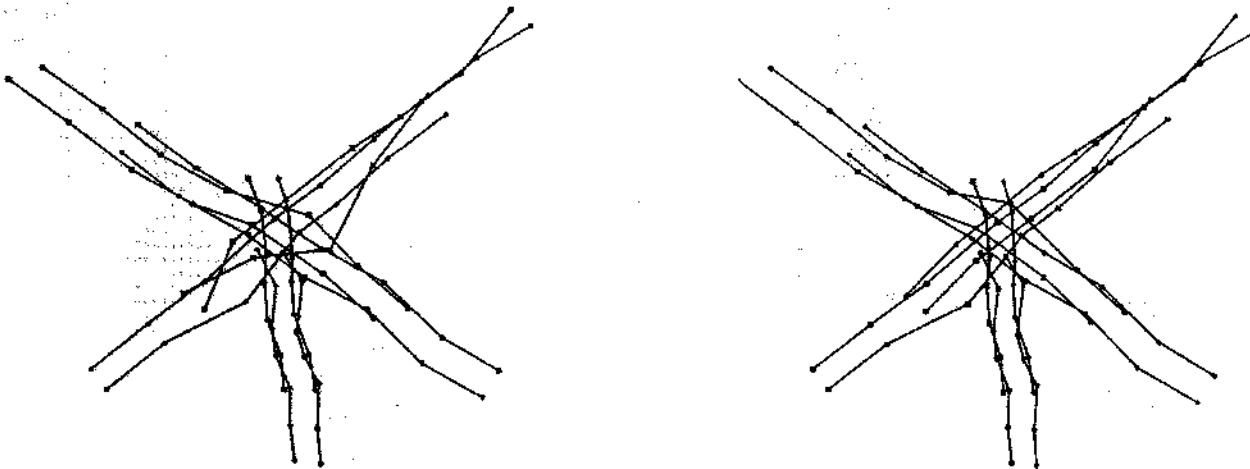
$$\begin{aligned} D_{ik}(x, y) &= 1 && \text{if } D_{ik}(x, y) \geq T \\ &= 0 && \text{otherwise.} \end{aligned}$$

can then detect most of the moving object region in the frame F_i , provided k is suitably chosen. The threshold value T can be determined by looking at the difference picture histogram.

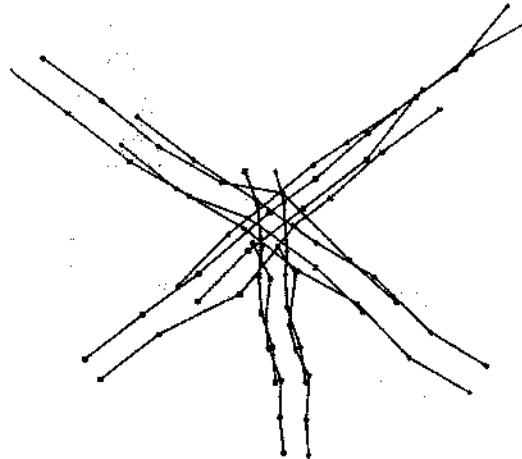
This difference picture computation scheme was applied to the image sequence of Fig. 7, which contains three moving objects with partial occlusion taking place in one of the frames. Since there was sufficient displacement of objects in successive frames, the value for k was taken to be 1. A threshold value of 20 for T was found to yield satisfactorily portions of moving object regions from different frames. Fig. 16 shows portions of interest from four thresholded difference pictures using frames F_3 and F_7 . After performing the connectivity analysis and discarding regions with an area smaller than 20 pixels, moving object regions from each frame were extracted. Except for one difference picture, all others yielded three moving regions. Centroid values of these regions in different thresholded difference pictures were then used as the input data to the hypothesize-and-test occlusion algorithm. The trajectories obtained by this algorithm are shown in Fig. 17 where a big circle around one point indicates that the circled point was predicted. Thus, even by doing a crude segmentation, it is possible to track moving objects based solely on the location of one representative point per object.

XII. CONCLUSION

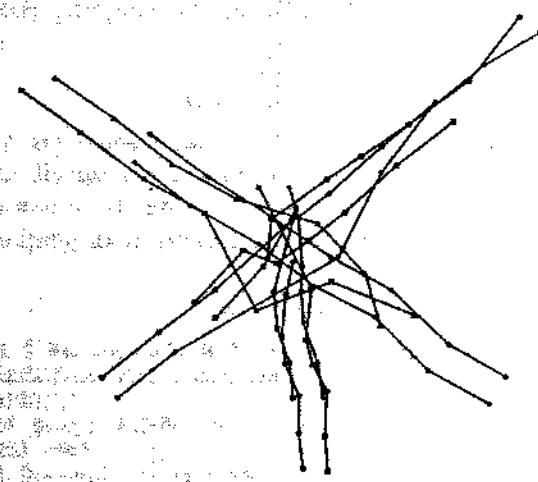
In this paper, we show that by considering an extended frame sequence, one can use coherence in spatio-temporal properties for establishing correspondence to obtain trajectories of objects. Quasi-dynamic approaches for the analysis of motion try to solve all problems in a *minimum* number of frames. Obviously, the limited amount of data



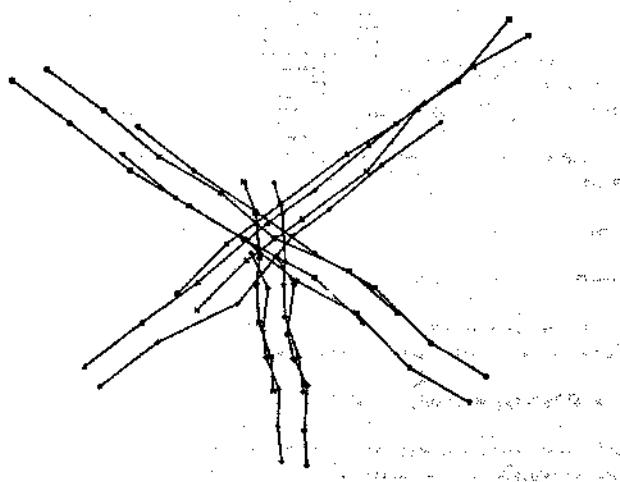
(a)



(b)



(c)



(d)

Fig. 15. The trajectories obtained by the occlusion algorithm for the hypothesized candidate pairs. (a) P_4-P_{12} , (b) P_4-P_{11} , (c) $P_{11}-P_{12}$, (d) The actual trajectories.

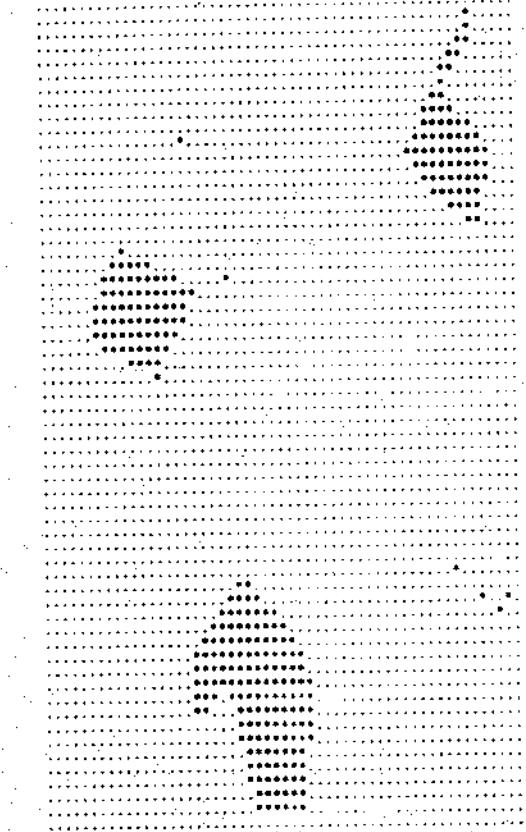
in the temporal domain necessitates some assumptions on the nature of objects. More importantly, such approaches require methods that are complex and sensitive to noise. By considering *dynamic scenes*, we can use techniques that can successively refine the information recovered from the sequence. Such methods are much less sensitive to noise and can usually be performed very fast. The smoothness of motion allows us to establish correspondence in case of occlusion of points also, in a limited situation. This is very encouraging because in most approaches to occlusion analysis one requires information about features also.

In this paper, we considered only the correspondence problem. After the trajectories are established, one may try to recover the structure of objects. As shown in [31], [28], one may use parameters of curves representing trajectories to obtain the motion characteristics and structure

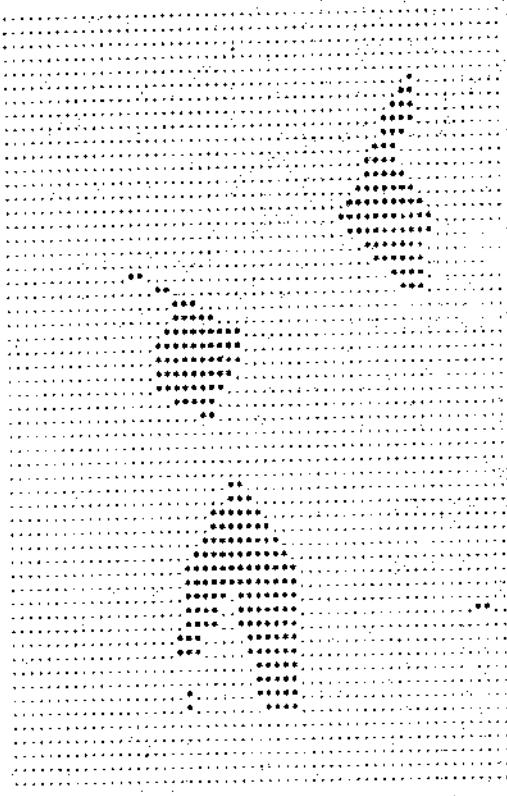
of objects. It seems that one may use successive refinement in the recovery of the structure also.

The proposed approach assumes that the motion is smooth. Indeed, in many applications, one is required to detect points of discontinuities in motion. The discontinuities in motion play an important role in the analysis and verbalization of the motion of objects. We are studying techniques for detecting discontinuities in motion characteristics and their use in describing the motion of objects [7].

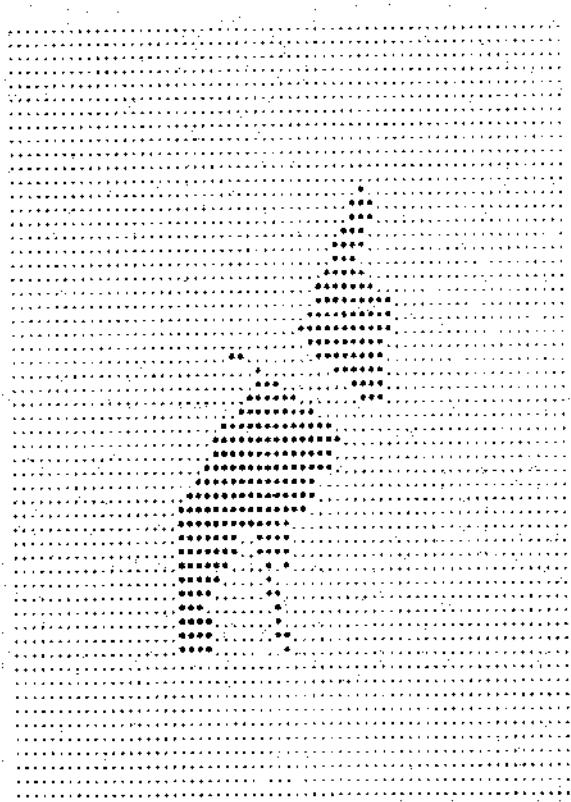
It should be mentioned here that in our algorithm we did not use any characteristics of points. The trajectories were determined using only locations of points in frames. The robustness and computations can be improved by considering pictorial characteristics of points in frames. This additional information may play a very important role in the initialization phase of the algorithm by eliminating



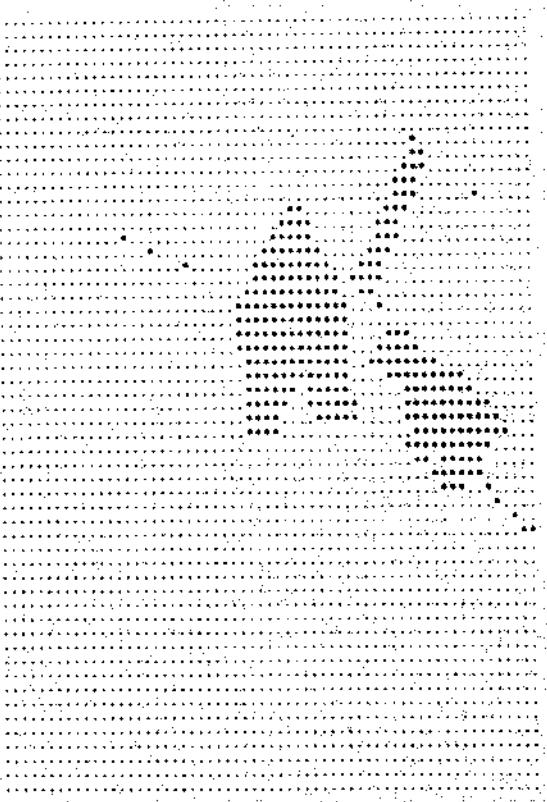
(a)



(b)



(c)



(d)

Fig. 16. The difference pictures for the sequence of Fig. 7. (a) D_{34} . (b) D_{45} . (c) D_{56} . (d) D_{67} . Note that due to occlusion D_{56} has only two regions.

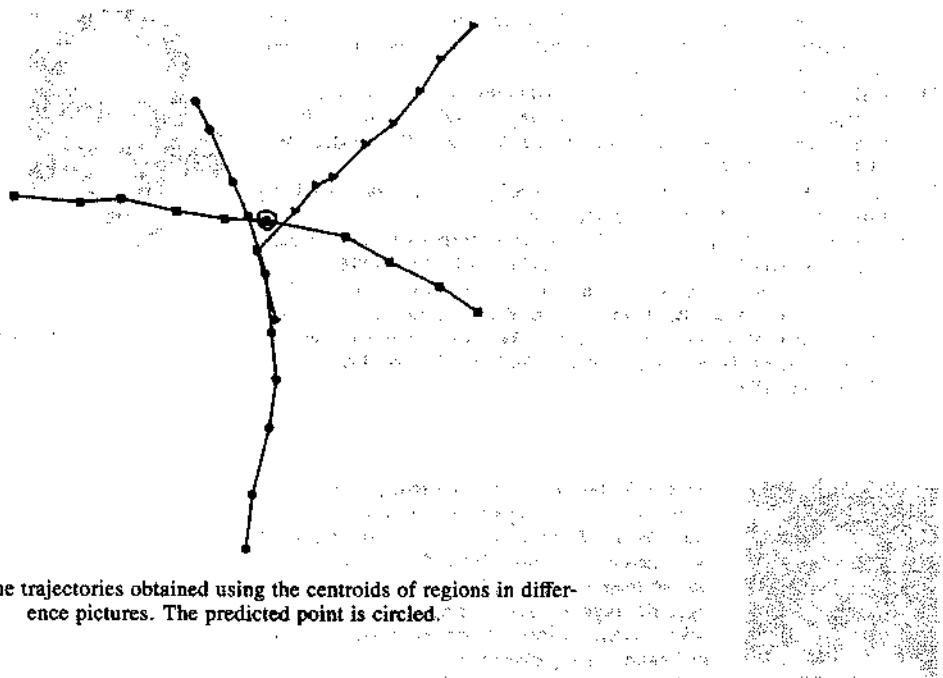


Fig. 17. The trajectories obtained using the centroids of regions in difference pictures. The predicted point is circled.

unlikely groupings of points, even if they are close to each other.

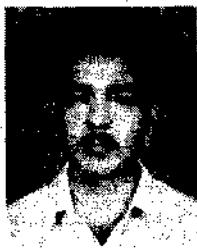
ACKNOWLEDGMENT

We are thankful to P. Besl and S. Haynes for many useful discussions and criticisms during the course of this work and to M. Shah, C. Born, V. Salari, and S. Vemuri for helping us in performing experiments.

REFERENCES

- [1] S. T. Barnard and W. B. Thompson, "Disparity analysis of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, pp. 333-340, 1980.
- [2] M. Brady, "Parallelism in vision," *Artif. Intell.*, vol. 21, pp. 271-283, 1983.
- [3] J. Donner, J. S. Lappin, and G. Perfetto, "Detection of three-dimensional structure in moving optical patterns," *J. Exper. Psychol.: Human Perception Performance*, vol. 10, no. 1, pp. 1, 1984.
- [4] L. S. Dreschler and H.-H. Nagel, "Volumetric model and 3-D trajectory of a moving car derived from monocular TV-frame sequences of a street scene," *Comput. Graphics Image Processing*, vol. 20, pp. 199-228, 1982.
- [5] B. E. Flinchbaugh and B. Chandrasekaran, "A theory of spatio-temporal aggregation for vision," *Artif. Intell.*, vol. 17, pp. 387-408, 1981.
- [6] J. J. Gibson, *The Ecological Approach to Visual Perception*. Boston, MA: Houghton Mifflin, 1979.
- [7] S. M. Haynes and R. Jain, "Low level motion event: Trajectory discontinuities," in *Proc. First Conf. AI Appl.*, pp. 251-256.
- [8] E. C. Hildreth, "Computations underlying the measurement of visual motion," *Artif. Intell.*, vol. 23, pp. 309-354, 1984.
- [9] D. D. Hoffman and B. E. Flinchbaugh, "The interpretation of biological motion," M.I.T., Cambridge, MA, AI Memo. 608; 1980.
- [10] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185-203, 1981.
- [11] R. Jain, "Direct computation of the focus of expansion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-5, pp. 58-64, 1983.
- [12] R. Jain and S. Haynes, "Imprecision in computer vision," *IEEE Comput.*, Aug. 1982.
- [13] R. Jain and H.-H. Nagel, "On the analysis of difference and accumulative difference pictures in dynamic scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, pp. 206-214, 1979.
- [14] C. P. Jerian and R. Jain, "Determining motion parameters for scenes with translation and rotation," in *Proc. Workshop Motion: Representation Contr.*, Toronto, Ont., Canada, Apr. 4-6, 1983.
- [15] M. Jenkin, "Tracking three dimensional moving light displays," in *Proc. Workshop Motion: Representation Contr.*, Toronto, Ont., Canada, 1983, pp. 66-70.
- [16] G. Johansson, "Spatio-temporal differentiation and integration in visual motion perception," *Psych. Res.*, vol. 38, pp. 379-383, 1976.
- [17] D. Marr, "Early processing of visual information," *Phil. Trans. Roy. Soc. London B*, vol. 275, pp. 483-524, 1976.
- [18] K. M. Mutch and W. B. Thompson, "Analysis of accretion and deletion at boundaries in dynamic scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, pp. 133-138, Mar. 1985.
- [19] U. Neisser, *Cognition and Reality*. San Francisco, CA: Freeman, 1976.
- [20] N. G. O'Brien and R. Jain, "Axial motion stereo," in *Proc. Workshop Comput. Vision*, Annapolis, MD, 1984.
- [21] J. O'Rourke and N. I. Badler, "Model-based image analysis of human motion using constraint propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, pp. 522-536, 1980.
- [22] K. Prazdny, "Egomotion and relative depth map from optical flow," *Biol. Cybernet.*, vol. 36, pp. 87-102, 1980.
- [23] J. M. Prager and M. A. Arbib, "Computing the optic flow: The MATCH algorithm and prediction," *Comput. Vision, Graphics, Image Processing*, vol. 24, pp. 271-304, 1983.
- [24] V. S. Ramachandran and S. M. Anstis, "Extrapolation of motion path in human visual perception," *Vision Res.*, vol. 23, pp. 83-85, 1984.
- [25] R. F. Rashid, "Towards a system for the interpretation of moving light displays," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, pp. 574-581, 1980.
- [26] J. H. Rieger and D. T. Lawton, "Determining the instantaneous axis of translation from optic flow generated by arbitrary sensor motion," in *Proc. Workshop Motion*, Toronto, Ont., Canada, 1983, pp. 33-41.
- [27] J. W. Roach and J. K. Aggarwal, "Determining the movement of objects from a sequence of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, pp. 554-562, Nov. 1980.
- [28] I. K. Sethi and R. Jain, "Determining three dimensional structure of rotating objects using a sequence of monocular views," Dep. Comput. Sci., Wayne State Univ., Detroit, MI, Tech. Rep., 1983.
- [29] M. A. Shah and R. Jain, "Detecting time varying corners," *Comput. Vision, Graphics, Image Processing*, vol. 28, pp. 345-355, 1984.
- [30] G. L. Shaw and V. S. Ramachandran, "Interpolation during apparent motion," *Perception*, vol. 11, pp. 491-494, 1982.
- [31] J. T. Todd, "Visual information about rigid and nonrigid motion: A geometric analysis," *J. Exper. Psychol. Human Perception Performance*, vol. 8, pp. 238-252, 1982.
- [32] R. Y. Tsai and T. S. Huang, "Estimating three-dimensional motion parameters of a rigid planar patch," in *Proc. PRIP*, 1981, pp. 94-97.
- [33] —, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," in *Proc. IEEE Conf. Pattern Recognition Image Processing*, 1982, pp. 112-118.
- [34] R. Y. Tsai, T. S. Huang, and W. L. Zhu, "Estimating three-dimen-

- sional motion parameters of a rigid planar patch, II: Singular value decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 525-534.
- [35] J. K. Tsotsos, "A framework for visual motion understanding," *Dep. Comput. Sci., Univ. Toronto, Tech. Rep. CSRG-114*, June 1980.
- [36] S. Ullman, *The Interpretation of Visual Motion*. Cambridge, MA: M.I.T. Press, 1979.
- [37] A. M. Waxman, "An image flow paradigm," in *Proc. Workshop Comput. Vision*, Annapolis, MD, 1984.
- [38] J. A. Webb and J. K. Aggarwal, "Structure from motion of rigid and jointed objects," *Artif. Intell.*, vol. 19, pp. 107-130, 1982.
- [39] T. D. William, "Depth from motion in real world scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, pp. 511-516, 1980.
- [40] M. Yachida, M. Asada, and S. Tsuji, "Automatic analysis of moving images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, pp. 12-19, Jan. 1981.



Ishwar K. Sethi (M'78) is currently an Associate Professor in the Department of Computer Science, Wayne State University, Detroit, MI, which he joined in September 1982. Prior to that he was on the faculty at the Indian Institute of Technology, Kharagpur, India, for eleven years. His research interests include computer vision, image processing, and pattern recognition.

Dr. Sethi is a member of the Association for Computing Machinery.



Ramesh Jain (M'79-SM'83) received the B.E. degree from Nagpur University, India, in 1969, and the Ph.D. degree from the Indian Institute of Technology, Kharagpur, India, in 1975.

He is an Associate Professor of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. His current research interest is in computer vision and artificial intelligence. Formerly he worked at Wayne State University, Detroit, MI, the University of Texas, Austin, the University of Hamburg, West Germany, and the Indian Institute of Technology.

Dr. Jain is a member of the Association for Computing Machinery, the American Association for Artificial Intelligence, the Pattern Recognition Society, and the Cognitive Science Society.

Generalizing Epipolar-Plane Image Analysis on the Spatiotemporal Surface

H. HARLYN BAKER
ROBERT C. BOLLES

Artificial Intelligence Center, SRI International, 333 Ravenswood Avenue Menlo Park, CA 94025

Abstract

The previous implementations of our *Epipolar-Plane Image Analysis* mapping technique demonstrated the feasibility and benefits of the approach, but were carried out for restricted camera geometries. The question of more general geometries made the technique's utility for autonomous navigation uncertain. We have developed a generalization of our analysis that (a) enables varying view direction, including variation over time (b) provides three-dimensional connectivity information for building coherent spatial descriptions of observed objects; and (c) operates sequentially, allowing initiation and refinement of scene feature estimates while the sensor is in motion. To implement this generalization it was necessary to develop an explicit description of the evolution of images over time. We have achieved this by building a process that creates a set of two-dimensional manifolds defined at the zeros of a three-dimensional spatiotemporal Laplacian. These manifolds represent explicitly both the spatial and temporal structure of the temporally evolving imagery, and we term them *spatiotemporal surfaces*. The surfaces are constructed incrementally, as the images are acquired. We describe a tracking mechanism that operates locally on these evolving surfaces in carrying out three-dimensional scene reconstruction.

Introduction

In an earlier publication in this journal [1], we described a sequence analysis technique developed for use in obtaining depth estimates for points in a static scene. The approach bridged the usual dichotomy of passive depth sensing in that its large number of images led to a large baseline and thus high accuracy, while rapid image sampling gave minimal change from frame to frame, eliminating the correspondence problem. Rather than choosing quite disparate views and putting features into correspondence by stereo matching, with this technique we chose to process massive amounts of similar data, but with much simpler and more robust techniques. The technique capitalized on several constraints we could impose on the image acquisition process, namely:

1. The camera moved along a linear path.

2. It acquired images at equal spacing as it moved.
3. The camera's view was orthogonal to its direction of travel.

With these constraints, we could guarantee that

1. Individual scene features would be observed in single epipolar planes over the period of scanning.
2. Images of these planes could be constructed by collecting corresponding image scan lines in successive frames.
3. The motion of scene features in these images would appear as linear tracks.

We termed these image planes *epipolar-plane images*, or EPIs, and the process *Epipolar-Plane Image Analysis*.

1.2 Problems with the Previous Approach

In that earlier paper we commented on our previous dissatisfactions with the approach, and the limitations that would restrict its usefulness. Summarizing, the limitations were

- L_1 Orthogonal viewing would preclude many of the camera attitudes one would expect to be necessary for an autonomous vehicle—notably that attitude in which the vehicle is looking along its direction of motion, or when it is to track some particular feature and follow it while moving across the scene.
- L_2 A constant rate of image acquisition would be difficult to guarantee, and probably not be desirable in a general context. Sampling rates will be affected heavily by computational demands on the system, and vehicle velocities may be dictated by higher-level concerns.
- L_3 A linear path would be an unacceptable or highly improbable trajectory in most every situation except extended flight.
- L_4 Static scenes are the *least* likely—winds blow, clouds move; often a moving object in a scene is the one of most interest.

The dissatisfactions were

- D_1 The analysis should proceed sequentially as the imagery is acquired. To insist that all data be available before scene measurement can begin would eliminate one of the principal goals of the process—to provide timely information for a vehicle in motion.
- D_2 The EPI partitioning, through its selection of the temporal over the spatial analysis of images, could not provide spatially coherent results. It produced point sets. We attempted clustering operations on these, but were never satisfied with such a post hoc approach. The proper approach to obtaining spatial coherence in our results would begin with not losing it in the first place.

2 New Approach to EPI Analysis

2.1 Generalizations

We have developed generalizations to our earlier approach that enable us to resolve L_1 , L_2 , D_1 , and

D_2 . Arbitrary and varying camera attitudes and velocities are permissible in our new formulation, and we process the data sequentially as acquired, forming estimates, of increasing precision, descriptive of spatial contours rather than points. The generalizations also suggest a mechanism for dealing with the nonlinear path issue of L_3 . Although we have not pursued this as yet, in section 3.4 we outline an approach consistent with our EPI analysis.

L_4 rises as an incompatibility between our performance desires for a vision system and our definition of the task we choose to address. We wish to build three-dimensional descriptions of scenes, and it is inappropriate to expect this to be possible if our view of the scene is undergoing change unrelated to our active pursuit of observations. In our previous publication we discussed this motion issue, and suggested means to recognize its presence in a scene. Once distinguished from the static elements, it would be possible to invoke higher-order models and filters to estimate these objects' dynamics (as done by Broida and Chellappa [2] and Gennery [3], but our current interest is in modeling static structure.

It is worth repeating this to clarify our goals in the current work. We are not working with changing scenes, nor is our aim to build descriptions of moving or deforming objects. Our camera is all that moves, and any changes in the imagery arise strictly from this movement. Our goal is to model the geometry of a real static scene through which the camera is moving. This distinguishes us from most of the current efforts in spatiotemporal analysis that use image-plane velocities for measuring arbitrary flows (for example Heeger [7]), or that combine the measured flow with assumptions of constant 3D motion and rigidity for estimating known-order analytic surfaces (i.e., Waxman and Wohn [4], Waxman et al. [5], and Subbarao [6]).

In common with our earlier work, our new approach involves the processing of a very large number of images acquired by a moving camera. The analysis is based on three constraints:

1. The camera's movement is restricted to lie along a linear path.
2. The camera's position and attitude at each imaging site are known.

3. Image capture is rapid enough with respect to camera movement and scene scale to ensure that the data is, in general, temporally continuous.

Within this framework, we generalize from the traditional notion of epipolar lines to that of epipolar planes—a set of epipolar lines sharing a property of transitivity (which we discuss in section 2.2). We formulate a tracking process that exploits this property for determining the position of features in the scene. This tracking occurs on

what we term the *spatiotemporal surface*—a surface defining the evolution of a set of scene features over time. Critical to visualizing this space-time approach is obtaining an understanding of the geometry of the sensing situation, and this is described in the next section.

2.2 Geometric Considerations of Camera Path and Attitude

Figure 1 shows the geometry pertaining to our analysis, indicating several imaging positions

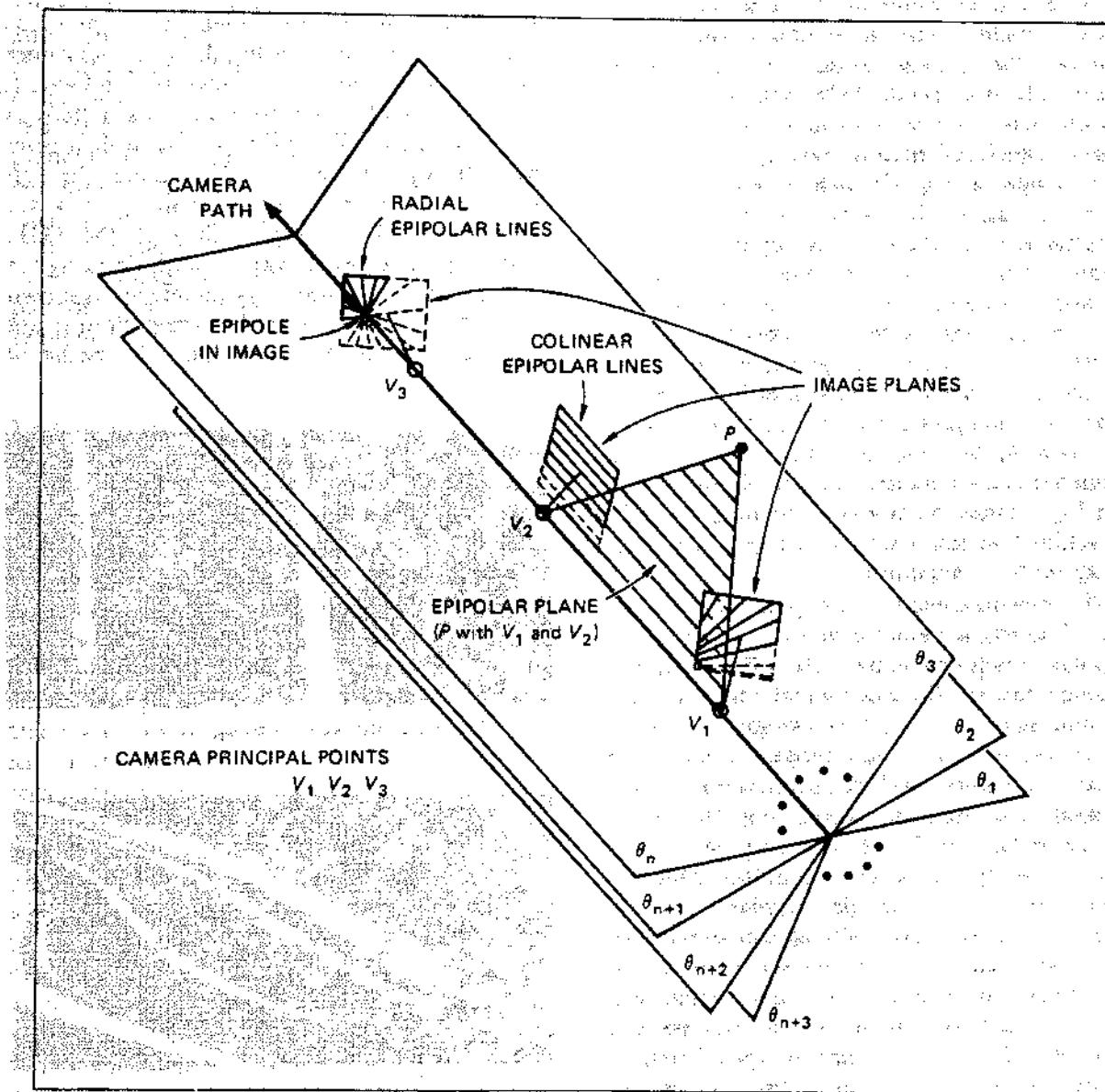


Fig. 1. General epipolar configuration.

and attitudes along a straight path. The camera is modeled as a pin-hole with image plane in front of the lens. For each feature P in the scene and two viewing positions such as V_1 and V_2 , there is an *epipolar plane* that passes through P and the line joining the two lens centers. This plane intersects the two image planes along corresponding *epipolar lines* (note that, here, intersection and projection are, in a sense, equivalent). An *epipole* is the intersection of an image plane with the line joining the lens centers. In motion analysis, an epipole is often referred to as the *focus of expansion* (FOE) because the epipolar lines radiate from it. The camera moves in a straight line, and the lens centers at the various viewing positions lie along this line. Notice that the FOE is the image of the camera path. This structuring divides the scene into a pencil of planes passing through the camera path, several of which are sketched ($\theta_1, \theta_2, \theta_3, \theta_n, \dots, \theta_{n+3}$). This pencil is crucial to our analysis. We view the space as a cylindrical coordinate system with axis the camera path, angle defined by the epipolar plane, and radius the distance from the axis. Note that a scene feature is restricted to a single epipolar plane, and any scene features at the same angle (within the discretization) share that plane. This means that, as in our earlier work, the analysis of a scene can be partitioned into a set of analyses, one for each plane, and these planes can be processed independently. In section 3 we describe how we organize the data to exploit this constraint.

With viewing direction orthogonal to the direction of travel, as depicted at V_2 in figure 1, the epipolar lines for a feature such as P are horizontal scan lines, and these occur at the same vertical position (scan line) in all the images. This is the camera geometry normally chosen for computer stereo vision work. Each scan line is a projected observation of the features in an epipolar plane. The projection of P onto these epipolar lines moves to the right as the camera moves to the left. If one were to take a single epipolar line (scan line) from each of a series of images obtained with this camera geometry and compose a spatiotemporal image, with horizontal being spatial and vertical being temporal, one would see a pattern as in the EPI of figure 2. For this type of motion, feature trajectories are straight lines, as can be



Fig. 2. Orthogonal viewing.

seen. This is the case handled by our previous analysis. If, on the other hand, the camera were moving with an attitude as shown at V_1 in figure 1, the set of epipolar lines would form a pattern as shown in figure 3. For this type of motion, feature trajectories are hyperbolas. Notice that the epipolar lines are no longer scan lines—they are oriented radially and pass through the FOE. Allowing the camera to vary its attitude along the path gives rise to spatiotemporal images as shown in figure 4. Here, the epipolar line pattern is not fixed from frame to frame, and the paths of



Fig. 3. Fixed, nonorthogonal viewing.



Fig. 4. View direction varying.

features in the EPI are neither linear nor hyperbolic—in fact they are arbitrary curves.

The transitivity property mentioned in section 2.1 arises from the fact that any pair of lines selected from the set form a corresponding pair. That is, for the set of epipolar lines $E^0 = (e_0^0, e_1^0, \dots, e_n^0)$ from epipolar plane θ over images I_0 through I_n , any two members comprise a pair of corresponding epipolar lines— e_0^0 with e_1^0 , e_1^0 with e_2^0 , et cetera. This occurs because the camera's linear path guarantees that a single pencil of planes defines the epipolar mapping over the entire sequence. Thus, any mapping done on the basis of e_0^0 with e_1^0 and then e_1^0 with e_2^0 implies the mapping of e_0^0 with e_2^0 . A similar argument holds for all pairs of mappings in E^0 , and the transitivity follows. If the camera path were nonlinear, no single pencil of planes could be defined, and no such set E^0 could be formed. The only complicating detail with the varying-attitude case (as indicated in figure 4) is that the pattern of epipolar lines changes from image to image: For a fixed camera attitude the pattern is the same for all images in the sequence.

2.3 Keeping the Problem Linear

Recall that our goal is to determine the position of stationary features in the scene: We do this by tracking their appearance over time as they project onto these epipolar planes. Obviously in the case of orthogonal viewing (e.g., as in figure 2 and at V , in figure 1), the tracking is linear. For general camera attitudes, including varying, it is nonlinear. Computational considerations make it extremely advantageous for the tracking to be posed as a linear problem. To maintain the linearity regardless of viewing direction, we find not linear feature paths in the EPIs (figures 2 through 4), but linear paths in a *dual space*. The insight here (introduced by Marimont [8]) is that no matter where a camera roams about a scene, for any particular feature, the *lines of sight* from the camera's principle point through that feature in space all intersect at the feature (modulo the measurement error). A line of sight is determined by the line from the principal point through the point in the image plane where the projected feature is ob-

served. From mathematical duality, the duals of these lines of sight lie along a line whose dual is the scene point (see figure 5); fitting a point to the lines of sight is a linear problem. This, then, gives us a metric for linear tracking of features: We map feature image coordinates to lines of sight, and use an optimal estimator to determine the point that minimizes the variance from those lines of sight.

Our estimation is done in the scene Cartesian space, not the dual space, because the error metric, nonlinear in the dual space, has more intuitive meaning and better behavior in scene space. The estimated error in each observation is a function of the size of the Gaussian filter employed and the distance of the feature from the camera. We currently model only these uncertainties in image-plane observations, and not others related to the strength of the feature signal or uncertainty in the position of the camera. These others will have to be modeled in a complete solution.

2.4 Transformations Required

Having decided on a representation that restores the linearity of our estimator, we must now demonstrate a mechanism for extracting the feature observations from the individual images in which they occur and grouping them by epipolar plane. Only in the case of viewing angle orthogonal to the motion is this grouping simple (figure 2), and this was the case our earlier work addressed. To obtain this structuring in the general cases, we could take one of two approaches. The first is to transform the images from the Cartesian space in which they are sampled to an epipolar representation (as has been done by Baker et al. [9] and by Jain et al. [10]). Because of aliasing effects (particularly on the observation variances) and nonlinearities in the mapping (it is singular when the FOE is in the image, and could require an infinite imaging surface for the reprojection), we prefer to avoid this transformation. Probably the best solution would be to use a sensor that delivers the data directly in the epipolar form—a spherical sensor having meridian scanning would accomplish this (the flow geometry that is the basis for such a sensor was discussed by Gib-

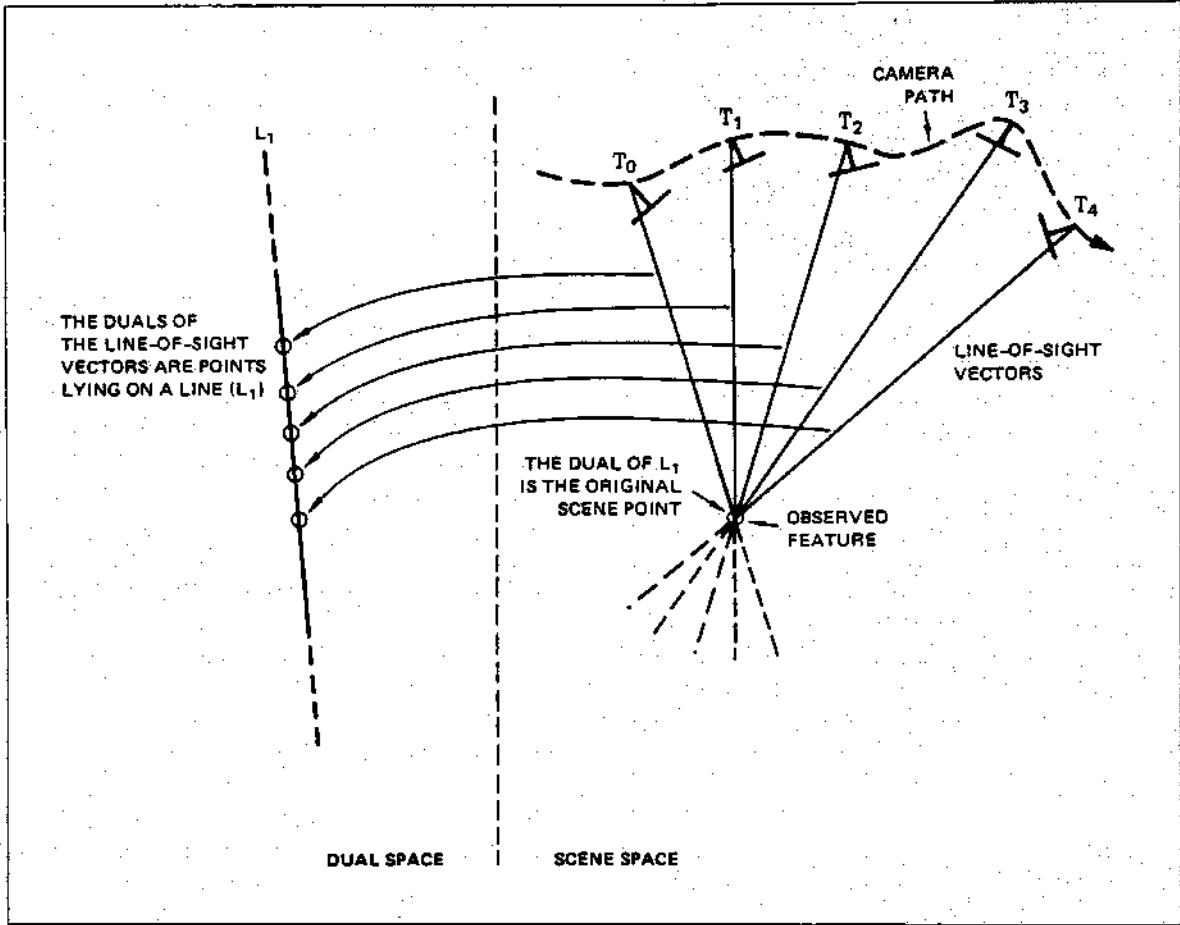


Fig. 5. Line-of-sight duality.

son [11] nearly forty years ago). Because such a sensor is not yet available, we choose an alternate approach: to *transform the features* we detect in image space to the desired epipolar space, the cylindrical coordinate system of figure 1. Here the singularity at the FOE presents no problem, and the observation variances are uniform. The structure we have developed for implementing this transformation brings us several other advantages, as the next section describes.

3 The Spatiotemporal Surface

3.1 Structuring the Data— Spatiotemporal Connectivity

We collect the data as a sequence of images, in fact stacking them up as they are acquired into a

spatiotemporal volume, as shown in figure 6. As each new image is obtained, we construct its *spatial* and *temporal* edge contours. These contours are three-dimensional zeros of the Laplacian of a chosen three-dimensional Gaussian (Buxton and Buxton [12] and Heeger [7] also use spatiotemporal convolution over an image sequence), and the construction produces a spatiotemporal *surface* enveloping the signed *volumes* (note that, in two dimensions, edge contours envelop signed *regions*). The *spatial* connectivity in this structure lets us explicitly maintain object coherence between features observed on separate epipolar planes; the *temporal* connectivity gives us, as before, the tracking of features over time. See the companion paper in this issue [13] for a description of how these surfaces are constructed.

The need for maintaining this spatial connec-

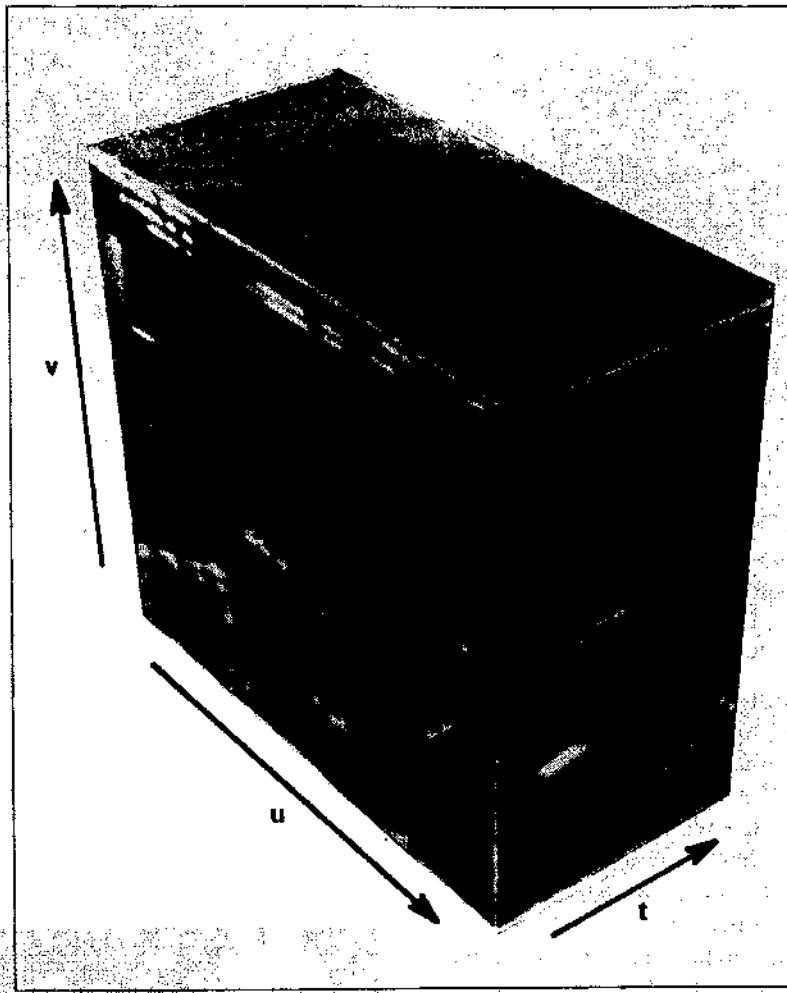


Fig. 6. Spatiotemporal volume.

tivity can be observed by viewing our earlier results [1], one set of which is shown in figure 7. There, in processing the EPIs independently, we obtained separate planes of isolated scene feature estimates. Wishing to exploit the fact that there should be some spatial coherence between these sets of points, we used proximity of the resulting estimates on adjacent planes to filter outliers. Features not within the error (covariance) ellipses of those above or below them (i.e., those which could not be joined into a 3-space contour) were discarded. The remaining point field (figure 7) was sparse and fragmented, and not really representative of the continuous solid surfaces visible in the scene. The problem, however, did not lie with this post hoc filtering but with the loss of spa-

tial connectivity in the first place. Our separation of the data into EPIs, and then subsequent independent processing of these, lost the spatial connectivity apparent in the original images. We maintained instead the temporal connectivity that was critical to the feature tracking. For spatial connectivity in the scene reconstruction, spatial connectivity in the imagery must be preserved. The next two figures present a simplified example of this spatial and temporal connectivity. Figure 8 shows a sequence of simulated images depicting a camera zooming in on a set of rectangles; figure 9 shows a rendered view of the spatiotemporal surfaces arising from this motion. The spatial and temporal interpretation of these surfaces should be quite apparent.



Fig. 7. Orthogonally viewed scene: Results (displayed for crossed-eye viewing).

In our spatiotemporal-surface representation, feature observations bear (u, v, t) coordinates, and are spatiotemporal voxel facets. Figure 12 shows a mesh description of the facets for the spatiotemporal surfaces associated with the forward-viewing sequence whose first and last images are depicted in figure 10. These images are much more complex than those of figure 8. Let us reemphasize that the surface is defined at the zeros of a Laplacian of a 3D Gaussian applied over the sequence: There is no thresholding, and the features are simply zero crossings. In the interest of clarity, the surface representations we

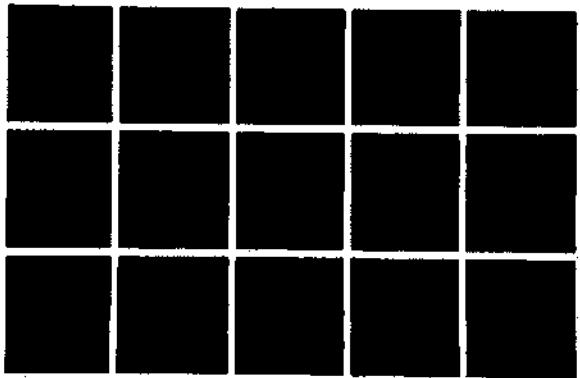


Fig. 8. Simulation: Linear path, motion toward center.

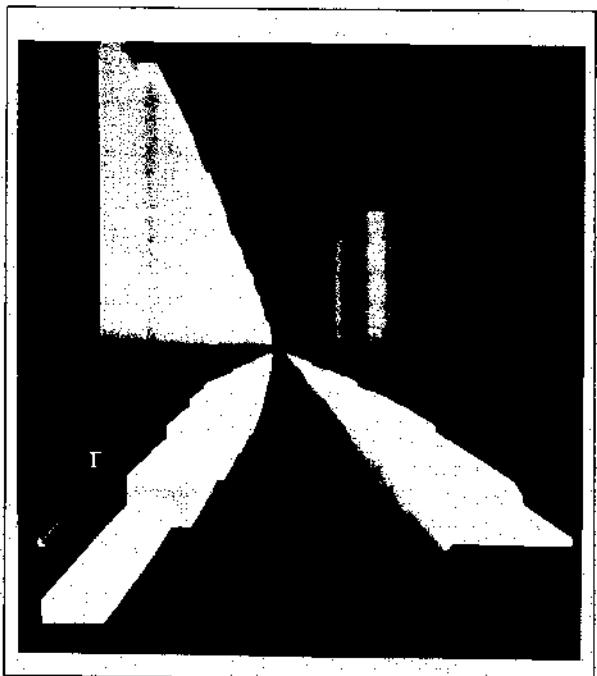


Fig. 9. Surfaces of figure 8 rendered for display.



Fig. 10. Sequence 1st and 128th images.

will show in the remaining figures are based on a simplified version of this imagery—one-eighth the linear resolution of the originals. Figure 11 shows these two frames at the reduced resolution.

Others have addressed this problem of combining spatial and temporal information, although no one has either built surfaces such as these or attempted to maintain explicit track of the temporal change. Perhaps the closest is Waxman [14], who discusses the use of *evolving contours*—isolated 2D contours whose projections over time can be used in deriving the shape of a restricted class of analytic surfaces. He provides no method for tracking the contours



Fig. 11. 1st and 128th images at 1/8 resolution.

through time, however, or for extracting them from real images—nor does he develop a methodology for utilizing the temporal evolution of individual components of the contours over multiple frames. Later work by Waxman and colleagues [15], presenting *convected activation profiles*, involves spatiotemporal convolution of Gaussian gradients applied at features detected in the individual spatial images by a DOG operator. In this, estimates of image-plane velocities are formed from quotients of the spatiotemporal gradients. There is, however, no estimate of scene motion, and no notion of motion associated with specific objects in the field of view—motion

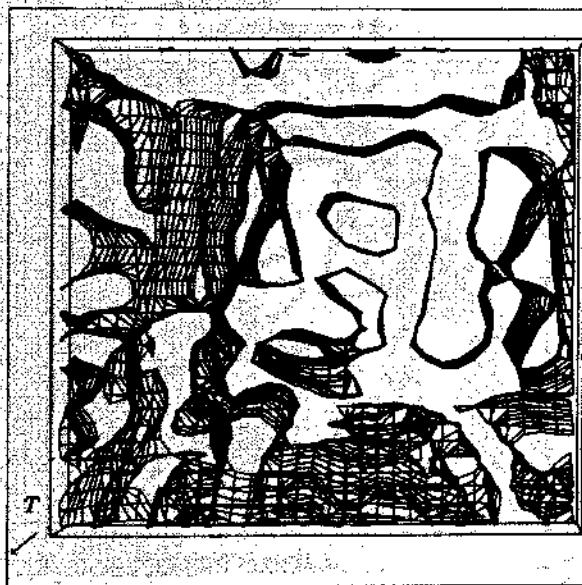


Fig. 12. Spatiotemporal-surface representation, first 10 frames.



is ascribed to pixels in the plane. Others, for example Hildreth and Grzywacz [16], who work with velocity point sets, and Negahdaripour and Horn [17], who determine relative motion of a plane from image gradients, also do not address these issues of local shape, establishing correspondence over time, associating movement with objects, or extracting the measures from real images. Although we have directed our efforts only at ego motion, our space-time surfaces provide a complete representation of these other projective velocity measures, and maintain a continuous track relating them to their underlying scene features. Our work in the future will include looking into using the surface representation for this more general form of motion analysis.

3.2 Structuring the Data— Epipolar-Plane Representation

As mentioned in the previous section, for non-orthogonal viewing directions, epipolar lines are not distinguished by the spatial v scan-line coordinate. To obtain this necessary structuring we develop within the spatiotemporal-surface representation an *embedded* representation that makes the epipolar organization explicit. Over each of the sequential images, we transform the

(u, v, t) coordinates of our spatiotemporal zeros to (r, h, θ) cylindrical coordinates (θ indicates the epipolar-plane angle ($\theta \in [0, 2\pi]$)); the quantized resolution in θ is a supplied parameter; and the transform for each image is determined by the particular camera parameters). In this new coordinate system, we build a structure similar to our earlier EPI edge contours, but dynamically organized by epipolar plane. This is done by intersecting the spatiotemporal surfaces with the pencil of appropriate epipolar planes (as figure 1). We weave the epipolar connectivity through the spatiotemporal volume, following the known camera viewing direction changes. Figure 13 shows a sampling of the spatiotemporal surfaces as they intersect the pencil of epipolar planes (every fifth plane is depicted). You will notice the obvious radial flow pattern away from the epipole (FOE). Figure 14 shows seven of these surface/plane intersections, along with the associated bounding planes (refer to figure 1). The edge that all share is the camera path (the epipole). These seven planes show exactly the contours one would detect in spatiotemporal intensity images such as depicted in figure 3.

In figure 15 we isolate a single surface from the top left of figure 12, and shows its spatiotemporal structure. Figure 16 shows the same surface structured by its epipolar-plane components. The

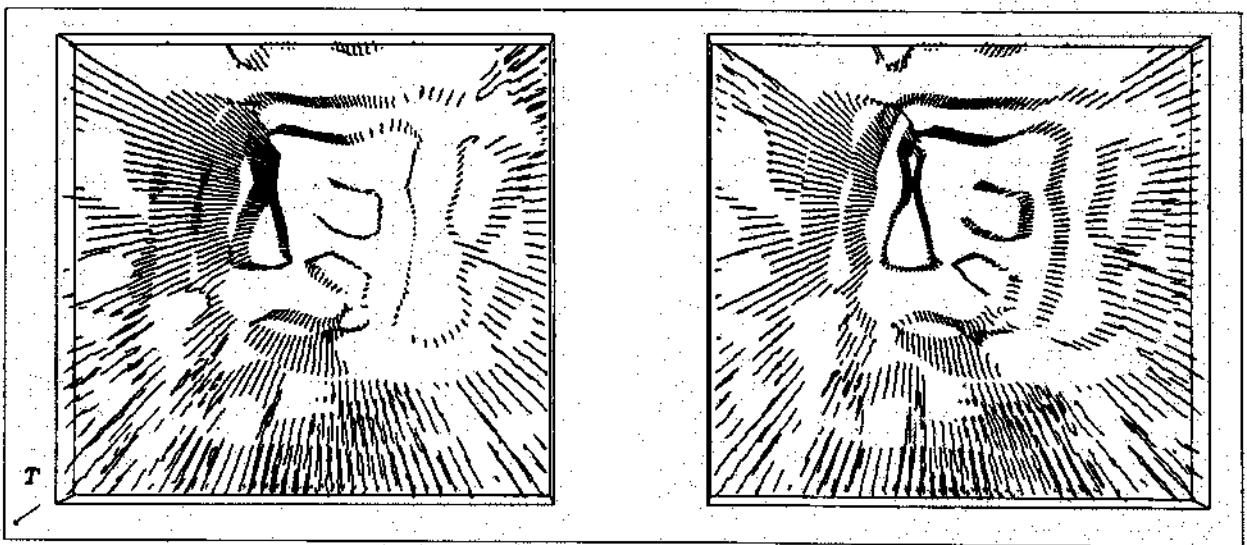


Fig. 13. Epipolar-plane surface representation.

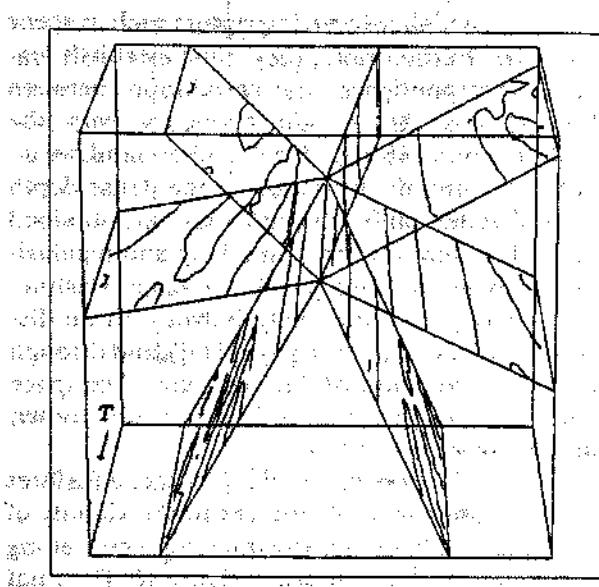


Fig. 14. Intersection: 7 epipolar planes, spatiotemporal surfaces.

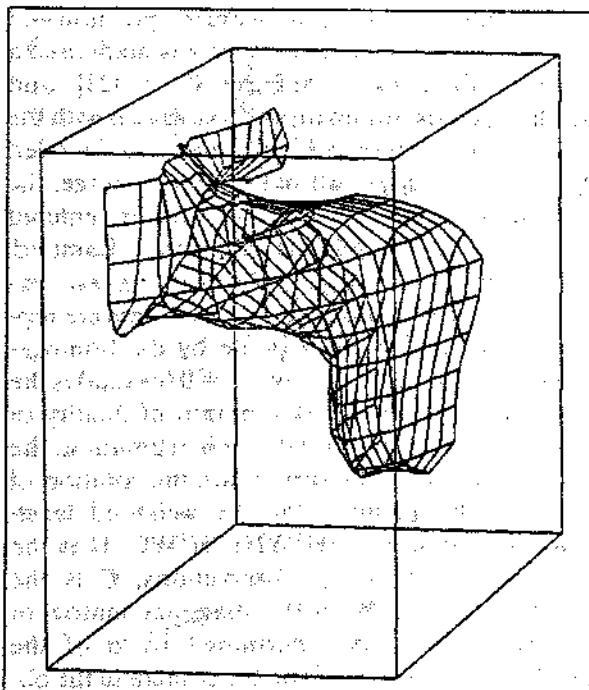
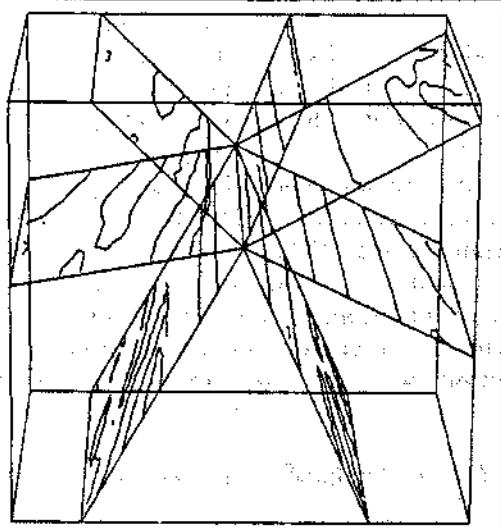


Fig. 15. Spatiotemporal surface.

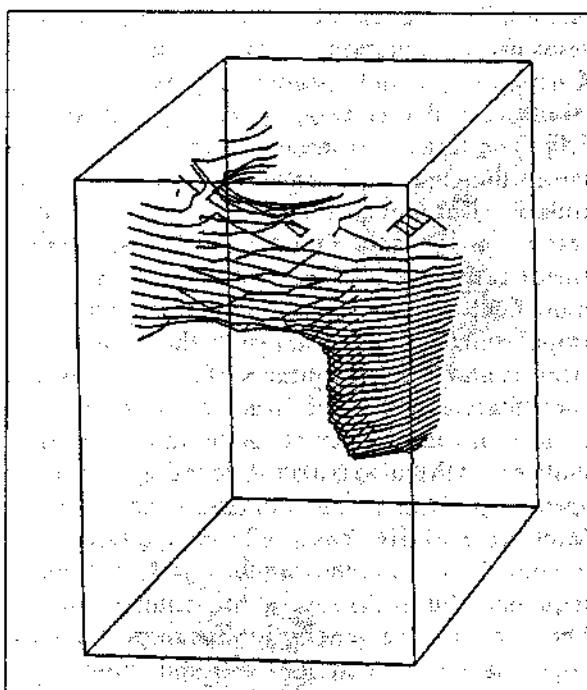


Fig. 16. Epipolar planes.

companion paper [13] gives details of the intersection operation on the spatiotemporal surface. Recall that the displays are in space-time image coordinates. If the camera had been allowed to vary its attitude, the planes depicted in figure 14 would appear skewed, perhaps helical, mirroring the migration of epipolar lines as they are projected on the imaging surface. They might vary in a manner similar to that in which figure 4 varies from figure 2, and for similar reasons. To facilitate presentation, we have not demonstrated this more general camera movement; it is, however, covered by our analysis and implementation.

3.3 Feature Tracking and Estimation

Our approach to scene reconstruction involves tracking scene features as they move in space-time, and to use techniques from estimation theory in approximating and maintaining estimates of their position. This is in distinction with, for example, the work of Hildreth and Grzywacz [16], Waxman and Wohn [4], and others who do not utilize this particular mathematics. Researchers who have built tracking systems using estimation theory include Broida and Chellappa [2] and Gennery [3], as mentioned, Matthies et al. [18], Dickmanns [19], and Hallam [20]. The latter two describe vehicle navigation controllers that work sequentially (as does ours), utilizing Kalman and other filters for estimating motion parameters. Our tracker is a sequential linear estimator, and is implemented as a Kalman filter without the extrapolation phase. Extrapolation is unnecessary since the camera constraints and the space-time surface tell us where each feature will move from frame to frame—there is no need to extrapolate and verify this. Notice that this also makes it clear that there is no *aperture problem* in our approach. The work of Matthies et al. [18] has similarities to ours in its pursuit of scene depth from the analysis of image sequences, but lacks several important elements. These include the generality with respect to view angle that comes with our use of the line-of-sight formulation, and the explicit use of spatial connectivity—they obtain only scene point estimates (as we had with our earlier approach),

rather than higher-level descriptors such as scene contours. Furthermore, they must establish feature correspondence via correlation between frames, and this is not necessary with the spatiotemporal surface. On the other hand, we do not aim currently at producing the dense depth maps that they do. Their depth maps are obtained through a combination of tracking and regularization. When we attempt full-surface reconstruction, we will do so with analysis over scale (as discussed in the companion paper [13]), and through the use of inference on the computed free space (the determination of scene free space was shown in our earlier paper [1]).

Figure 17 shows the tracking of scene features on the spatiotemporal surfaces in the vicinity of the surface of figure 15. The tracking occurs along paths such as those shown in figure 16. The final pair shows, in crossed-eye stereo form, the result of the tracking after 10 frames. The coding is as follows: initiation of a feature tracking is marked by a circle; the leading observation of a feature (active front) is shown as an X; lines join feature observations; 5 observations (an arbitrary number, 2 may be sufficient) must be acquired before an estimate is made of the feature's position—at that point an initial batch estimate is made, and a Kalman filter (discussed by Gelb [21] and Mikhail [22]) is turned on and associated with the feature—this initiation of a Kalman filter is coded by a square; where two observations merge, the tracking is stopped and the features are entered into the data base—this is coded by a diamond.

As mentioned earlier, observations are expressed as line-of-sight vectors, and these are represented in the epipolar plane by the homogeneous line equation $ax + by - c = 0$ (its dual is the point (a, b, c) —see the description of duality in section 2.3). For the initial batch estimation, the coordinates (X) of the feature are the solution of the normal equations for the weighted least-squares system: $\mathbf{X} = (\mathbf{H}^T \mathbf{W} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{W} \mathbf{C}$. \mathbf{H} is the $m \times 2$ matrix of (a_i, b_i) observations; \mathbf{C} is the vector of c_i ; and \mathbf{W} is the diagonal matrix of observation weights, determined by σ of the Gaussian, the distance from the camera to the observed feature at observation position i , and the focal distance. We estimate \mathbf{X} first without weights, then compute the weighted solution and

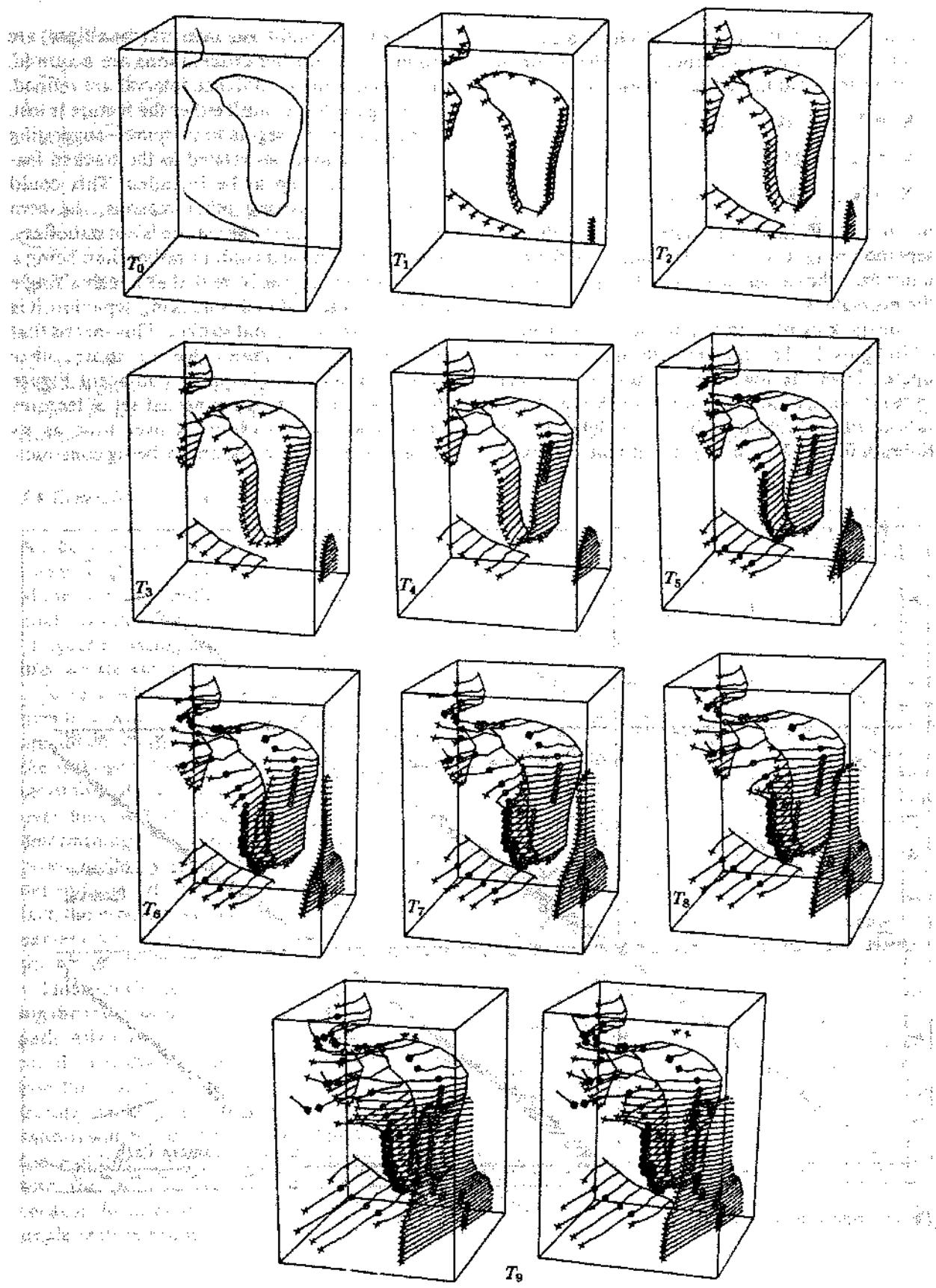


Fig. 17. Sequential feature tracking on the spatiotemporal surface.

the desired covariance matrix, V . Given a current estimate X_{i-1} and covariance V_{i-1} , the Kalman filter at observation i updates these as

$$K_i = V_{i-1} H_i^T / [H_i V_{i-1} H_i^T + w_i]$$

$$V_i = [I - K_i H_i] V_{i-1}$$

$$X_i = X_{i-1} + K_i [c_i - H_i X_{i-1}]$$

K_i is the 2×1 Kalman gain matrix, and w_i is the observation weight, a scalar, dependent on the distance from the camera at observation position i to the estimate X_{i-1} .

The tracking of an individual feature is depicted in figure 18. The camera path runs across the figures from the lower left. Lines of sight are shown from the camera path through the observations of the feature at the upper right. As the Kalman filter is begun (T_4), an estimate (marked

by an \times) and confidence interval (the ellipse) are produced. As further observations are acquired, the estimate and confidence interval are refined. Tracking continues until either the feature is lost, or the error term begins to increase—suggesting that observations not related to the tracked feature are beginning to be included. This could arise because, among other reasons, the zero crossing is erroneous, the feature is not stationary, or the feature is on a contour rather than being a single point in space. Note that although a single feature is presented in this tracking depiction, it is part of a spatiotemporal surface. This means that we have explicit knowledge of those other features to which this is spatially adjacent. Figure 19 shows a contour—a connected set of features on such a surface—observed over time as its shape evolves. Such contours are being construct-

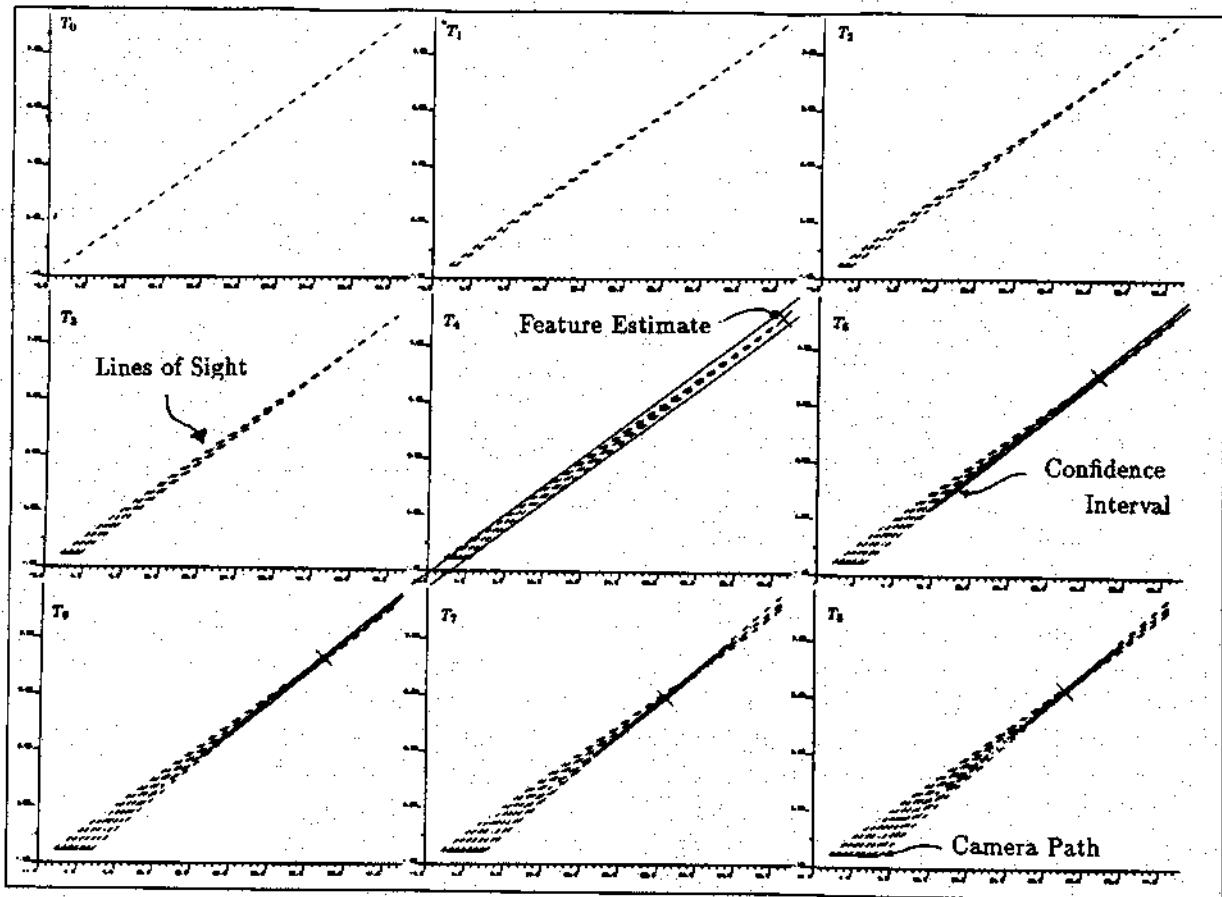


Fig. 18. Sequential estimation.

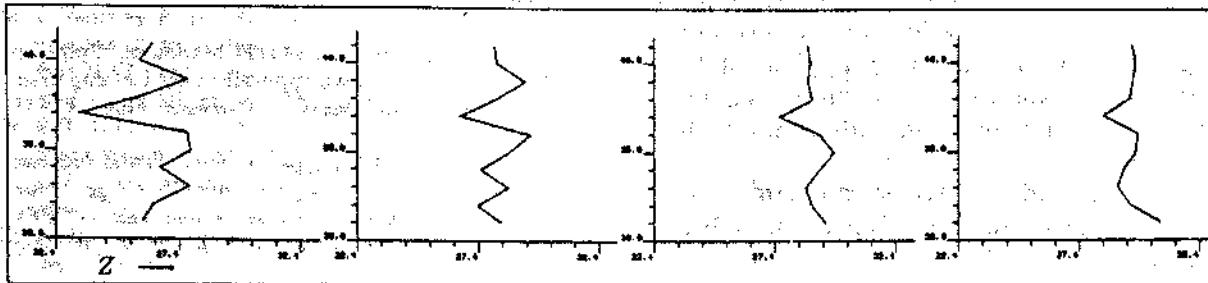


Fig. 19. Contour evolving over time.

ed and refined over the entire image as the analysis progresses. Our current representation of scene structure is based on these evolving contours.

3.4 Generality from the Spatiotemporal Surface

A crucial constraint of the current epipolar-plane image analysis is that having a camera moving along a linear path enables us to divide the analysis into planes, in fact, the pencil of planes of figure 1 passing through the camera path. With this, we are assured that a feature will be viewed in just a single one of these planes, and its motion over time will be confined to that plane. Another crucial constraint is the one we generalized from the orthogonal viewing case—we know that the set of line-of-sight vectors from camera to feature over time will all intersect at that feature, and determining that feature's position is a linear problem. The linearity of the estimator does not depend upon the linearity of the camera path. In fact, the problem would remain linear even if the camera meandered in three dimensions all over the scene.

This knowledge gives us a possibility of removing the restriction that limits us to a linear camera path. All that the linear path guarantees is that the problem is divisible into epipolar planes. If we lose this constraint, then we cannot restrict our feature tracking to separate planes. The observations will, however, still form linear paths in the space of line-of-sight vectors (not to be confused with the (u, v, t) observation space). This is because the lines of sight will all pass through the single feature point. The motion of these obser-

vations will give us *ruled* surfaces in this space—visualize pick-up-sticks jammed in a box, with the sticks being the rules. The rules can be used in the same way they have been with the linear path constraint, to determine the positions of features in the scene. The difference is that the linearities must be located—and the spatiotemporal surface is just the place for doing this. It would also be possible to track using the epipolar constraints that apply pairwise between images—that the constraints are limited to pairwise use arises because, for a nonlinear path, the images will not have the transitivity property we cited earlier.

It is equally worth noting that, when the camera attitude and position parameters are not provided, the spatiotemporal surface contains everything that is necessary for determining them. This is, of course, another problem, but one that must be addressed for a realistic vision system. Our initial work in this involves locating distinctive projective features on the spatiotemporal surface—dihedrals selected using Förstner's measure [23]—and tracking them. Depending upon knowledge of the features chosen, these can enable estimation of both relative and absolute camera parameters [24].

This generality suggests there is even broader application for the technique than we had initially thought—it seems quite adaptable to nonlinear camera paths; and should be usable equally in refining the camera model or solving for its unknown parameters.

4 Conclusions

We showed, in our earlier work, the feasibility of extracting scene depth information through

Epipolar-Plane Image Analysis. Our theory applies for any motion where the lens center moves in a straight line, with the earlier implementation covering the special case of camera sites equidistant and viewing direction orthogonal to the camera path.

The generalizations obtained through spatio-temporal-surface analysis bring us the advantages of

- Incremental analysis
- Unrestricted viewing direction (including direction varying along the path)
- Spatial coherence in our results, providing connected surface information for scene objects rather than point estimates structured by epipolar plane
- The possibility of removing the restrictions that fix us to a known linear path

The current implementation, running on a Symbolics 3600, processes the spatiotemporal surfaces at a 1-KHz voxel rate. The associated intersecting, tracking, and estimation procedures bring this rate down to about 150 Hz, 75 percent of which is consumed in the surface intersection (the surface intersection would not be required if we had a sensor of the appropriate geometry). Both the feature tracking and the surface-construction computations are well suited to MIMD (perhaps SIMD) parallel implementations. With these considerations, and the process's inherent precision and robustness, spatiotemporal-surface-based epipolar-plane image analysis shows great promise for tasks in real-time autonomous navigation and mapping.

Acknowledgements

This research has been supported by DARPA Contracts MDA 903-86-C-0084 and DACA 76-85-C-0004. David Marimont, currently with Xerox PARC, was crucial in the development of this work, providing insights for both the geometry and mathematics of the tracking process and the design of the surface builder. Lynn Quam has provided excellent image manipulation and graphics tools, and, whenever required, thoughtful assistance.

References

1. R.C. Bolles, H.H. Baker, and D.H. Marimont, "Epipolar-plane image analysis: An approach to determining structure from motion," *Intern. J. Computer Vision* 1:7-55, June 1987.
2. T.J. Brodsky and R. Chellappa, "Kinematics and structure of a rigid object from a sequence of noisy images," *Proc. Workshop on Motion: Representation and Analysis*, IEEE Computer Society, Kiawah Island, SC, pp. 95-100, May 1986.
3. D.B. Gennery, "Tracking known three-dimensional objects," *Proc. Nat. Conf. Artif. Intell.*, Pittsburgh, pp. 13-17, August 1982.
4. A.M. Waxman and K. Wohn, "Contour evolution, neighborhood deformation, and global image flow: Planar surfaces in motion," *Intern. J. Robotics Research* 4:95-108, Fall 1985.
5. A.M. Waxman, B. Kamgar-Parsi, and M. Subbarao, "Closed-form solutions to image flow equations for 3D structure and motion," *Intern. J. Computer Vision*, 1:239-258, October 1987.
6. M. Subbarao, "Interpretation of image motion fields: A spatio-temporal approach," *Proc. Workshop on Motion: Representation and Analysis*, IEEE Computer Society, Kiawah Island, SC, pp. 157-165, May 1986.
7. D.J. Heeger, "Depth and flow from motion energy," *Proc. 5th Nat. Conf. Artif. Intell.*, Philadelphia, pp. 657-663, August 1986.
8. D.H. Marimont, "Projective duality and the analysis of image sequences," *Proc. Workshop on Motion: Representation and Analysis*, IEEE Computer Society, Kiawah Island, SC, pp. 7-14, May 1986.
9. H.H. Baker, T.O. Binford, J. Malik, and J.F. Moller, "Progress in stereo mapping," *Proc. DARPA Image Understanding Workshop*, Arlington, VA, pp. 327-335, June 1983.
10. R. Jain, S.L. Bartlett, and N. O'Brien, "Motion stereo using ego-motion complex logarithmic mapping," *IEEE PAMI* 9:356-369, May 1987.
11. J.J. Gibson, *The Perception of the Visual World*. Houghton Mifflin: Boston, 1950.
12. B.F. Buxton and H. Buxton, "Monocular depth perception from optical flow by space time signal processing," *Proc. Roy. Soc. London, Ser. B*, 218:27-47, 1983.
13. H.H. Baker, "Building surfaces of evolution: The weaving wall," *Intern. J. Computer Vision* (this issue).
14. A.M. Waxman, "An image flow paradigm," *Proc. Workshop on Computer Vision: Representation and Control*, IEEE Computer Society, Annapolis, MD, pp. 49-57, April 1984.
15. A.M. Waxman, J. Wu, and F. Bergholm, "Convected activation profiles and the measurement of visual motion," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI, pp. 717-723, June 1988.
16. E.C. Hildreth and N.M. Grzywacz, "The incremental recovery of structure from motion: Position vs. velocity based formulations," *Proc. Workshop on Motion: Representation and Analysis*, IEEE Computer Society, Kiawah Island, SC, pp. 137-143, May 1986.

17. S. Negahdaripour and B.K.P. Horn, "Direct passive navigation," *IEEE Trans. PAMI* 9:168-176, January 1987.
18. L. Matthies, R. Szeliski, and T. Kanade, "Incremental estimation of dense depth maps from image sequences," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI, pp. 366-374, June 1988.
19. E.D. Dickmanns, "An integrated approach to feature based dynamic vision," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI, pp. 820-825, June 1988.
20. J. Hallam, "Resolving observer motion by object tracking," *Proc. 8th Intern. Joint Conf. Artif. Intell.*, Karlsruhe, West Germany, pp. 792-798, August 1983.
21. A. Gelb (ed.), *Applied Optimal Estimation*. Written by the Technical Staff, The Analytic Sciences Corporation, MIT Press, Cambridge, MA, 1974.
22. E.M. Mikhail, with F. Ackerman, *Observations and Least Squares*. University Press of America, Lanham, MD, 1976.
23. W. Förstner, "A feature based correspondence algorithm for image matching," *Proc. Symp. "From Analytical to Digital," Intern. Archives of Photogrammetry and Remote Sensing*, vol. 26-III, Rovaniemi, Finland, August 1986.
24. W. Förstner, "Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision," *Computer Vision, Graphics, and Image Processing* 40:273-310, December 1987.

Analysis of a Sequence of Stereo Scenes Containing Multiple Moving Objects Using Rigidity Constraints*

Zhengyou Zhang Olivier D. Faugeras Nicholas Ayache

INRIA

Domaine de Voluceau
Rocquencourt, BP 105
78153 Le Chesnay FRANCE

Abstract

In this paper, we describe a method for computing the movement of objects as well as that of a mobile robot from a sequence of stereo frames. Stereo frames are obtained at different instants by a stereo rig, when the mobile robot navigates in an unknown environment possibly containing some moving rigid objects. An approach based on rigidity constraints is presented for registering two stereo frames. We demonstrate how the uncertainty of measurements can be integrated with the formalism of the rigidity constraints. A new technique is described to match very noisy segments. The influence of egomotion on observed movements of objects is discussed in detail. Egomotion is first determined and then eliminated before determination of the motion of objects. The proposed algorithm is completely automatic. Experimental results are provided. Some remarks conclude this paper.

Keywords: Motion from Stereo, Egomotion, Multiple Object Motions, Mobile Robot, 3D Matching, Rigidity Constraints, Uncertainty of Measurements.

1 Introduction

Instead of the interpretation and analysis of general 3D motion from two-dimensional images, our research focuses on motion from stereo. There is a broad range of applications of motion from stereo such as mobile robot navigation, target tracking and dynamic surveillance. In this article, we restrict the domain to that of mobile robot navigation in an unknown indoor environment where possibly other objects, such as other mobile robots, may also be moving.

Given a sequence of stereo frames, two operating modes are available. The first, called *Bootstrapping* mode, refers

to the case where nothing about the kinematics of 3D tokens is known. In order to determine ego- and object motion, we have to bring into correspondence some 3D tokens in successive frames using available constraints. When we start a session for analyzing the scene, we begin with the bootstrapping mode, because the only *a priori* information available is a partial estimation of the movement of robot from odometry. After the first few frames, however, the problem becomes simpler. We can use some *a priori* information about the kinematics of tokens extracted from previous frames to help in the estimation of the motion of 3D tokens. This is the second mode, called *Steady-state* mode. Some investigations have already been conducted in this direction [BC86a, BC86b, WHA87, Dic87].

Notice that even in the steady-state mode, there is always a little bit of the bootstrapping mode in the scene where new 3D tokens appear in the field of view about which no *a priori* information is available.

Due to the lack of *a priori* information, the bootstrapping problem is more difficult than the steady-state one. But to deal with the steady-state problem, we must take several (five, for instance) stereo frames a second, and the hardware requirements are more stringent. This article deals only with the bootstrapping problem.

Figures 1 and 2 show two 3D frames at two different instants which are reconstructed by our passive trinocular stereo system [AL87a, AL87b]. Two remarks should be made:

- Line segments are oriented due to the intensity contrast.
- Segments are noisy. Uncertainty is partially manipulated. Each segment is characterized by its orientation vector D and its midpoint M , and also by the covariance matrices W_D and W_M , corresponding to the uncertainty in its orientation and its midpoint, respectively.

*This work was supported in part by ESPRIT project P940.

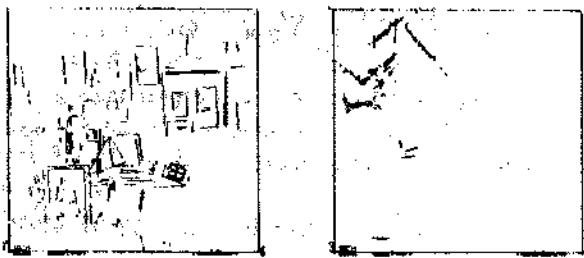


Figure 1: Stereo frame 1



Figure 2: Stereo frame 2

In [FAZ88], we have proposed an algorithm based on the hypothesize-verify paradigm to match two stereo frames. In this paper, we develop the generation of hypotheses phase and give some new results. Section 2 describes the matching process. The use of the rigidity constraint is explained in detail and a new technique is presented to match very noisy line segments. Section 3 discusses the influence of egomotion on the observed object motions. Section 4 presents some experimental results and Section 5 concludes this paper with some remarks.

2 Matching Process and Motion Estimation

2.1 General Presentation

To determine 3D continuous (small) motion from two-dimensional images, we can compute the apparent velocity field of the objects from the intensity variations in the images and relate it to the motions of objects, which in general does not require *a priori* information about feature correspondences [Hil83,Horn86]. In other cases, such as the interpretation and analysis of discrete motion from

2D or 3D frames, feature correspondences of objects between frames should be established before the computation of motion [Ull79,Hua86,AW86,CH87]. Even in the steady-state case, feature correspondences are needed to update the motion parameters. Of course, the size of the search space is drastically reduced compared with that in the bootstrapping one.

The rigidity assumption about the environment and objects is used in most matching algorithms. Our matching process is divided into two stages. In the first stage called *Generation of Hypotheses*, the rigidity constraint is heavily used to generate some hypotheses of segment correspondences between two successive frames. An estimate of motion can be computed from each hypothesis using an Iterative Extended Kalman Filter. In the second stage called *Verification*, we propagate this estimate in the whole frame to try to match more segments. Finally, the best hypothesis is retained.

In [CH87], a rigidity constraint is used in a different way, but the matching algorithm is also divided into two stages. An angular constraint on pairwise line segments is used in the first stage. Pairings whose orientations can be registered by using a tree search procedure are regarded as potential matches. A rotation is then available for each potential match. In the second stage, a Hough-like procedure is used to find translations that would bring segments into correspondence. As a result, line segments which undergo the same motion in space are grouped together.

2.2 Rigidity Constraint

The rigidity constraint implies the conservation of local structure of objects, such as angle and distance between two line segments, during their motions. However, the 3D segments which we have are very noisy (see Figures 1 and 2) and we cannot recover local structure exactly, especially the distance between two lines. Indeed, a significant change in the distance between two segments may result from a slight disturbance on the orientation of a segment. So, instead of the distance between two segments, we use in our algorithm the distance between the midpoints of two segments.

More precisely, if two pairings of segments can form one potential match, they must satisfy the following four conditions. Let AB, CD be two segments in scene 1, $A'B', C'D'$ in scene 2 and M_1, M_2, M'_1, M'_2 be midpoints of $AB, CD, A'B', C'D'$, respectively, the four conditions are then:

$$\begin{aligned} |AB| &\approx |A'B'| \text{ and } |CD| \approx |C'D'| \\ \widehat{AB} \cdot \widehat{CD} &\approx \widehat{A'B'} \cdot \widehat{C'D'} \\ |M_1 M_2| &\approx |M'_1 M'_2| \\ AB \cdot \widehat{M_1 M_2} &\approx A'B' \cdot \widehat{M'_1 M'_2} \end{aligned}$$

In the following sections, we formalize the length and an-

gle constraints explicitly taking into account the uncertainty of measurements. We approximately model the squared length of a segment and the cosine of the angle between two segments as gaussian random variables.

2.2.1 Length Constraint

Given a segment AB in the first scene with the covariance matrix W_D of its orientation, we can compute the variance of its squared length.

Let δ be the uncertainty in the orientation, with

$$E[\delta] = 0 \text{ and } E[\delta\delta^t] = W_D$$

then, the squared length t^2 of segment AB can be represented as following:

$$\begin{aligned} t^2 &= |AB + \delta|^2 [AB + \delta] \\ &= AB^t AB + 2AB^t \delta + \delta^t \delta \end{aligned}$$

If we neglect the second order terms, we have the variance of t^2

$$\begin{aligned} \text{Var}[t^2] &= E[(t^2 - E[t^2])^2] \\ &= 4AB^t W_D AB \end{aligned}$$

So, we can impose that two segments which can be matched (AB in scene 1, $A'B'$ in scene 2) must satisfy:

$$||AB||^2 - ||A'B'||^2 < 2\kappa\sqrt{AB^t W_D AB} \quad (1)$$

where κ is a coefficient. Looking at the table of the erf function, we can choose an appropriate κ . For example, we can take $\kappa = 1.5$ for a probability of 87% when we consider the lengths of segments, and $\kappa = 1$ for a probability of 68% when we consider the distance between the midpoints of the two segments. That is, we impose a stricter constraint on the distance between midpoints than on the lengths of segments.

2.2.2 Angle Constraint

Given two segments AB and CD in the first scene, let δ_1 and δ_2 be the uncertainties in their orientations, $W_{\delta_1} = E[\delta_1\delta_1^t]$ and $W_{\delta_2} = E[\delta_2\delta_2^t]$ are known, and suppose that $E[\delta_1] \cdot E[\delta_2] = 0$.

The cosine of the angle between these two segments is:

$$\cos \theta = \frac{(AB + \delta_1) \cdot (CD + \delta_2)}{|AB + \delta_1| \cdot |CD + \delta_2|}$$

Notice that

$$\frac{1}{\sqrt{x + \epsilon}} \approx \frac{1}{\sqrt{x}} - \frac{\epsilon}{2x^{3/2}} \quad \text{if } \epsilon \ll x,$$

and that

$$|AB + \delta| = \sqrt{|AB|^2 + 2AB \cdot \delta + |\delta|^2}$$

we have

$$\cos \theta = \frac{AB \cdot CD + AB \cdot \delta_2 + \delta_1 \cdot CD}{|AB| \cdot |CD|}$$

Here, we have neglected terms of orders higher than 2.

Suppose that AB and CD are independent, we can now easily compute the variance of $\cos \theta$:

$$\text{Var}[\cos \theta] = \frac{1}{|AB|^2 |CD|^2} (AB^t W_{\delta_2} AB + CD^t W_{\delta_1} CD)$$

Let $A'B'$, $C'D'$ be 2 segments in scene 2 to be matched with AB and CD , they should satisfy the following constraint:

$$|\cos(\widehat{AB \cdot CD}) - \cos(\widehat{A'B' \cdot C'D'})| < \frac{\kappa}{|AB| |CD|} \sqrt{AB^t W_{\delta_2} AB + CD^t W_{\delta_1} CD} \quad (2)$$

where κ is a coefficient. We can choose $\kappa = 0.8$ for a probability of 57% or $\kappa = 1$ which corresponds to a probability of 68% for the four segments to match. Here, we choose κ smaller than for the length constraint (see Formula 1), due to the fact that the angle is more robust to segmentation errors than the length.

2.3 Deriving Additional Constraint and Solving Local Similarity

To make full use of 3-D information obtained by our passive stereo system and to reduce the number of incorrect hypotheses, we can derive an additional constraint on the orientation of segments.

2.3.1 Orientation Congruency

Proposition : Suppose an oriented segment undergoes a rigid displacement between two successive frames with a rotation angle between 0 and π , then the change of orientation of the segment is less than or equal to the rotation angle between the two frames.

Proof : Let $l = (p, q, r)^t$ be the unit orientation vector of the segment in the first frame, and l' be that in the second frame, then we have $l' = Rl$ where R is the rotation matrix.

If we choose the exponential representation of rotation matrix (cf. [FAF86,Fau88]), that is

$$R = e^H$$

where

$$H = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$$

The vector $r = [a \ b \ c]^t$ gives the direction of axis of rotation and its norm is the rotation angle around this axis. The

relationship between R and r is the following:

$$R = \begin{bmatrix} \cos\theta + a^2g(\theta) & abg(\theta) - cf(\theta) & acg(\theta) + bf(\theta) \\ abg(\theta) + cf(\theta) & \cos\theta + b^2g(\theta) & bcg(\theta) - af(\theta) \\ acg(\theta) - bf(\theta) & bcg(\theta) + af(\theta) & \cos\theta + c^2g(\theta) \end{bmatrix}$$

where $\theta = \sqrt{a^2 + b^2 + c^2}$ is the rotation angle, $f(\theta) = \frac{\sin\theta}{\theta}$, and $g(\theta) = \frac{1 - \cos\theta}{\theta^2}$.

Now it is easy to compute the angle between l and l' :

$$\begin{aligned} \cos(l, l') &= \cos(l, Rl') \\ &\equiv \frac{1 - \cos\theta}{\theta^2} (ap + bq + cr)^2 + \cos\theta \end{aligned}$$

Because $\frac{1 - \cos\theta}{\theta^2} (ap + bq + cr)^2 \geq 0$, $\cos(l, l') \geq \cos\theta$, i.e. $|l'| \geq |\theta|$ for $0 \leq \theta < \pi$.

Note that when $\theta \neq 0$ (not pure translation), $|l'| = \theta$ if and only if $ap + bq + cr = 0$, that is, if a segment undergoes a movement whose rotation axis is perpendicular to the segment, then the angle of the change of segment orientation is equal to the rotation angle. If not, it is less than the rotation angle. Q.E.D.

In general, the rotation angle between two successive frames does not go beyond 60 degrees, so we can impose that the orientation difference of a pairing of segments to be matched must be less than 60 degrees.

2.3.2 Solving Local Similarity

In an indoor environment, we encounter many things which are similar such as: two windows, two tables, one window and one table, two sides of a cube block. While this phenomenon (so-called *local similarity*) can be partly solved by the above constraint (*orientation congruency*), several cases remain unsolved such as illustrated in Figure 3.

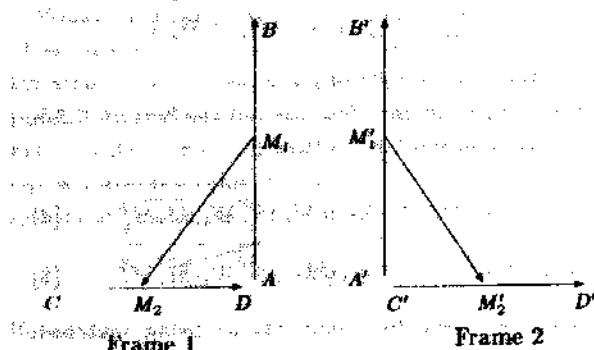


Figure 3: Local similarity

The two pairings in Figure 3 satisfy all the previous constraints, but this is evidently a wrong hypothesis. The problem is that we use "cosine" in the formulation of angle constraint. To solve it, we can use some "sine-like" information. We can, for instance, examine the congruency of

orientation between the normal to AB and M_1M_2 , and that to $A'B'$ and $M'_1M'_2$. As described in Section 2.3.1, we can impose that orientation difference of normals must be less than 60 degrees. In Figure 3, the normal points "in" the paper in the first scene, and that in the second scene points "out" of the paper, so the angle between the two normals is 180 degrees, which is greater than 60 degrees. Thus, we can eliminate this improper match.

2.4 Reducing the Complexity of the Algorithm

Now, we can use the constraints described above to generate hypotheses of matches between two successive frames. If we explore all possibilities of pairs of segments, the complexity is $O(p^2q^2)$ where p is the number of segments in the first frame and q is that in the second one. The algorithm is given in Figure 4.

```

for each two segments  $l_1$  and  $l_2$  of frame 1, do
    for each two segments  $l'_1$  and  $l'_2$  of frame 2, do
        if  $l_1, l_2$  and  $l'_1, l'_2$  satisfy all the constraints
            described above,
            then they are retained as a hypothesis.
    endif;
done;
done.

```

Figure 4: Algorithm of hypothesis generation

In order to obtain more reliable hypotheses, we use triplets of segments. Let l_1, l_2 and l'_1, l'_2 be two compatible pairings (in the sense that they satisfy the above constraints), if another pairing l_3 and l'_3 is also compatible with l_1 and l'_1 , then we verify if this pairing is compatible with l_2 and l'_2 . If so, l_1, l_2, l_3 and l'_1, l'_2, l'_3 can form an hypothesis. This means that each hypothesis has propagated at least one segments, and therefore we have a more reliable hypothesis, on one hand; on the other hand, we increase the complexity of the algorithm.

Notice that we do not want to recover all matches between two frames, but to recover all potential motions between two frames, so we need not to explore the search tree exhaustively. There are a number of methods for reducing the complexity:

Sort the segments Sort all segments in each frame in decreasing length order, so that we can easily find, by binary search, the segments in the second frame which are compatible in length with the segments in the first one.

Control search depth Rather than find all possible matches compatible with a certain estimation of displacement, we can stop if we have found a sufficient number of compatible pairings (5, for instance).

Avoid redundant hypotheses If a pairing is already retained as a potential match in some early hypothesis, it is not necessary to continue our search, because it does not give us new information about the motion between two frames.

Reduce search width Consider segments of the first frame only in the central part of the frame, because segments on the sides are likely to move out of the view field in the next frame.

Reduce the number of segments Choose only the longest segments in the first scene, for instance, the p/n longest ($n = 1, 2$ or 3).

Other constraints can be easily integrated in the algorithm to speed the generation process. For example, the assumption that objects and robot only move horizontally (in the ground plane) can be used.

2.5 Motion Estimation

For each hypothesis, we can compute a preliminary estimate of motion using an Extended Kalman Filter [FAF86, AF87, FAZ88, Jaz70]. We apply this estimate to the first frame and compare the transformed frame with the second one. If a transformed segment from the first frame is near enough to some segment in the second frame (cf. Section 2.6), then this pairing is considered as matched, and again, the Extended Kalman Filter is used to update the motion estimate. In the end, the optimal motion estimate of the best hypothesis is retained as the motion between these two frames. Best is quantified as the hypothesis which can bring the largest number of segments into correspondence and yield a minimal error for these matches [FAZ88].

2.6 New Technique to Match Noisy Segments

In our earlier version of the algorithm, we measured the generalized Mahalanobis distance between endpoints of segments. A number of segments can not be matched using this technique, because a segment of an object can be differently segmented in successive images. In this section, we present a new technique which can match very noisy segments.

Let l_1 be a segment in the first frame transformed in the second frame, and l_2 a segment in the second frame.

Step 1: Examine the similarity of orientation. Using Formula 2 described in Section 2.2.2, if the following condition:

$$1 - \cos \theta < \frac{\kappa_u}{|l_1| |l_2|} \sqrt{l_1^T W_{l_2} l_1 + l_2^T W_{l_1} l_2} \quad (3)$$

is satisfied, we proceed to the next step. The above condition is reasonable (cf. Formula 2), because $E[\theta] = 0$, i.e. $E[\cos \theta] = 1$.

Step 2: Examine the distance between two segments. Two segments which can be matched must be in one of the following configurations (Figure 5):

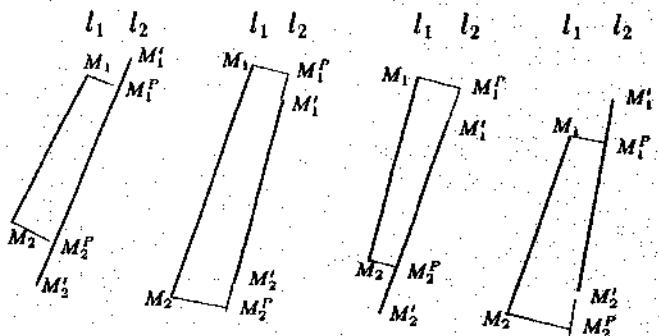


Figure 5: Configurations of matched segments

In order to know whether the configuration of two segments to be matched is among the above four configurations, we can project the two endpoints M_1 and M_2 of segment l_1 on segment l_2 (M_1^P, M_2^P are these projections, respectively), and examine the following condition:

$$(M_{1z}^P \geq M_{2z}^P \text{ and } M_{2z}^P \leq M_{1z}^P) \\ \text{or } (M_{1z}^P \leq M_{2z}^P \text{ and } M_{2z}^P \geq M_{1z}^P)$$

where M_z is the z coordinate of point M .

Using again the formula 1 described in the Section 2.2.1, if the following conditions are satisfied:

$$|M_1 M_1^P|^2 < 2\kappa_l \sqrt{M_1 M_1^P W_{l_1} M_1 M_1^P} \quad (4)$$

$$\text{and } |M_2 M_2^P|^2 < 2\kappa_l \sqrt{M_2 M_2^P W_{l_2} M_2 M_2^P} \quad (5)$$

we then consider these two segments as being matched. Note that the above condition is reasonable, since the expected distance between M_i and M_i^P is zero. In the Formulae 4 and 5, $W_{\delta i} = W_{M_i} + W_{M_i^P}$. The covariance matrix $W_{M_i^P}$ of point M_i^P can be approximated as follows:

$$W_{M_i^P} = \frac{1}{(M_{2z}^P - M_{1z}^P)^2} [(M_{2z}^P - M_{1z}^P)^2 W_{M_i^P} + (M_{1z}^P - M_{2z}^P)^2 W_{M_i^P}]$$

Using this new technique and choosing appropriate κ_u and κ_l , we can recover almost all possible matches between

successive frames, even if the segments are very noisy. Note that the condition 3 is not needed theoretically since we have the conditions 4 and 5. In practice, we can use the condition 3 to reject a large portion of the candidates before we enter the second step, whose computation is more expensive. Furthermore, we can replace the right hand side of Equation 3 by a constant. If Θ (for instance, 30 degrees) is the given tolerance in orientation, we can replace Equation 3 by the following:

$$\cos \theta > \cos \Theta.$$

3 Influence of Egomotion on Observed Object Motion

If there are some moving objects in the environment, we can find multiple possible motions when we register two successive frames: egomotion and motions corresponding to objects.

Suppose now there is only one moving object (the following results can be easily extended to the case of multiple objects). Using the matching process described above, we can recover two different motions: R_E, t_E , the inverse of robot motion and R'_O, t'_O , the object motion as observed by the robot (in the coordinate system of the second frame). But notice that R'_O, t'_O is not the real motion of the object. Indeed, it is influenced by the egomotion and we have following relations:

$$\begin{cases} R_O = R'_O R_E \\ t_O = t'_O - R_O t_E \end{cases} \quad (6)$$

where R_O, t_O is the real object motion.

Equation 6 can be easily verified (cf. Figure 6). If the object is static, it should be observed by the robot at O'_{b2} in the second frame. $O'_{b2} = R_E O_{b1} + t_E$, where O_{b1} is the object position in the first frame. In reality, the object is observed in O_{b2} in the second frame, so the difference between O'_{b2} and O_{b2} is the real motion of the object (still in the coordinate system of second frame), that is:

$$O_{b2} = R_O O'_{b2} + t_O$$

But, $O_{b2} = R'_O O_{b1} + t'_O$
and
 $O'_{b2} = R_E O_{b1} + t_E$

After a simple calculation, we obtain Equation 6.

So we have at least two approaches to correctly recover the object motions. (1) Recover all possible motions between two frames, then determine the egomotion and use the above equation to compute the real motions of objects.

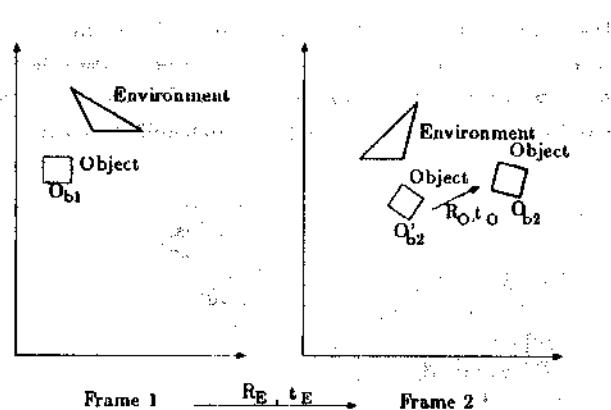


Figure 6: Influence of egomotion on observed object motion

(2) If the segments corresponding to the environment are much more numerous than those of objects, we need not explore all possible matches. We can choose only half of the segments in the first frame and take as the egomotion the motion which can bring into correspondence the largest number of segments. It is reasonable to choose only half of the segments, because we only need to match 3 segments belonging to the environment to compute the egomotion. Once the egomotion is recovered, we can apply it to the first frame. Applying again the above matching process on the transformed frame and on the second one, we now recover the real motions of objects, since we work in the same coordinate system (that of the second frame).

There remains a problem, however. How to select the egomotion among all possible motions? As stated above, if we know that the environment has more segments than objects, we can then select the motion estimate that matches the largest number of segments. If we know an *a priori* estimate of the robot motion (for example, an estimate given by the odometric system of robot), we can select the one which is nearest to the *a priori* estimate and yields the smallest matching error.

4 Experimental Results

In this section, two experimental examples are provided to demonstrate the matching process and the multiple object motion determination. In each figure given, the left hand is the front view and the right hand is the top view.

4.1 Example for Matching Process

Figures 1 and 2 are two stereo frames constructed by our mobile robot in two different positions. The triangle in each frame represents the optical centers of the cameras of our trinocular stereo system. We have 261 segments in the first

frame and 250 segments in the second. Note that there is a large displacement between these two positions (about 10 degrees of rotation and 75 centimeters of translation) which can be noticed by superposing the two frames (see Figure 7).

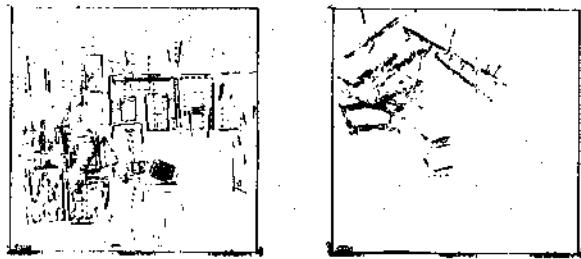


Figure 7: Superposition of the two original frames : Segments of frame 1 are in dashed lines and those of frame 2 are in solid lines

Applying our hypothesis generation procedure on these two frames, we obtain 11 hypotheses. All these hypotheses are propagated in the whole frame to match more segments and update the motion estimate. In the end, 7 hypotheses correctly give the estimate of displacement (with a slight difference). The one which matches the largest number of segments and gives the minimal matching error is kept as the best one. To determine how good this estimate is, we apply it to the first frame and superpose the transformed one on the second, i.e. in the coordinate system of the second frame (see Figure 8). One can observe the very good accuracy of this estimate.

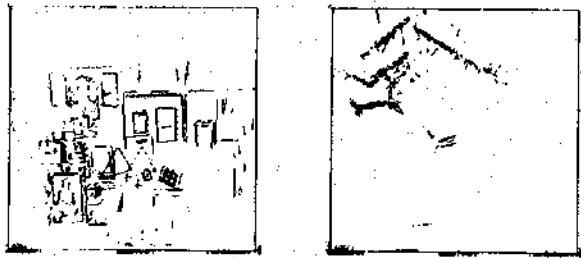


Figure 8: Superposition of the transformed frame of the first one (in dashed lines) on the second (in solid lines) in the coordinate system of frame 2

In the propagation phase, we also recover matched segments. If we use the criterion of the generalized Mahalanobis distance between endpoints of segments as in our

early version of algorithm, we recover only 76 matches (see Figures 9 and 10). If we use the new technique described in Section 2.6, we recover 156 matched segments (60% of the total segments) (see Figures 11 and 12). A remarkable improvement can be observed.

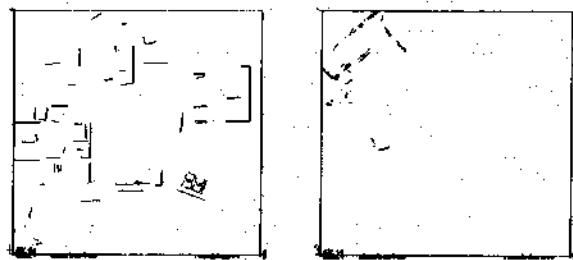


Figure 9: Matched segments of frame 1 based on distance of endpoints

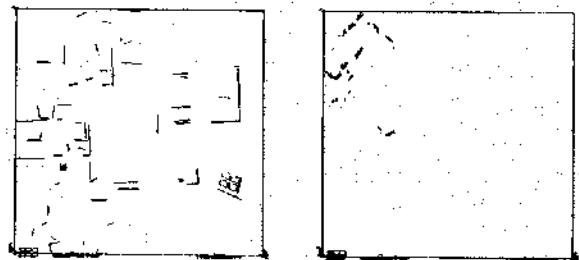


Figure 10: Matched segments of frame 2 based on distance of endpoints

In order to improve the accuracy of the segment measurements, we can fuse each pairing of matched segments. Figure 13 shows the superposition of matched segments of two frames (in the coordinate system of frame 2) and Figure 14 is the result after fusion. The improvement in accuracy is obvious.

The error of the estimate with respect to that given by the odometer of the robot is 0.66 degrees in rotation and 0.3 centimeters in translation. This error is due to the cumulation of errors in all different phases of our navigation loop, especially the error of the mechanical system of the mobile robot.

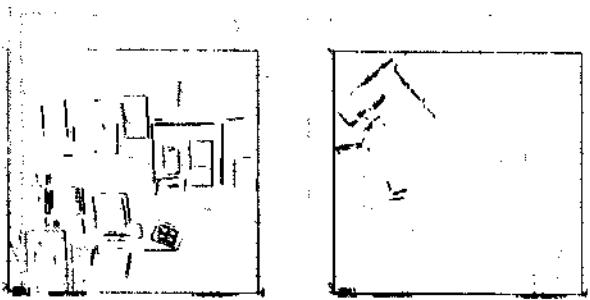


Figure 11: Matched segments of frame 1 using the new technique described in the paper.

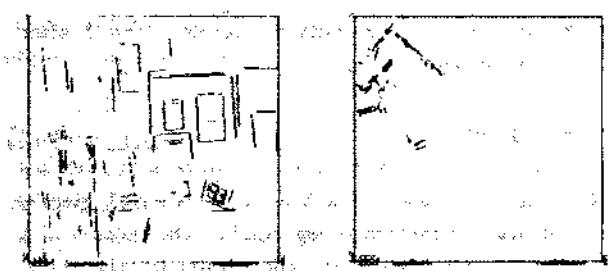


Figure 12: Matched segments of frame 2 using the new technique described in the paper

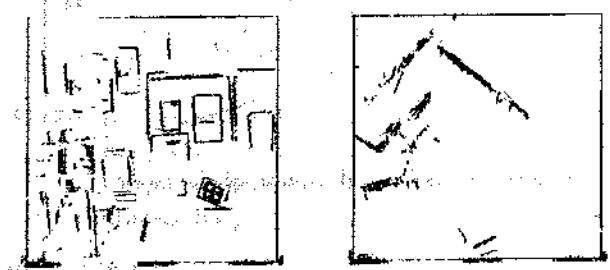


Figure 13: Superposition of the matched segments after computation of motion

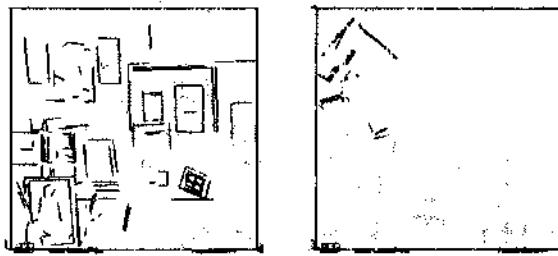


Figure 14: Fusion of all matched segments

4.2 Example of the Determination of Multiple Objects Motions

We adopt the second approach described in Section 3 to determine the multiple objects motions. We demonstrate the procedure on an experimental example; Figures 15 and 16 are the horizontal and vertical projections of the reconstructed segments of an indoor scene observed by the mobile vehicle in two different positions. The two positions differed only by a rotation of the cameras by an angle of 5 degrees with respect to an almost vertical axis. We have added to the scene two synthetic objects, a small house and a chair, which have two different motions and whose segments are corrupted by noise. The simulated motion of the house is a rotation of 30 degrees with respect to a vertical axis and a translation of about 80 cms while the chair has been rotated by -40 degrees with respect to a vertical axis and translated also by about 80 cms. In a near future, we shall give examples with real moving objects.



Figure 15: First 3D snapshot

In order to speed up the process of hypotheses generation, we choose only the $p/2$ longest segments in the first scene. Eight hypotheses are generated among which, five are correctly corresponding to the static environment, one is correctly corresponding to a moving object, and two are wrong



Figure 16: Second 3D snapshot

hypotheses (after verification, we find that these correspond to very few (0 to 3) segments matched). If we consider that the number of segments of the environment is larger than that of segments on the objects, we can choose the hypothesis that matches the largest number of segments as the best one. Figure 17 shows the result obtained when applying the estimated egomotion to the segments in scene 1 and displaying them in dotted lines overlaying scene 2 where segments are drawn in continuous lines. Clearly, most of the background has been matched.

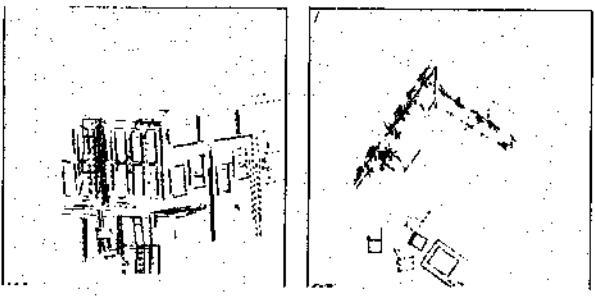


Figure 17: Result of applying the estimated egomotion from Scene 1 to Scene 2 to the segments of Scene 1

After determining the motion of the robot, we have matched a number of segments of the static environment between the two 3D snapshots. But there still exist some segments of the environment and of the moving objects which have not been matched. This is shown in Figure 18.

If we apply the estimate of the displacement of the robot that we have obtained to the first scene, and remove the segments matched (including those segments matched by the above technique) from both scenes, we get two 3D snapshots which contain only moving objects (and also some segments of the environment which have not been matched). The influence of the egomotion on the motions of these objects has been cancelled. This is shown in Figure 18 which shows the first scene after applying the estimated egomotion to its

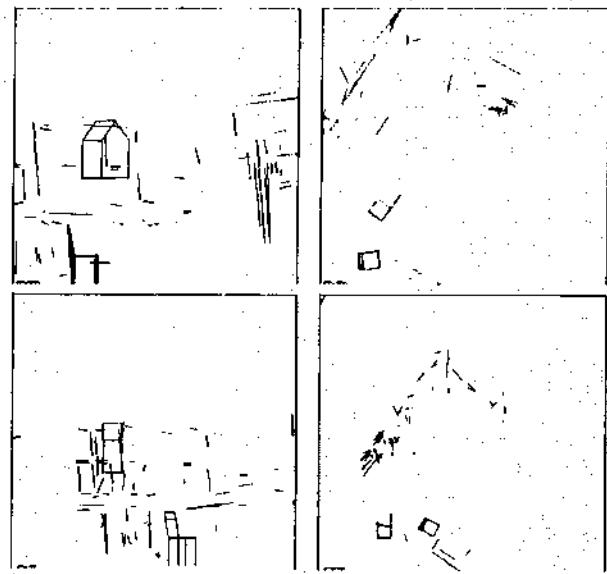


Figure 18: Unmatched segments in scenes 1 and 2 after cancellation of the egomotion

unmatched segments.

At this point, we apply again the procedure of hypotheses generation and verification. Each hypothesis which gives a different estimate of motion is regarded as the motion of a moving object. Figure 19 shows the results for the chair, and Figure 20 shows them for the house. Clearly, the two motions have been well recovered.

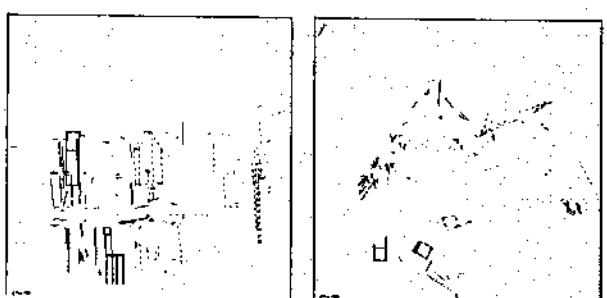


Figure 19: Estimated motion of the chair

5 Conclusion

In this paper, we have presented an algorithm based mainly on the exploitation of the rigidity constraint to determine multiple objects motions as well as the egomotion of the robot. The uncertainty of measurements is partially taken into account in the formalism of the rigidity constraint. The

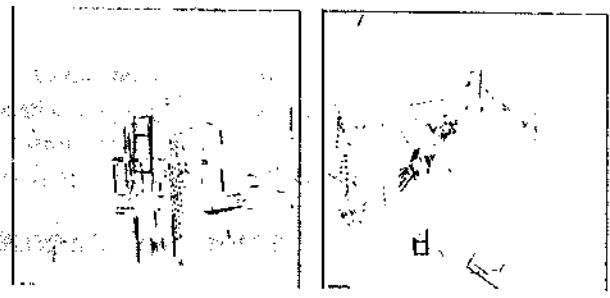


Figure 20: Estimated motion of the house

constraint of segment orientation similarity is derived. A new technique is proposed to match noisy segments and uncertainty is again taken into account. The influence of egomotion on observed object motion is discussed in detail and two approaches are proposed to recover object motions correctly. The proposed algorithm is completely automatic. Two experimental examples are provided and excellent results are observed. The algorithm can be easily implemented in parallel.

References

- [AW86] J.K. Aggarwal and Y.F. Wang, "Analysis of a sequence of images using point and line correspondences", In *Proc. Conference on Robotics and Automation*, pages 1275-1280, IEEE, 1986.
- [AF87] N. Ayache and O.D. Faugeras, "Maintaining representations of the environment of a mobile robot", In *International Symposium on Robotics Research*, August 1987. Santa-Cruz, California.
- [AI87a] N. Ayache and F. Lustman, "Fast and reliable passive binocular stereovision", In *Proc. First International Conference on Computer Vision*, pages 422-427, IEEE, June 1987. London, U.K.
- [AI87b] N. Ayache and F. Lustman, "Binocular stereovision, recent results", In *Proc. International Joint Conference on Artificial Intelligence*, August 1987. Milano, Italy.
- [BC86a] T.J. Broida and R. Chellappa, "Kinematics and structure of a rigid object from a sequence of noisy images", In *Proc. IEEE Workshop on Motion Representation and Analysis*, pages 95-100, Charleston, SC, May 1986.
- [BC86b] T.J. Broida and R. Chellappa, "Kinematics and structure of a rigid object from a sequence of noisy images: A batch approach", In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 176-182, Miami Beach, FL, June 1986.
- [CH87] H.H. Chen and T. Huang, "An algorithm for matching 3-D line segments with application to multiple-object motion estimation", In *Proc. of the IEEE Computer Society, Workshop on Computer Vision*, pages 151-156, November 30-December 2, 1987.
- [Dic87] E.D. Dickmanns, "4D-dynamic scene analysis with integral spatio-temporal models", In *Proc. ISSR'87*, pages 73-80, Santa-Cruz, 1987.
- [FAF86] O.D. Faugeras, N. Ayache, and B. Favereon, "Building visual maps by combining noisy stereo measurements", In *Proc. International Conference on Robotics and Automation*, pages 1433-1438, April 1986. San Francisco, CA, USA.
- [Fau88] O.D. Faugeras, *A Few Steps Toward Artificial 3D Vision*, Technical Report N° 790, INRIA, B.P. 105, 78153 LE CHESNAY CEDEX, 1988.
- [FAZ88] O.D. Faugeras, N. Ayache, and Z. Zhang, "A preliminary investigation of the problem of determining ego- and object motions from Stereo", In *Proc. International Conference on Pattern Recognition*, 1988, 9th, Rome, Italy.
- [Hil83] E.C. Hildreth, *The Measurement of Visual Motion*, MIT Press, Cambridge, 1983.
- [Horn86] B.K.P. Horn, *Robot Vision*, MIT Press and McGraw-Hill Book Company, 1986.
- [Hua86] T.S. Huang, "Motion Analysis", In *AI Encyclopedia*, Wiley, 1986.
- [Jaz70] A.M. Jazwinsky, *Stochastic Processes and Filtering Theory*, Academic Press, 1970.
- [Ull79] S. Ullman, *The Interpretation of Visual Motion*, MIT Press, Cambridge 1979.
- [WHA87] J. Weng, T.S. Huang, and N. Ahuja, "3-D Motion Estimation, Understanding, and Prediction from Noisy Image Sequences", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-9, No. 3, pages 370-389, 1987.

Chapter 7: Knowledge-Based Vision

Computer vision systems recover useful information about a scene using knowledge of both image formation and the application domain. Image-understanding tasks have proven to be exceedingly difficult in general domains. Consequently, knowledge-engineering techniques have become popular in computer vision systems. We will briefly discuss these techniques and their impact on computer vision research in this chapter.

Knowledge representation

The central issue in the higher levels of a vision system is the representation and use of knowledge. The term, "knowledge representation" is nebulous. "Knowledge" is a collection of descriptions, assimilation procedures, and problem-solving methods that serve to organize and summarize observations and to support a useful degree of problem-solving ability.²⁴⁴ A "representation" is a particular kind of symbol; the symbol's structure is perceived to correspond in some way to the structure of the thing for which the symbol stands. (A symbol stands for something else by reason of association or convention.) "Knowledge representation" combines data structures and interpretive procedures that, if used correctly in a program, will lead to knowledgeable behavior of a computer vision system. Researchers interested in knowledge representation have designed several data structure classes for storing information in computer programs, and they have developed procedures that enable intelligent manipulation of these data structures to make inferences. Many of these representations can be used in computer vision systems.

In knowledge-based vision systems, good solutions often depend on good representations. The choice of representation is crucial for the characterization of visual data, since representational primitives effectively limit what a system can perceive, know, or understand. ("Representational primitives" are the primitive elements and operators from which an open-ended range of learned concepts can be constructed.) However the choice of representation is difficult, since many possibilities exist and criteria for comparing different representations are unclear.

We must encode different kinds of knowledge for image understanding. Object representation must include individual instances, object classes, object descriptions, object relationships, and constraints. Also, performance knowledge — the knowledge of how to perform different tasks — must be encoded. This encoding can involve such tasks as finding a particular object's edges, linking edge points, and finding all points that belong to a region. Measuring performance is important in providing a feedback path by which a system can modify its strategy during processing. However, defining the necessary performance criteria is difficult, especially in low-level tasks such as segmentation. We also use metaknowledge, which is knowledge about what we know — that is, knowledge about the extent and reliability of our knowledge, in our problem solving activities.

Barr and Feigenbaum²⁴⁵ and Rich²⁴⁶ provided good descriptions of popular knowledge representation techniques that fall into four categories: formal logic, semantic nets, production systems, and frames.

In a computer vision system, the computer is fed information via sensing devices and then must use knowledge about the object to arrive at a decision. Various knowledge representation techniques used to tackle the decision-making problem are discussed in this section.

Semantic networks. A semantic net represents information as a set of nodes interconnected by labeled arcs that represent relationships among the nodes. Originally, Quillian²⁴⁷ and Shapiro and Woodmansee²⁴⁸ proposed semantic nets as a knowledge representation technique. While many kinds of networks exist, they all share the following features:

- All contain a data structure of nodes representing concepts — generally, a hierarchy of nodes connected by ISA and other property links.
- All contain specialized inferential procedures operating on the data structure of nodes. This involves the inheritance of information from the top levels of the hierarchy downward along ISA links.

Production systems. Production system architecture is the knowledge representation method that has become popular in expert systems. Newell²⁴⁹ advocated this representation scheme as a model of human reasoning. Representing knowledge in this form is a natural way to extract and encode rule-based knowledge in many applications. Also, schema-based systems can be implemented using production system architecture.

A rule-based system contains antecedent-consequent pairs, called "rules." Rule antecedents usually examine data; rule consequents are responsible for data modification. Production systems are rule-based; that is, control structures can be mapped

into relatively simple recognize-act paradigms. Production systems typically have the following three parts: (1) a database, (2) a set of production rules, and (3) an interpreter. The database (working memory) contains all currently true facts. Production rules are of the form antecedent = consequent. The interpreter matches antecedents with the database, selecting the appropriate rule to fire. We can define the execution of such a production system as a series of recognize-act cycles.

Production systems can represent knowledge in several independent modules, allowing greater modularity in the knowledge base than that allowed by any other scheme. In production systems, we can add or delete rules without changing other rules. Since, in most applications, the knowledge base is changed very frequently by adding and deleting rules, the system can be maintained with minimal effort. Furthermore, imprecise rules can be used, with some measure of belief or reliability associated with each rule. The inference mechanism then uses this measure to reach appropriate conclusions. A probabilistic rule-based system, in which a probability is associated with each rule, provides such a mechanism. Such a system can represent heuristic information as a set of probabilistic production rules.

The disadvantage of production systems is that certain rule modules may interfere with other modules. Antecedent parts of several rules may be satisfied simultaneously, and the interpreter must choose the most appropriate rule to be fired. In such cases, programming would be difficult.

Nazif and Levine²⁵¹ describe an expert system for low-level image segmentation — a system whose creation was motivated by the desire to provide an explicit scheme for representing knowledge embodied by numerous segmentation heuristics available in the literature. A rule-based approach provides users with an expressive coding mechanism that accommodates diverse knowledge sources. Such a representational mechanism can capture general-purpose knowledge about image formation and perceptual grouping laws. Production systems in general include condition-action rules that capture general knowledge about an image's independent low-level properties.

Frames (or schemata). Representation for vision applications should be able to capture more structured production systems information and should allow representation of some common knowledge about objects. Minsky²⁵⁰ proposed that a useful knowledge base should be composed of highly modular chunks, called "frames" or "schemata." (The term "schemata" refers to "frames" used in a visual context; however, the literature uses these two terms interchangeably.) A frame is a data structure representing a stereotyped situation or class of objects. Attached to each frame are several kinds of information, such as how to use a frame and what can be expected to happen next. Frames have become popular in applications such as computer vision and natural-language understanding.

A frame can be considered a slot-and-filler structure formed by the following network of nodes and relationships:

- The frame's top levels are fixed and represent things that are always true.
- The frame's lower levels have many slots or terminals that must be filled by specific instances or data (describing aspects of objects). Associated with each slot may be a set of conditions that must be met by any filler for that slot.

Knowledge representation schemes occupy a continuum from totally declarative to totally procedural. In declarative schemes, knowledge is explicit and available for modification. In procedural schemes, the execution flow can be more easily traced; consequently, the control strategy being employed is more readily apparent. Schema-based approaches combine positive aspects of each representation scheme to overcome the limitations of any single method.

Besides knowledge representation, another reason for the interest vision researchers have evinced in schemata is the viewpoint that cognitive structures useful for vision are anticipatory schemata. The schema is perception's central cognitive structure. Neisser²⁵¹ defines schemata as those portions of the perceptual cycle that are internal to the perceiver and modifiable by experience. Schemata accept information as it becomes available at sensory surfaces and are changed by that information; they direct movements and exploratory activities that make more information available, by which information they are further modified.

Perceptual schemata are plans for learning about objects and events and for obtaining more information to fill in the format, or the slot into which a particular value goes. We can regard plans as attached procedures. During the course of perception, the schema's nature evolves from general to precise; we call this evolution "instantiation," since the schema is filled with particular instances of the more general concepts it represents. This process occurs gradually — an important point — as inferences made during one perception cycle dictate the next cycle's course of action.

Schemata are a means by which related facts can be arranged in a manner that eliminates redundancy. Furthermore, schemata can contain directives for using information present within them. Explicit relationships among schemata in a semantic network allow inferences to be drawn based on the interconnection of different nodes (for instance, the direct descendants of a node). Since schemata contain both a declarative part (facts and network links) and a procedural part (methods for manipulating facts), they can exploit the strengths of each part. Many schemata have been developed. We can judge their merits by how well they encode the knowledge required to carry out the tasks for which they have been designed.

CHAPTER 6

Artificial Vision

Research trends

Interpretation of an image, in fact of any perceptual process, uses knowledge and reasoning at every stage. This knowledge could be about entities in an image and the domain of application. Early processing conventionally uses very general models — for example, edges modeled as intensity steps; later processing uses explicit object models. Image interpretation requires knowledge about the following items:

- The environment, including illumination;
- Objects: their geometry and other properties;
- Relationships among different representations;
- The sensing process (image formation); and
- The behavior of operators for different processing tasks.

All systems either explicitly or implicitly use models to represent knowledge about all of these items. Clearly, both the efficacy and the flexibility of a system depend on the models used. The more general the models are, the more flexible the system is. We have selected a set of 4 papers that are representative of papers dealing with topics covered in this chapter. Two are included in this book and the remaining two can be found in the companion book *Computer Vision: Advances and Applications*. We begin this chapter of *Principles* with the paper "Survey of Model-Based Image Analysis Systems," in which Binford discusses many important issues related to the role of knowledge in vision systems, including the strengths and limitations of several systems. In "Perceptual Organization and the Representation of Natural Form," Pentland discusses in *Advances* several issues related to representation of geometric information with the aim of object recognition. Superquadrics were popularized by Pentland in computer vision for representation and recognition of three-dimensional objects.

The major problem with general models is the computational power required to implement and use them. Compared to general models, specialized models result in relatively rigid systems that are computationally more efficient. Early in the game, vision researchers were aware of the issue of computational efficiency, and all debates about top-down versus bottom-up control structures were related to this issue. Of course, in addition to computational efficiency, many other issues should be considered in determining what models to use in a system and how to use them. Considering only the computational aspects first, we can draw a parallel here between general and specific models, and how to use them, on the one hand, and search and knowledge, on the other. Search is very important; however, without adequate use of knowledge, it is computationally impractical. This issue is basic to artificial intelligence. In fact, the best explanation of why complex systems have to abandon the goal of perfection is well advocated by Simon²⁵² in his principle of "satisficing." In vision, the use of general models may provide flexibility, but at the cost of Herculean computation. General models require more computation to use them while specific models require very little computation. Clearly, we need to select models that provide maximum flexibility at a reasonable computational cost.

In "Low-Level Image Segmentation: An Expert System" in *Principles*, Nazif and Levine provide a good example of how shallow, general knowledge can be used for image segmentation. Interestingly, general rules can be used, even at very early stages, in solving complex problems. Irvin and McKeown²⁵³ show how domain-dependent knowledge can be used to solve some complex problems in the interpretation of scenes containing disparate objects. In "The Schema System," Draper et al. in *Advances* describe in detail a knowledge-based vision system that uses many different types of knowledge at different levels to analyze natural scenes.

One purpose of perception is to represent past experiences compactly and retrieve them efficiently when needed. To allow compact and flexible representation, object models should be structured. Structured models may use parametric representation of components. Moreover, at any given instant, the knowledge of the world as constructed by the system is incomplete, inexact, and uncertain. The representation should allow this incompleteness and uncertainty.²⁵⁴ Although the last few years have seen increasing use of knowledge in vision systems, techniques for reasoning about geometric information are still in their infancy.

References Cited Chapter 7

244. R.J. Brachman and B.C. Smith, eds., *SIGART Newsletter, Special Issue on Knowledge Representation*, 1980.
245. A. Barr and E.A. Feigenbaum, *The Handbook of Artificial Intelligence*, William Kaufmann Publishers, Inc., San Mateo, Calif., 1982.
246. E. Rich, *Artificial Intelligence*, McGraw-Hill, New York, N.Y., 1983.
247. M.R. Quillian, "Semantic Memory," in *Semantic Information Processing*, M. Minsky, ed., MIT Press, Cambridge, Mass., 1968.
248. S.C. Shapiro and G.H. Woodmansee, "A Net-Structure-Based Relational Question Answerer," *Proc. Int'l Joint Conf. Artificial Intelligence*, William Kaufmann Publishers, Inc., San Mateo, Calif., 1971, pp. 325-346.
249. A. Newell, "Production Systems: Models of Control Structures," in *Visual Information Processing*, W.G. Chase, ed., Academic Press, New York, N.Y., 1973.
250. M. Minsky, "A Framework for Representing Knowledge," in *The Psychology of Computer Vision*, P.H. Winston, ed., McGraw-Hill, New York, N.Y., 1975.
251. U. Neisser, *Cognition and Reality: Principles and Implications of Cognitive Psychology*, W.H. Freeman, San Francisco, Calif., 1976.
252. H.A. Simon, *The Sciences of the Artificial*, 2nd edition, MIT Press, Cambridge, Mass., 1981.
253. R.B. Irvin and D.M. McKeown, Jr., "Methods for Exploiting the Relationship Between Buildings and their Shadows in Aerial Imagery," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 19, No. 6, 1989, pp. 1564-1575.
254. A.R. Rao and R.C. Jain, "Knowledge Representation and Control in Computer Vision Systems," *IEEE Expert*, Vol. 3, No. 1, 1988, pp. 64-79.