PERGAMON

Pattern Recognition 33 (2000) 1511–1524

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Weighted matchings for dense stereo correspondence[☆]

Gabriel Fielding, Moshe Kam*

Data Fusion Laboratory, Department of Electrical and Computer Engineering, Drexel University, 3141 Chestnut St, Philadelphia, PA 19104, USA

Received 8 January 1999; accepted 1 June 1999

Abstract

The calculation of matches between pixels, points, or other features in stereo images is known as the correspondence problem. This problem is ill-posed due to occlusions; not every pixel, point or feature in one stereo image has a match in the other. Minimization of a cost function over some local region and dynamic programming algorithms are two well-known strategies for computing dense correspondences. However the former approach fails in regions of low texture, while the latter imposes an ordering constraint which is not always satisfied in stereo images. In this study, we present two new techniques for computing dense stereo correspondence. The new methods are based on combinatorial optimization techniques which require polynomial computation time. The first method casts the selection of matches as the *assignment problem*, solved efficiently by finding a maximum weighted matching on a bipartite graph. The second is a *greedy algorithm* which computes suboptimal weighted matchings on the bipartite graphs. Both methods use *occlusion nodes* when no matches exist. The resulting disparity maps have desirable properties such as dense correspondence, while avoiding the drawbacks associated with ordering constraints. Three existing matching approaches are also reviewed for comparative purposes. We test all five techniques on real and synthetic stereo images using performance criteria which specifically measure occlusion detection. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Stereo vision; Dynamic programming; Matching algorithms; Bipartite graph; Weighted matching; Greedy matching

1. Introduction

A fundamental problem in stereo vision is to determine where corresponding pixels, points, or other features occur in images taken from stereo cameras. Feature-based methods identify particular features (e.g., point, corner, line) in each image and perform feature matching, thus establishing correspondence on a sparse set of points in the image. Correlation-based stereo[1] compares intensity values within local windows in stereo images in order to establish correspondence. The use of windows avoids numerous problems with feature extraction and results in a dense set of range information about the scene.

The matching phase of a stereo algorithm often requires that a local feature (or local region) in one image be compared with several candidates in the other image in search of a match. In correlation-based stereo the first step is to get an integer estimate for d in the relation $R(y, x_r + d) = L(y, x_l)$. As the true value of d may not be an integer, there are several algorithms that refine the estimate, using either interpolation [2], iterative schemes such as relaxation [3], or more complicated adaptive

[☆]This work was funded by an ONR sponsored NDSEG research fellowship, by the National Science Foundation through grants ECS 9057587, ECS 9216588, and ECS 9512363, and by the Electric Power Research Institute through grant SF 958.

*Corresponding author. Tel.: +215-895-6920; fax: +215-895-1695.

E-mail address: kam@minerva.ece.drexel.edu (M. Kam).

[1] Also called area-based stereo by some authors [1].

0031-3203/00/$20.00 © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.
PII: S0031-3203(99)00132-6

matching techniques [4]. Their success often depends on the closeness of the initial estimate to the correct value of d. Most of these techniques are too slow for real-time stereo, and hence we focus here on deterministic algorithms that provide fast integer estimates of d.

A very popular approach to solving the correspondence problem in stereo is the *minimization of sums-of-squared differences* [5]. This method is easily implemented and has been used for real-time navigation of mobile robots [6]. Another matching approach [7] uses *dynamic programming* with a *monotonic ordering constraint*. Matching is carried out by minimizing a performance index for an entire epipolar line. Thus, locally "good" matches may be ignored in favor of a matching that is "better" for the whole line. The monotonic ordering constraint reduces the computational complexity but, as we shall show, it faces difficulties when narrow occluding objects are present.

In this paper we introduce two alternative matching techniques which maximize cost functions along the entire epipolar line. The first technique casts stereo matching as an integer programming problem whose solution can be found in polynomial time through a maximum weighted matching on a bipartite graph [8].[2] The second technique is a suboptimal greedy matching algorithm whose matching performance is comparable to both dynamic programming and maximum weighted matching, yet it has lower computational complexity. The two new matching algorithms are compared to existing matching strategies using both synthetic and real images.

2. Camera geometry and terminology

We use the pinhole camera model [10] and assume that surfaces are Lambertian [11]. When a light ray from a source is reflected off a surface and passes through the pinhole lens of a camera onto the sensor array, we call the subsequent pixel representation of the reflected ray a *projection* of the surface. Hence, an arbitrary object in space (i.e., a wall, container) will be represented by a collection of pixels in the image. We make the assumption that every pixel is a projection of one surface.[3] A more detailed description of image formation may be found in the literature for image synthesis (e.g., Ref. [12]).

[2] The Hungarian method can also be used, with inferior bounds on complexity, see Ref. [9].

[3] This is not true in general because the size of the pixel allows it to collect light from a volume in space which may include two or more surfaces at different distances from the cameras. These pixels usually occur on the boundaries of an object's projection in the image and constitute only a small percentage of the total pixels in the image.

Fig. 1 shows the image planes of two cameras in a parallel stereo configuration. The lenses are represented by points $\mathbf{f}_l$ and $\mathbf{f}_r$ which are drawn behind the image planes in order to simplify the projections. Suppose that we are looking for the point in space that created the projection in the right image given by point $\mathbf{a}$. Any point along the ray formed by $\mathbf{f}_r\mathbf{a}$ could have created that projection. Furthermore, we assume that the point in space that created point $\mathbf{a}$ has also created some point in the other image plane. The triplet, $\mathbf{f}_r$, $\mathbf{a}$ and $\mathbf{f}_l$ define a unique plane. The intersection of this plane with the left image and right image defines two lines known as *epipolar lines*. The epipolar lines represent the possible locations of corresponding projections. Thus, the set of pixels that need to be checked for similarity in the two images lie along predetermined lines. Given two corresponding epipolar lines, we seek to determine the correspondence for each pixel on the two lines.

A pair of stereo cameras can be *registered* such that corresponding epipolar lines are known in the images from each camera. This registration requires a calibration procedure (e.g., Refs. [13,14]). Here, we assume that registration has been performed already, and corresponding epipolar lines have been mapped to horizontal rows in the image.

We denote the left image $L(y_l, x_l)$ and the right image $R(y_r, x_r)$. The data for an image is arranged as a uniformly spaced array of integers where each integer represents the brightness of the image at that array element (pixel). Since we assume that calibration has been done, we set $y = y_r = y_l$. The dimensions of the images are $M \times N$ (M rows, N columns). We denote the minimum disparity and maximum disparity as $d_{\min}$ and $d_{\max}$, respectively, with the total range of disparities given by $\delta = |d_{\max} - d_{\min}| + 1$.

We define the disparity map $D_r(y, x_r)$ as an integer-valued array whose elements specify the offset of a pixel in the right image to its "match" in the left image (we use the right image as the reference). This means that given

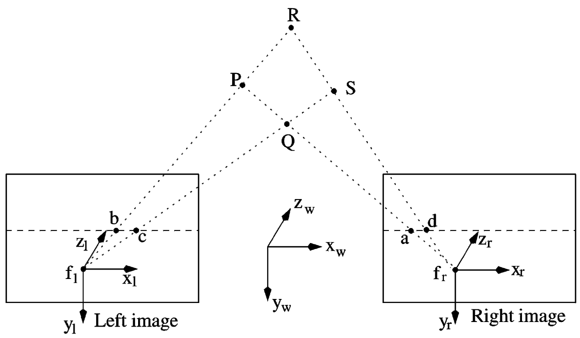


Fig. 1. Epipolar lines (shown as dashed lines) in parallel stereo camera systems.

pixel (y, x_r) in the right image, the matching pixel in the left image is located at pixel $(y, x_r + D_r(y, x_r))$; that is, $x_l = x_r + D_r(y, x_r)$. $D_r(y, x_r)$ is undefined for pixels which have no match. All matching algorithms in this paper compute integer-valued disparities. Since the true displacement is a real number, interpolation techniques can improve the *precision* of the disparity estimate [4].

3. Similarity functions

To quantify the similarity of corresponding pixels in two images, we use the following popular *similarity functions*:

The normalized correlation (*NC*):

$$f_{NC}(y, x, d) = -\sum_{i,j \in W} \frac{(R(y+j, x+i) - \mu_r)}{\sigma_R} \frac{(L(y+j, x+i+d) - \mu_l)}{\sigma_L}. \quad (1)$$

The sum-of-squared differences (*SSD*):

$$f_{SSD}(y, x, d) = \sum_{i,j \in W} [R(y+j, x+i) - L(y+j, x+i+d)]^2. \quad (2)$$

The sum-of-absolute differences (*SAD*):

$$f_{SAD}(y, x, d) = \sum_{i,j \in W} |R(y+j, x+i) - L(y+j, x+i+d)|. \quad (3)$$

In Eqs. (1)–(3), W is a window around the pixel (y, x) given by $W = \{(i, j): -p \leqslant i \leqslant p, -q \leqslant j \leqslant q\}$. Here, μ_l, μ_r are the mean pixel values over the left and right windows, and σ_l, σ_r are the standard deviations of the pixel values in the left and right windows, respectively. Each of the functions in Eqs. (1)–(3) takes a minimum value when the image intensities in the left and right windows are identical. For Eqs. (2) and (3) the minimum value is zero and for Eq. (1) this value is -1.

4. Matching algorithms

In Fig. 2 we have arranged the left and right epipolar lines to show one representation of a surface observable in both stereo views. This depiction of matching between epipolar lines was first given in Ref. [15]. Since each element on an epipolar line is discrete because of the sampling process, we may use the coordinate along that line as an index into a two-dimensional array. The coordinates x_l and x_r specify a unique element in the array. We fill each element in the array with the value from the similarity function as described in Section 3. we call the array the *cost matrix*, c, where given a similarity function f_ξ (where ξ stands for SSD, SAD, or NC) and a row of interest y:

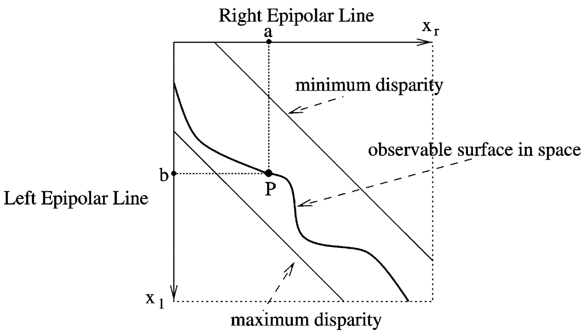


Fig. 2. Epipolar line matching for stereo.

$$c(x_l, x_r) = -f_\xi(y, x_r, x_l - x_r). \quad (4)$$

Any point, on a surface in space, that is "matchable" will have an ordered pair (x_l, x_r) of left and right epipolar coordinates that specify the cost of the match through (4). This ordered pair also defines that point's location in world coordinates through the geometry of stereo cameras. For example, **P** in Fig. 2 corresponds to the pairing (**b**, **a**). Hence, determination of the corresponding pair (x_l, x_r) determines the surface location. Since not every point in space is matchable (e.g., occlusions), some elements of the left and right epipolar lines have no corresponding match. To account for this, we define an *occlusion cost* that effectively screens out features with no good matches. Computation of this occlusion cost depends on the similarity function being used and is discussed in the Appendix A.

We present five matching techniques. The first three (local search, the left–right heuristic, and dynamic programming) are from the stereo-matching literature; the last two (maximum weighted matching and greedy matching) are the contributions of this study.

4.1. Existing matching methods

The existing matching methods we review are *local search* [5,6] a modification of local search called the *left–right heuristic* [16,17], and dynamic programming [18,7].

4.1.1. Local search [5,6]

A widely used approach for determining disparity is to compute the values of a similarity function for each candidate pixel and to choose the disparity that corresponds to the minimum of that function. Using the right image as the reference for our disparity map, we compute the disparity on row y as

$$D_r(y, x_r) = \underset{d_{\min} \leqslant d \leqslant d_{\max}}{\operatorname{argmin}} \; c(x_r + d, x_r). \tag{5}$$

The similarity functions are not guaranteed to be unimodal, and the global minimum may not be unique. Furthermore, the correct match may correspond to a value other than a minimum of the similarity function. When there is very little texture in a scene, the similarity function may have a wide shallow shape with a plateau of indistinguishable minima. Various methods of coping with these limitations of the similarity functions have been attempted. One approach [19] is to require that the minimum of the similarity function be "lower" than any other value by some percentage. This approach fails when the correct match is not the global minimum. In Table 1 we give the pseudo-code description for local search incorporating the threshold for the occlusion cost, C_O.

If each pixel has δ potential matches then finding the "best" match through minimization requires $\delta - 1$ comparisons. With N pixels on the epipolar line, the overall complexity of local search is $O(\delta N)$.

4.1.2. The left–right heuristic [16,17]

Disparity values for corresponding pixels should be equal in absolute value but have opposite algebraic signs [17]. The search for a local minimum as described in Table 1 yields the minimum in each *column* of the epipolar cost matrix. Searching for a minimum along the *row* of the cost matrix would yield a disparity with respect to the left epipolar line. This suggests an algorithm that requires the left and right disparities to agree; that is, if the minimum in row k occurs in column j then we require that the minimum in column j be row k. One stereo algorithm [16] used a strategy to refine disparity estimates by horizontally flipping the left and right images and then re-analyzing them by using the flipped left image as the new right image and the flipped right image as the new left image. Instead of refining estimates, we save those matches that agree between the right and left epipolar lines. We call this method the *left–right heuristic* (LRH).[4] We define two vectors LeftMate and RightMate whose elements are the index of the matching pixel on the opposite epipolar line. If separate minimizations along rows and columns produced the same match then we would have RightMate[LeftMate[i]] = i for each row i. In Table 2 we give the pseudo-code description of the left–right heuristic, again including the occlusion cost.

[4] Agreement between left and right disparities is less likely when the surface is sharply sloping away from the stereo cameras.

Table 1
Algorithm for local search with occlusion cost threshold

```
for j = 1 to N do
  if ($\min_k$ c(k,j)$C_O$), RightMate[j] = $\operatorname{argmin}_k$ c(k,j)
  else RightMate[j] = UNMATCHED
```

Table 2
Left–right heuristic algorithm

```
for j = 1 to N do RightMate[j] = $\operatorname{argmin}_k$ c(k,j)
for i = 1 to N do LeftMate[i] = $\operatorname{argmin}_k$ c(i, k)
for j = 1 to N do
  if (LeftMate[RightMate[j]] $\neq$ j) AND c(RightMate[j], j) $\leqslant C_O$) then
    RightMate[j] = UNMATCHED
```

As with local search, each of the N pixels on an epipolar line has δ possible matches. For each of these pixels, we perform $\delta - 1$ comparisons to find the minimum. The *left–right heuristic* requires minimization for each row and column in the cost matrix and then N comparisons to check for agreement. So the overall complexity is $O(\delta N)$.

4.1.3. Dynamic programming [7,18]

Dynamic programming (DP) is a method of organizing an optimization problem to exploit the recursive structure of necessary calculations [20]. Several studies have used this matching approach for stereo matching (e.g., Refs. [7,21]). Central to the use of DP is the *monotonic ordering constraint* which states that if x_{r_i} matches x_{l_j} then $x_{r_{i+1}}$ can only match $x_{l_{j+k}}$ with $k \geqslant 1$. Without the monotonic ordering constraint, one is required to perform an exhaustive search. The computational complexity of the dynamic programming approach with the monotonic ordering constraint is $O(\delta N)$ [7].

Fig. 3 shows a graph on which the shortest path between the vertices labeled S and T can be calculated using DP. The dotted edges allow for occlusions to occur; their weights are given by the occlusion cost C_O. We associate the weight from the epipolar cost matrix, $c(x_l, x_r)$, with the diagonal (solid) edges. While a Dijkstra-type shortest path algorithm [20] solves this problem, a more efficient algorithm is possible because of the regular structure of the graph. Here we show the solution using DP; for a more detailed discussion see Refs. [7,21] (see Table 3).

Each vertex is labeled with its predecessor using the array P(i,j) so we can backtrack from vertex

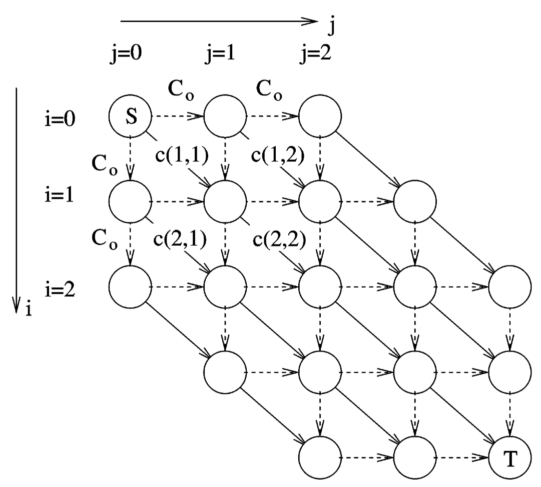


Fig. 3. A graph representation of the stereo matching problem where the similarity between pixels is represented as a weight on the diagonal edges of the graph. The dotted edges allow for occlusions to occur and their weights are given by a cost parameter, C_O.

T = (N, N) to find which edges are in the shortest path. A horizontal edge corresponds to an occlusion in the right disparity map and a diagonal edge represents a matching in the disparity map.

Since DP is effectively searching for a shortest path through a graph, any match declared on an epipolar line affects subsequent matches. Of critical importance in DP solutions to stereo matching is the choice of the *cost of occlusion*, C_O since performance can vary significantly when C_O is changed. Setting C_O too large results in no occlusions—the shortest path follows the direct diagonal path from S to T. Setting C_O too low results in selecting occlusions *only*. In Ref. [7], the authors propose a maximum-likelihood framework for selecting the value of C_O. In Appendix A, we expand on this method to suggest a cost which allows operation with lower SNR than provided for in Ref. [7].

The monotonic ordering constraint effectively keeps local disparities near each other without using a regularization framework [22] or a relaxation technique [3], both of which are time consuming. The monotonic ordering constraint is *not satisfied* where the occluding objects are narrow. Use of DP under these conditions often results in a disparity map that partially detects or completely misses narrow objects. In Fig. 4(a) we show a matching for two epipolar lines. Segment **p** can be visualized as the surface of a narrow pillar which is visible in both images. The right epipolar line has an occluded region **q** which has no matches along the left epipolar line. The monotonic ordering constraint forces us to choose either path 1 or path 2 as shown in Fig. 4(b). The surface corresponding to path 1 "detects" the object but fails to correctly match a large portion of the surface further from the cameras. The surface corresponding to path 2 fails to "detect" the object completely.

Consider the synthetic images in Fig. 5(a) and (b). Here, three narrow occluding objects are present in a fairly textured environment. The right obstacle poses a challenge for DP for the reasons just explained. We compare the true disparity map in Fig. 5(c) with results from a dynamic programming-based matching algorithm in Fig. 6; clearly the right-hand side obstacle is mostly undetected.

4.2. New matching methods

In this section we describe two new matching strategies for calculating dense correspondences in the epipolar array. Both methods maximize a cost function over the epipolar array; the first method computes the maximum

Table 3
Algorithm for dynamic programming-based stereo matching (adapted from Ref. [7])

```
for i = 0 to N do P(i, 0) = 2; V(i, 0) = i*$C_O$; Initialize 1st column of vertices
for j = 0 to N do P(0, j) = 3; V(0, j) = j*$C_O$; Initialize 1st row of vertices
for i = 1 to N do
  for j = 1 to N do
    v1 = V(i − 1, j − 1) + c(i, j)  Matching edge cost
    v2 = V(i − 1, j) + $C_O$  Vertical edge/Occlusion
    v3 = V(i, j − 1) + $C_O$  Horizontal edge/Occlusion
    V(i, j) = min(v1, v2, v3)  Choose shortest path
    P(i, j) = argmin(v1, v2, v3)  Assign predecessor (1, 2, or 3)
i = N; j = N;
while (j > 0)  Backtrack from vertex T
  if P(i, j) = = 1 then RightMate[j] = i; i = i − 1; j = j − 1;
  else if P(i, j) = = 2 do i = i − 1;
  else RightMate[j] = UNMATCHED; j = j − 1;
```

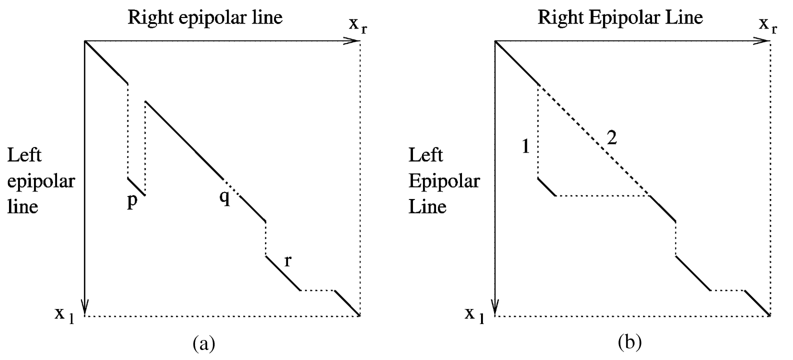


Fig. 4. (a) One epipolar line with two occluding objects and an occluded region **q** created by the object **p** (b) the DP solution has two possible paths, choosing (1) leaves a large region unmatched and choosing (2) leaves the surface undetected.

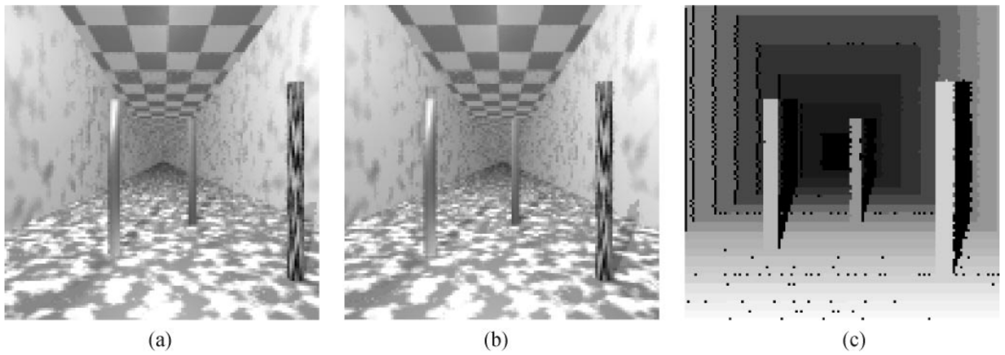


Fig. 5. A synthetic stereo pair with narrow objects, (a) left camera, (b) right camera, and (c) the true integer disparity map with respect to the right image.

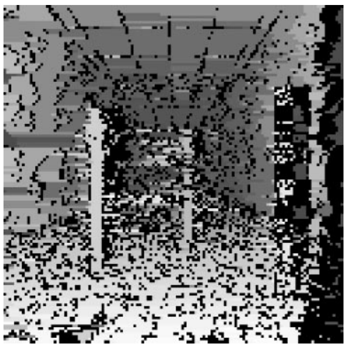


Fig. 6. Disparity map generated via the DP approach presented in Ref. [7] using SSD with 3 × 3 windows.

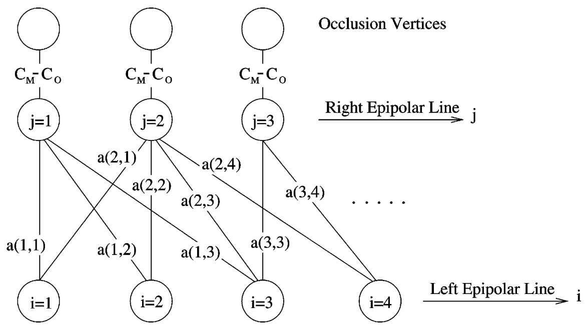


Fig. 7. A representation of the correspondence problem as a weighted graph.

weighted matching while the second method uses a greedy strategy to find a near-optimal maximum weighted matching.

4.2.1. Maximum weighted matching and the Hungarian method

Fig. 7 shows a graph representation of the matching problem. Pixels from the left and right epipolar lines are drawn as nodes (circles) and potential matches between those pixels as drawn as edges (lines). With each edge we associate the weight $a(x_l, x_r) = C_M - c(x_l, x_r)$ where C_M is an upper bound on the appropriate similarity function. We have also linked an "occlusion" node to each node in right epipolar line with edge cost $C_M - C_O$. By doing this, we can include the occlusion cost as a matchable node.[5] The calculation of disparities can be viewed as a weighted matching problem on a bipartite graph. We desire to find the set of edges for which the sum of these chosen edge weights is maximum while no two edges touch the same vertex (pixel).

Finding an optimal matching for bipartite graphs is known in the combinatorial optimization literature as the *maximum weighted matching* (MWM) *problem* [23]. Formally, a bipartite graph $G = (V, U, E)$ has two distinct sets of nodes (pixels) V and U and a set of edges E representing potential matches. Associated with each edge $e \in E$ is a cost c_e. This cost is the same as defined earlier where $c_e = a(x_l, x_r)$ with $x_r \in V$ and $x_l \in U$. A *matching* is defined as a subset of edges ($\tilde{E} \subseteq E$) such that no two edges share a vertex. A *maximum weighted matching* has $\sum_{e \in \tilde{E}} c_e$ maximum.

We define an indicator matrix, $g(x_l, x_r)$, where $g(x_l, x_r)$ is 1 if we decide that the pixels (y, x_r) and (y, x_l) match and 0 if they do not. We require that all matches be unique, namely if $g(x_l, x_r) = 1$, then $g(k, x_r) = 0$ for all $k \neq x_l$. Likewise, if $g(x_l, x_r) = 1$, then $g(x_l, k) = 0$ for all $k \neq x_r$. We can cast the problem of choosing assignments for pairs of pixels as the following integer programming problem:

$$\begin{aligned}
\text{Maximize: } & \sum_{x_l}\sum_{x_r} a(x_l, x_r)g(x_l, x_r) \\
& \quad + (1 - g(x_l, x_r))(C_M - C_O) \\
\text{Subject to: } & g(x_l, x_r) \geqslant 0 \quad \forall x_l, x_r, \\
& \sum_{x_r} g(x_l, x_r) = 0 \text{ or } 1, \text{for all } x_l, \\
& \sum_{x_l} g(x_l, x_r) = 0 \text{ or } 1, \text{for all } x_r. \qquad (6)
\end{aligned}$$

In an earlier study [9] we solved this integer programming problem in polynomial-time by using the 'Hungarian method' of König and Egvary (e.g., Ref. [24]). The algorithm is based on primal-dual methods given by Lawler [8] which terminates with a matching of maximum weight, but not necessarily one of maximum cardinality. The complexity of the algorithm is $O(N^3)$ where N is the number of pixels on an epipolar line (see Table 4).

Table 4
Maximum weighted matching

```
create U and V vertices (including occlusion vertices)
assign edge weights, E, between vertices using c(x_l, x_r)
  or C_O
construct graph, G = (V, U, E)
compute maximum weighted bipartite matching on G;
for each node v ∈ V,
  if mate[v] ∈ Occlusion vertex
    mate[v] = UNMATCHED;
```

The integer program problem in Eq. (6) and the weighted bipartite matching problem are equivalent [24]. We use a graph algorithm library [25] with an efficient implementation of a bipartite maximum weighted matching algorithm whose complexity is $O(\delta N^2)$. Hereafter we shall refer to both of these approaches as maximum weighted matchings (MWMs).

4.2.2. A greedy algorithm for weighted matchings

Greedy algorithms represent a well-known alternative to global optimization procedures [20]. We show how to apply a greedy strategy to stereo matching using the epipolar cost matrix. This method builds a matching by progressively adding edges with maximum weights in the cost matrix $a(x_l, x_r)$ that satisfy the criteria for a feasibly matching. This method differs from the maximum weighted matching approach in which the final matching weight is not guaranteed to be maximum.

Fig. 8 shows an example of how a greedy matching can differ from a maximum weighted matching. The resulting matching is *not* guaranteed to be globally optimal, however the matching given by the greedy algorithm is typically very close to the optimal matching and the algorithm has lower computational complexity.

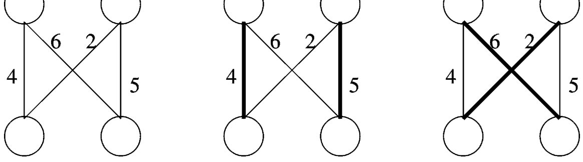


Fig. 8. The greedy matching approach. Given the graph on the left with four edges, there are two possible matchings. The matching in the middle is the maximum weighted matching whose weights total 9. The matching on the right is the greedy matching with weight 8. A greedy matching strategy would first add the edge with largest weight (in this case, the edge with value 6). The edges with weight 5 and 4 are considered next but do not satisfy the criterion for a valid matching. This leaves only the edge with weight 2 to be added for a total matching of weight 8.

[5] Alternatively, we could have linked the occlusion nodes to the left epipolar line which would result in a different matching; however, our experiments have shown that the differences in final matchings using the two linking schemes are insignificant. The same is true for adding *both* left and right occlusion nodes, with each edge having the occlusion cost $C_M - (C_O/2)$.

Table 5
Greedy algorithm for stereo matching

```
for j = 1 to N, Rightmate[j] = UNMATCHED    Initialize labels
S = sort($c(x_l, x_r)$)    Sort costs by value
while S is not empty,
  Remove maximum edge, a(i, j) from S    Pick best edge.
  if (a(i, j) < $C_M - C_O$), then break;    If below threshold, done;
  if (Rightmate[j] = = UNMATCHED) then    otherwise, if not matched
    Rightmate[j] = i    assign match
```

Given a bipartite graph $G = (V, U, E)$ and an associated cost with each edge, c_e, $e \in E$, the *greedy matching algorithm* proceeds as follows: Let $\tilde{E} = \emptyset$ and $M = E$. While there exists $e \in M, e \notin \tilde{E}$, such that $\tilde{E} \cup e$ is a feasible matching, let $e = \text{argmax}\{c_e: \forall e \in M\}$, and set $\tilde{E} = \tilde{E} \cup e$ and $M = M - \{e\}$. Implementation of this algorithm is given in Table 5.

Since there are about δN edges, the running time for the sorting phase is $O(\delta N \log(\delta N))$. The list scanning operation takes $O(\delta N)$ operations. So the overall complexity is $O(\delta N \log(\delta N))$ for each epipolar line.

5. Performance assessment

In order to quantify the performance of the five matching approaches, we have run the algorithms on synthetic 8-bit greyscale stereo images with random intensities and flat surfaces with random disparities. The images are 128×128 pixels with the intensity of background pixels uniformly distributed between 0 and 255. The resulting image is filtered with a 5×5 pixel Gaussian filter [26] to produce a smoothly varying pattern. Each image has between 1 and 10 objects in it. Each object has a random height and width, between 5 and 20 pixels. Each object has a distinct intensity pattern generated in the same way as the background pattern and the object has a random disparity (between 5 and 20 pixels). The disparity search range for all algorithms is between 0 and 39 pixels ($\delta = 40$). All algorithms used 3×3 windows and SSD as the similarity function. The cost of occlusion was set to $C_O = 542$ as discussed in Appendix A. In Fig. 9 we show a sample of two of the random stereo images that we have used, along with the true disparity map. Needless to say, the synthetic images that we experimented with do not exhaust all possible test data; however, they do provide quantitative measures to assess matching algorithm performance when subject to noisy stereo data with occlusions present.

In the computed disparity image, we have two types of *labels* for each pixel: *occluded* and *non-occluded*. Only pixels with a non-occluded label can be associated with a disparity. These two labels can be interpreted as corresponding to the two hypotheses: $H_0 \leftrightarrow$ occluded and $H_1 \leftrightarrow$ not occluded. A match is a binary decision: $D_0 \leftrightarrow$ pixel is occluded and $D_1 \leftrightarrow$ pixel is not occluded. For our synthetic stereo images we know the true matching labels for all pixels and we compute the statistics $P(D_1|H_0)$ and $P(D_1|H_1)$. The statistic $P(D_1|H_1)$ is the probability of *correct detection* (CD) and $P(D_1|H_0)$ is the probability of *false alarm* (FA). In the case of H_1 and D_1, we compute mean-squared-error (MSE) as a measure of matching accuracy.

In Table 6 we show the performance of the matching algorithms with $C_O = 542$ (see Appendix A) averaged over 2000 randomly generated stereo pairs. The first column gives the false alarm rate, the second column the correct detection rate and the third column shows the mean-squared-error. The fourth column gives the computational complexities of the five algorithms.

Local search has the highest false alarm rate as well as the highest detection rate and largest mean-squared-error. Compared to local search, the *left–right heuristic* improves the false alarm rate significantly; the reason is that the left and right disparities disagree in regions of occlusion and the LRH detects these disagreements. Detection of occlusions comes at the expense of a lower rate

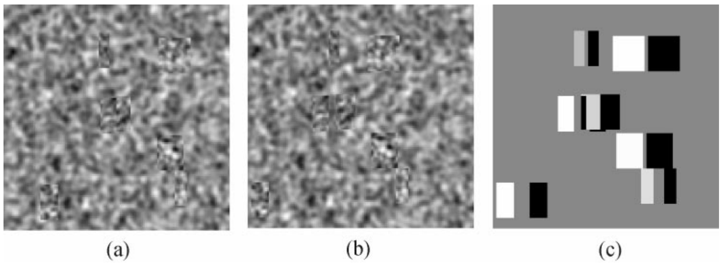


Fig. 9. A textured synthetic stereo pair [128×128], (a) left, (b) right, and (c) the true integer disparity map with respect to the right image. Occluded regions are marked in black.

Table 6
Performance of matching algorithms on the textured synthetic stereo test set

Algorithm	False alarm	Correct detection	MSE	Complexity
LS	0.1783	0.9449	4.2920	$O(\delta N)$
LRH	0.0404	0.7604	2.2887	$O(\delta N)$
DP	0.0735	0.9166	1.4239	$O(\delta N)$
MWM	0.0543	0.8832	2.4745	$O(\delta N^2)$
Greedy	0.0540	0.8882	2.9337	$O(\delta N \log(\delta N))$

of correct detections since some correct matching decisions are wrongly labeled occlusions.

Dynamic programming has a low probability of false alarm, a high probability of correct detection, and a low MSE. MWM has a lower probability of false alarm than DP but also a lower probability of correct detection. The greedy algorithm's performance is virtually identical to that of MWM.

Since algorithm performance is dependent on variations in occlusion cost, we have also computed the ROC (receiver operating characteristic) for the matching algorithms as C_O varies, as well as the variation in MSE with C_O. These plots are given in Fig. 10.

In Fig. 10(a) we have plotted the probability of correct detection against the false alarm rate with C_O as a parameter. The global matching strategies cluster together, yielding similar performances while local search (the innermost curve) has the worst performance. LRH initially stays close to the performance of the global matching approaches and then levels off quickly. While its false alarm rate never exceeds 0.10, its correct detection rate also never reaches 0.8.

In Fig. 10(b) we show the mean-squared error as a function of the parameter C_O. For values of C_O around the threshold of 542 (computed in Appendix A), DP, LRH, and MWM each have relatively low MSE. LRH has a low MSE over a large range of occlusion costs. The greedy algorithm, while similar to MWM, has a consistently higher MSE for a given C_O. While local search initially has the highest MSE, the MSE for dynamic programming continue to rise with the threshold C_O and eventually surpasses even local search. This is a result of *no occlusions* being chosen for large C_O and hence, the shortest path becomes the direct path from S to T in Fig. 3. All other algorithms have bounded errors as C_O varies. One conclusion of Table 6 and Fig. 10 is that the two new algorithms will be preferred for values of C_O exceeding 3000.

In Table 7 we show performance statistics at a fixed false alarm rate; we also give the occlusion cost for which that false alarm rate was achieved. The left–right heuristic is omitted because it cannot operate in the region specified. In this case, the global matching approaches offer superior performance.

While these statistics provide measures for the performance of different matching algorithms, none of them fully characterizes matching performance. For example, as was pointed out with DP, narrow occluding objects may be completely missed while maintaining an acceptable *average* correct detection rate. In the next section, we examine the qualitative performance of the matching algorithms using a variety of real and synthetic images.

6. Examples

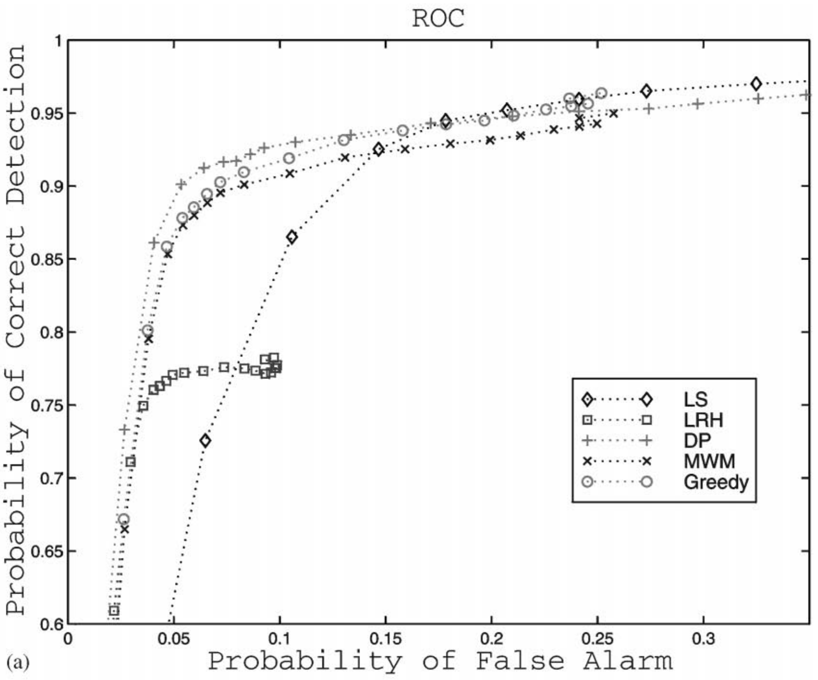
6.1. Narrow occluding objects

One of the major benefits of dynamic programming is that the ordering constraint is usually satisfied in real-world environments; however, narrow occluding objects in the foreground of the image pose a problem, as discussed in Section 4.1.3. In Fig. 11, we show synthetic stereo images that have two narrow occluding objects. The results of stereo analysis with a disparity range of 40 pixels and $C_O = 4875$ are shown in Fig. 12. Minimization with occlusion estimation in Fig. 12(a) still mislabels all of the points in the occluded regions, producing "ghosts". The left–right heuristic in Fig. 12(b) produces a disparity map with fewer false-alarms in the occluded regions yet fewer correct detections of matchable pixels. Dynamic programming in Fig. 12(c) has failed to detect most of the left obstacle. Both MWM and the greedy algorithm, in Figs. 12(d) and (e), respectively, correctly match both obstacles although they have more "noise" artifacts as a result of their matching strategy. In situations like these, the new algorithms have a clear advantage over existing techniques.

6.2. Low-texture regions

Correlation-based stereo methods encounter difficulties with regions which have very little texture because the similarity functions produce multiple minima. In Fig. 13 we present a rendered scene where the floor is relatively featureless except for several shadows. The true disparity map is given in Fig. 13(c) with occluded regions marked in black. In Fig. 14 we show the disparity maps generated using the five methods described in this paper.

All algorithms use SSD with 3×3 windows, $\delta = 40$, and $C_O = 4870$. The disparity map generated using local search is given in Fig. 14(a). The majority of the floor in the scene is matched incorrectly. The correct disparity is found where there are shadows cast by the objects (providing intensity variation which allows for a correct matching). Outside these regions, the local search fails



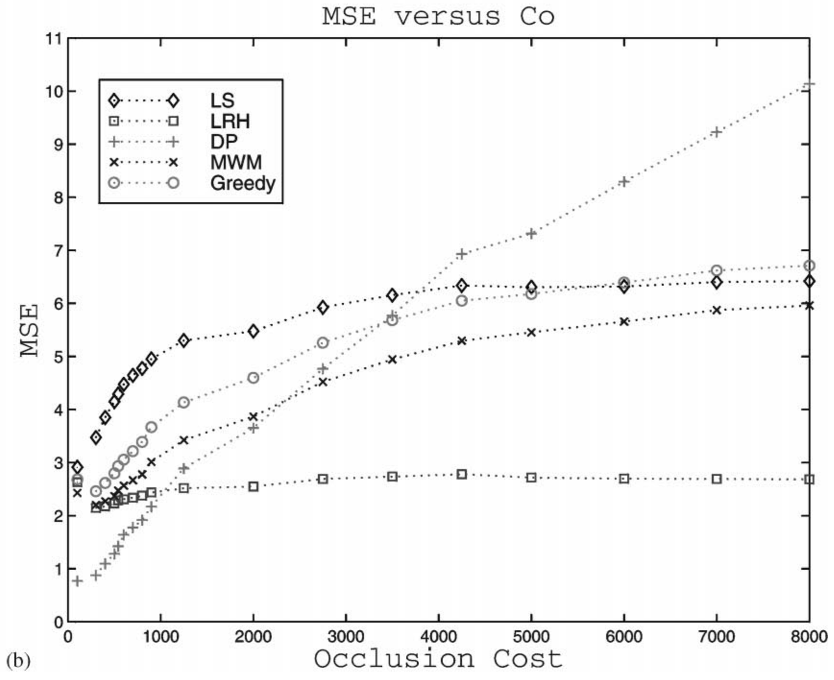


Fig. 10. Performance curves for the the matching algorithms in this paper when applied to images of the type shown in Fig. 9. Receiver operating characteristic in (a) gives probability of correct detection versus probability of false alarm. Mean-square error versus occlusion cost is shown in (b).

to find the correct matches. In Fig. 14(b) the left–right heuristic has eliminated false matches on the floor. However, we are left with very little disparity information in this region. Dynamic programming has correctly matched the floor, although artifacts from the matching process result in some incorrect matches on the left and right obstacles. Fig. 14(d) show that MWM correctly matches most of the floor. The disparity map from the greedy

Table 7
Performance measures for a constant false alarm rate of 0.10

	False alarm	Correct detection	MSE	Occlusion cost
LS	0.10	0.8355	3.7978	386.9
DP	0.10	0.9282	2.0439	851.2
MWM	0.10	0.9070	3.1332	1173.3
Greedy	0.10	0.9172	3.8388	1178.5

algorithm is shown in Fig. 14(e). While similar to MWM, the matching performance on the floor is not as good as that provided by MWM.

In Table 8 we have computed the performance statistics described in the previous section for the disparity maps in Fig. 14. These statistics show that the algorithms offer similar performance characteristics on these rendered images as for the synthetic images in the previous section. Although LRH has the lowest MSE, it only matches about half of the points in the image. Both MWM and DP have low MSE (relative to local search).

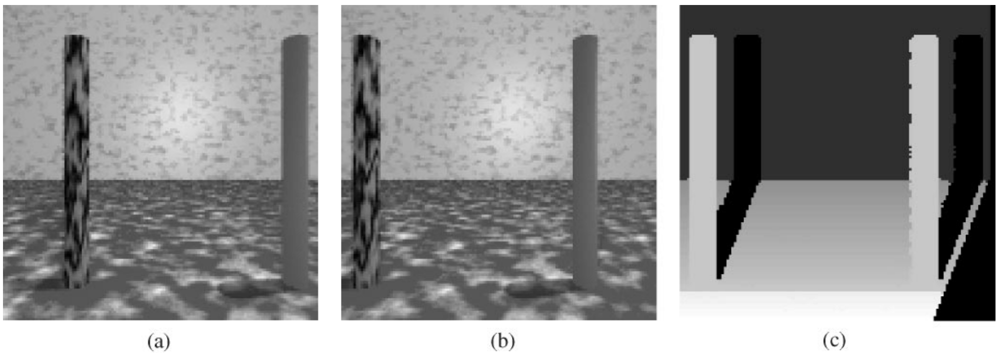


Fig. 11. Synthetic images [128 × 128 pixels] with two narrow occluding objects: (a) left, (b) right, and (c) truth with occlusions marked in black.

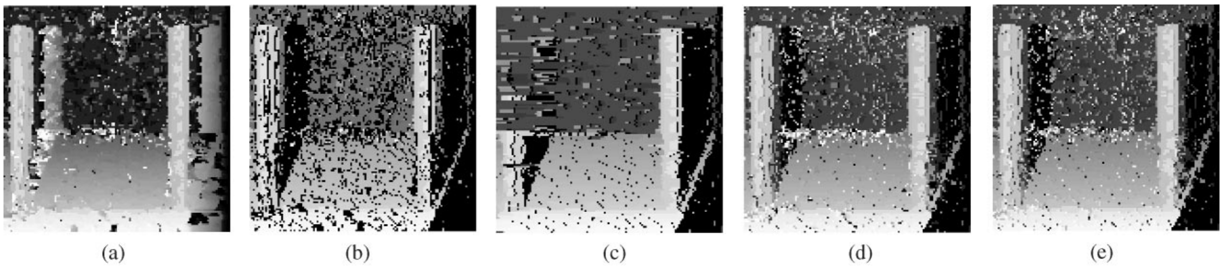


Fig. 12. Disparity maps using: (a) local search, (b) left–right heuristic, (c) dynamic programming, (d) maximum weighted matching, and (e) greedy algorithm.

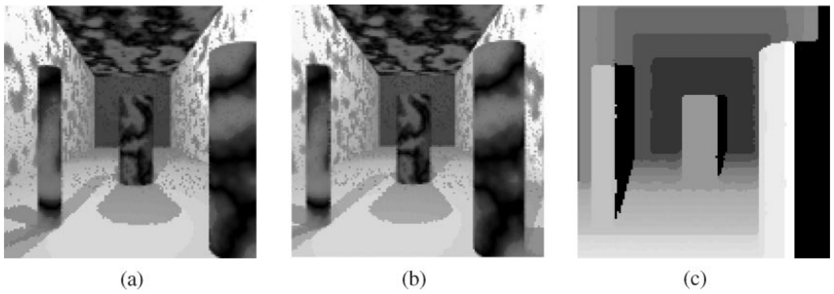


Fig. 13. A synthetic stereo pair with low-texture regions, (a) left, (b) right, and (c) the true integer disparity map with respect to the right image.

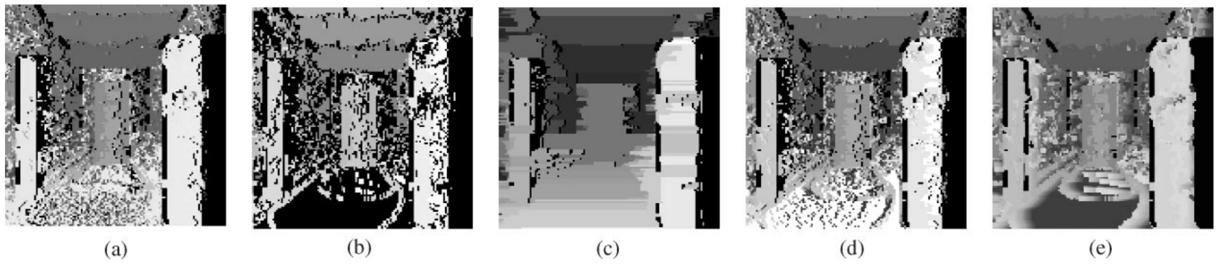


Fig. 14. Disparity maps using: (a) local search, (b) left–right heuristic, (c) dynamic programming, (d) maximum weighted matching, and (e) greedy algorithm.

Table 8
Performance measures for the disparity maps in Fig. 14 compared to the true disparity in Fig. 13

	False alarm	Correct detection	MSE
LS	0.57	0.94	28.1
LRH	0.22	0.56	7.6
DP	0.41	0.95	7.7
MWM	0.30	0.90	9.4
Greedy	0.29	0.89	13.0

The greedy algorithm has a false alarm rate and a correct detection rate similar to MWM with a higher MSE.

6.3. Natural images

In Fig. 15, we show two real stereo images [128 pixels square] of a terrain with trees, bushes and stumps. We analyze these images using 7×7 windows and normalized correlations as in Eq. (1), with $\delta = 60$ and $C_O = 0.4$.[6] The matching results are shown in Fig. 16 where all disparity maps have been post-processed with a 5×5 median filter to eliminate minor false matches. Local search, shown in Fig. 16(a) shows good performance, but there are several matching artifacts. Most of these errors are eliminated in Fig. 16(b) using the left–right heuristic; however, the coverage is less complete. DP, shown in Fig. 16(c), has very good performance as do MWM in Fig. 16(d), and the greedy algorithm shown in Fig. 16(e). In environments such as this, where narrow occluding objects are rare, the similar performance of the global matching methods favors selection of DP because of its speed advantage.

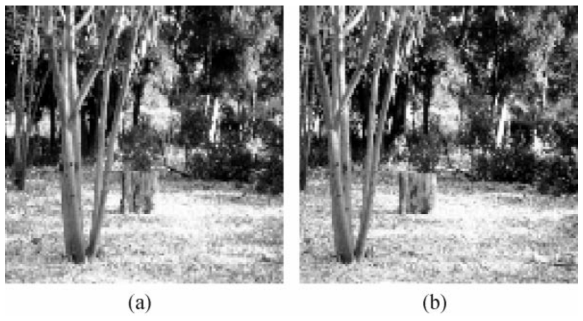


Fig. 15. Real stereo images (a) left, and (b) right [images are 256×256].

7. Conclusions

We have presented two new matching algorithms for solving the correspondence problem in stereo vision. They are based on formal combinatorial optimization techniques for finding matchings on graphs. These algorithms were compared to three existing stereo matching techniques using a set of performance criteria which address correct detections, false alarms in occluded regions, and matching accuracy (mean-squared-error). In general, the global matching schemes—dynamic programming, maximum weighted matching, and the greedy algorithm—show superior performance to local matching methods—local search and the left–right heuristic. Of the global matching approaches, dynamic programming is fastest. As long as the ordering constraint is not violated, dynamic programming is the matching algorithm of choice. In unstructured environments where narrow occluding objects may be present, maximum weighted matching and greedy matching are preferred over dynamic programming.

[6] For DP we use $\hat{C}_O = 0.2$ to elicit "better" performance. Given C_O for any of the matching algorithms except DP requires setting $\hat{C}_O = C_O/2$ to obtain a "comparable" matching. This is because DP can traverse two edges, effectively omitting a match at a given location.

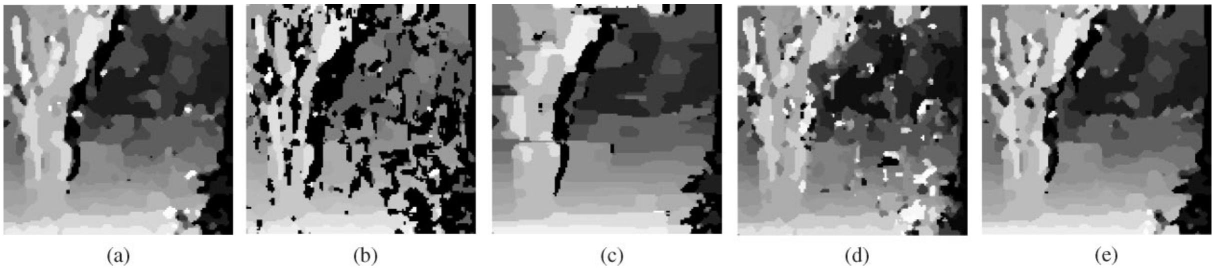


Fig. 16. Disparity maps generated from the stereo images in Fig. 15 using 7×7 windows and the similarity function in Eq. (1) using (a) local search, (b) left–right heuristic, (c) dynamic programming, (d) maximum weighted matching, and (e) greedy algorithm.

Appendix A. Computing the cost of occlusion

In Ref. [27] we describe a method of computing the *cost of occlusion*. This method is an expansion of a previous study by Cox et al. [7]. We use Eq. (2) and compute the occlusion cost based on the desired probability of detection P_D and the sensor noise σ.

Let z_{1,i_1} and z_{2,i_2} be measurements from corresponding epipolar lines in two cameras. If the measurements are matching (i.e., they are projections of the same point/surface in space), we assume that the difference between the measurements is an independent normal random variable $z_{1,i_1} - z_{2,i_2} \sim \mathrm{N}(0, \sigma^2)$. We define $z_{i_1,i_2} = z_{1,i_1} - z_{2,i_2}$ so the random variable, $z^2_{i_1,i_2}$ has a gamma distribution [28],

$$f_x(x) = \frac{c^b}{\Gamma(b)} x^{b-1} \mathrm{e}^{-cx}, \tag{A.1}$$

where $b = \frac{1}{2}$ and $c = 1/2\sigma^2$. Knowing the underlying distribution (A.1) of the matching cost, the probability P_D of obtaining a matching cost between 0 and $\tilde{C}_O$ can be calculated through the integral

$$P_D = \int_0^{\tilde{C}_O} \sqrt{\frac{1}{2\pi\sigma^2}}\, x^{-\frac{1}{2}} \mathrm{e}^{-x/(2\sigma^2)}\, \mathrm{d}x = \gamma\left(\frac{1}{2}, \frac{\tilde{C}_O}{2\sigma^2}\right), \tag{A.2}$$

where γ is the incomplete gamma function defined by

$$\gamma(b, x) = \frac{1}{\Gamma(b)} \int_0^x t^{b-1} \mathrm{e}^{-t}\, \mathrm{d}t.$$

Conversely, we may fix P_D in (A.2) and compute $\tilde{C}_O$ through tabulated percentile values of the cumulative gamma distribution or through numerical evaluation of the integral in Eq. (A.2). In this case, each matching instance may be considered a hypothesis test where we compare the test statistic (the matching cost) to the threshold $\tilde{C}_O$ which is determined by P_D.[7]

[7] The significance of the hypothesis test would be $1 - P_D$.

Let the current epipolar line be j_0. The neighborhood $\Omega_{s,i_0,j_0}(p, q)$ of a feature z_{s,i_0,j_0} in camera s, on epipolar line j_0 and feature at location i_0 is defined as:

$$\Omega_{s,i_0,j_0}(p, q) = [\{z_{s,i_0+i,j_0+j}\}, \text{ s.t. } -p \leqslant i \leqslant p, \\ -q \leqslant j \leqslant q].$$

For the epipolar line j_0, we compare corresponding features in the neighborhoods $\Omega_{1,i_1,j_0}(p, q)$ and $\Omega_{2,i_2,j_0}(p, q)$. The cost of matching two features is defined as the sum of squared differences over the neighborhood (i.e., a $2p + 1$ by $2q + 1$ rectangular window of pixels),

$$g_{j_0,i_1,i_2} = \sum_{i=-q}^{i=q} \sum_{j=-p}^{j=p} (z_{1,i_1+i,j_0+j} - z_{2,i_2+i,j_0+j})^2. \tag{A.3}$$

The local window has $n = (2p + 1)(2q + 1)$ elements and the difference between matching features is assumed to be an independent and identically distributed normal random variable, $\mathrm{N}(0, \sigma^2)$. Each term in the sum is a random variable having a gamma distribution with parameters $b = \frac{1}{2}$ and $c = 1/2\sigma^2$. Hence, the sum in Eq. (A.3) again has a gamma distribution (A.1) with the parameters $b = n/2$ and $c = 1/2\sigma^2$. The probability of detection, P_D and the cost of occlusion $\hat{C}_O$ are related through the incomplete gamma function

$$P_D = \gamma\left(\frac{n}{2}, \frac{\tilde{C}_O}{2\sigma^2}\right). \tag{A.4}$$

If we specify the probability of detection P_D, then we can compute the cost of occlusion from the percentile values of the gamma distribution function. For example, with $n = 9$, $P_D = 0.99$, and $\sigma = 5$ we have $C_O = 541.65 \approx 542$; with $n = 9$, $P_D = 0.99$, and $\sigma = 15$ we have $C_O = 4874.84 \approx 4875$.

References

[1] U.R. Dhond, J.K. Aggarwal, Structure from stereo: a review, IEEE Trans. Systems Man Cybernet 19 (6) (1989) 1489–1510.

[2] L. Matthies, R. Szeliski, T. Kanade, Kalman filter-based algorithms for estimating depth from image sequences, Int. J. Comput. Vision 3 (3) (1989) 209–238.

[3] G. Pajares, J.M. Cruz, J. Aranda, Relaxation by hopfield network in stereo image matching, Pattern Recognition 31 (5) (1998) 561–574.

[4] T. Kanade, M. Okutomi, A stereo matching algorithm with an adaptive window: theory and experiment, IEEE Trans. Pattern Anal. Mach. Intell. 16 (9) (1994) 920–932.

[5] T. Kanade, Development of a video rate stereo machine, Proceedings of the ARPA Image Understanding Workshop, November 1994, pp. 549–558.

[6] B. Ross, A practical stereo vision system, Proceedings of the IEEE Computer Vision and Pattern Recognition, 1993, pp. 148–153.

[7] I. Cox, S. Hingorani, S. Rao, B. Maggs, A maximum likelihood stereo algorithm, CVGIP: Image Understanding 63 (3) (1996) 542–567.

[8] E. Lawler, Combinatorial Optimization, Holt, Rinehart and Winston, New York, NY, 1976.

[9] G. Fielding, M. Kam, Applying the hungarian method to stereo matching, Proceedings of the IEEE Conference on Decision and Control, December 1997, pp. 549–558.

[10] B.K.P. Horn, Robot Vision, MIT Press, Cambridge, MA, 1983.

[11] D. Ballard, C. Brown, Computer Vision, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[12] A. Watt, 3D Computer Graphics, Addison-Wesley Publishing Co., New York, NY, 1993.

[13] R.Y. Tsai, A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses, IEEE J. Robot. Automat. RA-3 (4) (1987) 323–344.

[14] Guo-Qing Wei, Song De Ma, Implicit and explicit camera calibration: theory and experiments, IEEE Trans. Pattern Anal. Mach. Intell. 16 (5) (1994) 469–480.

[15] M. Drumheller, T.A. Poggio, On parallel stereo, International Conference on Robotics and Automation, 1986, pp. 1439–1448.

[16] M.J. Hannah, Sri's baseline stereo system, Proceedings of the DARPA Image Understanding Workshop, 1985, pp. 149–155.

[17] D.G. Jones, J. Malik, Computational framework for determining stereo correspondence from a set of linear spatial filters, European Conference on Computer Vision, 1992, pp. 395–410.

[18] H.H. Baker, T.O. Binford, Depth from edge and intensity based stereo, International Joint Conference on Artificial Intelligence, 1981, pp. 631–636.

[19] E. Krotkov, M. Hebert, R. Simmons, Stereo perception and dead reckoning for a prototype lunar rover, Autonomous Robots 2 (4) (1995) 313–331.

[20] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.

[21] Y. Ohta, T. Kanade, Stereo by intra- and inter-scanline search using dynamic programming, IEEE Trans. Pattern Anal. Mach. Intell. 7 (2) (1985) 139–154.

[22] T.A. Poggio, V. Torre, C. Koch, Computational vision and regularization theory, Nature 317 (1985) 314–319.

[23] R. Tarjan, Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.

[24] C. Papadimitriou, K. Steiglitz, Combinatorial Optimization, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[25] K. Mehlhorn, S. Naher, Leda: a Platform for Combinatorial and Geometric Computing, Cambridge University Press, Cambridge, 1999.

[26] B. Jahne, Digital Image Processing, 2nd Edition, Springer, New York, NY, 1993.

[27] G. Fielding, M. Kam, Computing the cost of occlusion, Comput. Vision and Image Understanding, 1998, submitted for publication.

[28] A. Papoulis, Probability, Random Variables, and Stochastic Processes, 3rd Edition, McGraw-Hill Book Company, New York, NY, 1991.

About the Author—GABRIEL FIELDING received the M.S. (Electrical Engineering) and Ph.D. degrees from Drexel University, in 1996 and 1999, respectively. From 1995–1998, he was a National Science and Defense Graduate Research Fellow sponsored by the Office of Naval Research. He has worked on computer vision projects with the Advanced Robotics Group at the Naval Research Labs in San Diego, CA. Currently, he is a research scientist with the Eastman Kodak Company in Rochester, New York. His research interests are motion estimation and parallel algorithms for image processing.

About the Author—MOSHE KAM was educated at Tel Aviv University (B.S. 1977) and Drexel University (M.Sc 1985, Ph.D. 1987). At present he is a Professor of Electrical and Computer Engineering at Drexel, and Director of its Data Fusion Laboratory. He is a recipient of an NSF Presidential Young Investigator Award (1990), the C.H. MacDonald award for the Outstanding Young Electrical Engineering Educator (1991), and the Drexel University Research Award (1998). His research interests are in System Theory, Detection and Estimation (especially distributed detection and decision fusion,) Robotics, Navigation, and Control.