
International Journal of Computer Vision 35(3), 269–293 (1999)


# Depth Discontinuities by Pixel-to-Pixel Stereo

STAN BIRCHFIELD AND CARLO TOMASI
*Department of Computer Science, Stanford University, Stanford, CA 94305*
birchfield@cs.stanford.edu
tomasi@cs.stanford.edu

*Received June 17, 1997; Revised August 2, 1999*

**Abstract.** An algorithm to detect depth discontinuities from a stereo pair of images is presented. The algorithm matches individual pixels in corresponding scanline pairs, while allowing occluded pixels to remain unmatched, then propagates the information between scanlines by means of a fast postprocessor. The algorithm handles large untextured regions, uses a measure of pixel dissimilarity that is insensitive to image sampling, and prunes bad search nodes to increase the speed of dynamic programming. The computation is relatively fast, taking about 600 nanoseconds per pixel per disparity on a personal computer. Approximate disparity maps and precise depth discontinuities (along both horizontal and vertical boundaries) are shown for several stereo image pairs containing textured, untextured, fronto-parallel, and slanted objects in indoor and outdoor scenes.

**Keywords:** stereo matching, depth discontinuities, dynamic programming, untextured scenes, image sampling

## 1. Introduction

Cartoon artists have known the perceptual importance of depth discontinuities for a long time. To create the illusion of depth, they paint characters and background on different layers of acetate, being careful to ensure a crisp delineation of foreground objects. Similarly, in human stereo vision, depth discontinuities are vividly perceived and help to carve out distinct objects as well as to elucidate the distance relations between them.

Depth discontinuities play a fundamental role in image understanding, as acknowledged by the insightful "layers" representation of image sequences (Wang and Adelson, 1994). For data-reduction, depth discontinuities are similar to intensity edges in preserving the "interesting" aspects of an image with much less data (Attneave, 1954; Malik and Perona, 1990), and they serve a purpose similar to segmentation because they tend to carve out the distinct objects in a scene (Chen and Lin, 1997; Gamble, 1989; Little and Gillett, 1990; Poggio et al., 1988). Depth discontinuities are more powerful than intensity edges, however, because they truly tend to outline the contours of objects rather than changes in pigmentation or illumination, and unlike the elusive problem of segmentation, depth discontinuities are well-defined.

In this paper we present a method for detecting depth discontinuities from a stereo pair of images that first computes a dense disparity map and then labels those points that exhibit a significant change in disparity. (A threshold is unfortunately inherent in any problem of estimating discontinuities from a discrete function.)

Many traditional stereo algorithms, such as those based on correlation windows, tend to blur depth discontinuities. Our algorithm, on the other hand, uses neither windows nor preprocessing of the intensities, and matches the individual pixels in one image with those in the other image. As a result, we preserve sharp changes in disparity while introducing few false discontinuities, with far less computation than would be required if disparities were computed to subpixel resolution (which of course would be necessary for the more common goal of accurate scene reconstruction). Thus, by sacrificing accurate disparities our

algorithm is able to quickly compute crisp and accurate discontinuities.

Like several previous algorithms (Belhumeur and Mumford, 1992; Cox et al., 1996; Geiger et al., 1995; Intille and Bobick, 1994), our approach first matches different epipolar scanline pairs independently, and detects occlusions simultaneously with a disparity map, using a form of dynamic programming. Then, a postprocessor propagates information between the scanlines to refine the disparity map, from which the depth discontinuities are detected.

Our approach contains four novelties: (1) a method for handling large untextured regions which present a challenge to many existing stereo algorithms; (2) a measure of pixel dissimilarity that overcomes the image sampling problem; (3) a technique to reduce dramatically the running time of dynamic programming by pruning unlikely search nodes; and (4) a postprocessor that quickly propagates disparities between scanlines to produce a cleaner disparity map. The combination of avoiding subpixel resolution, pruning bad nodes, and fast postprocessing results in an efficient algorithm that takes 600 nanoseconds per pixel per disparity on a personal computer, making it a candidate for real-time implementation. We demonstrate the algorithm's performance in a wide range of scenarios: indoor and outdoor scenes, textured and untextured objects, textured and untextured backgrounds, curved and planar surfaces, specular and matte surfaces, and fronto-parallel and slanted surfaces.

The paper is organized as follows. After briefly reviewing closely related work in Section 3, we describe our algorithm for matching two scanlines independently in Section 3. The following three sections then explain in more detail the three novelties of the matching process: a method for handling untextured regions, a pixel dissimilarity measure that is insensitive to sampling, and a pruning strategy for improving the speed over standard dynamic programming. The discussion of our algorithm ends in Section 7 with a presentation of the postprocessor. Sections 8 and 9 contain, respectively, experimental results on several pairs of images and a thorough demonstration of the importance of our dissimilarity measure. The final section contains some of our conclusions.

## 2. Previous Work

Some researchers have explored the possibility of detecting depth discontinuities from stereo images directly, by purely local means. Little and Gillett (1990) propose two algorithms, the first of which matches pixels in the two images while keeping track of the pixels that lie in the forbidden zone[1] of at least one match, for all possible disparities. From these pixels, which are basically the occluded pixels, the depth discontinuities are inferred. In the second algorithm, a search is conducted for each pixel in one image to find a corresponding match in the other image, and a depth discontinuity is declared if there is more than one good match. This second approach is similar to the mixture-motion methods aimed at detecting motion discontinuities by identifying small regions that contain more than a single motion (Black and Anandan, 1990; Spoerri and Ullman, 1987).

Toh and Forrest (1990) describe a local method in which the stereo cameras are assumed to be fixated on an intensity edge roughly perpendicular to the baseline: A depth discontinuity is declared if either the left or the right sides of the boundary do not match. This work was extended by Wixson (1993), whose algorithm detects and links near-vertical edges in untextured images, then matches the edges from the two images. Once correspondence is established, the left and right regions of the edges are examined, and if one of them does not match well, then a depth discontinuity is declared. Since the region size depends on the amount of texture in the region, Wixson's extension is more global than the other methods.

There are two main limitations with these techniques. First, the local methods require texture throughout the images because depth discontinuities cannot be detected locally in the absence of texture. Secondly, the identification of occlusions with depth discontinuities is not correct. Algorithms which make this connection are not able to detect depth discontinuities that lie along boundaries parallel to the baseline, because these discontinuities do not give rise to occlusion. Horizontal boundaries abound in man-made scenes and cannot be ignored.

As mentioned before, another promising approach is to first compute a dense disparity map, and then to detect the sharp changes in disparity. With the exception of the cooperative algorithm of Marr and Poggio (1976) which was tested only on random-dot stereograms, early stereo algorithms (Baker and Binford, 1981; Ohta and Kanade, 1985; Grimson, 1985) tended to smooth the disparities across the depth discontinuities, mainly due to their reliance upon simple interpolation schemes that did not allow for sharp changes in disparity. Unfortunately, the popular technique of window-based correlation has the same problem, and the solution of

iteratively reducing the window size based on the amount of disparity variation within the window is computationally expensive (Jones and Malik, 1992; Kanade and Okutomi, 1994).

More recent stereo algorithms incorporate the phenomena of occlusions and depth discontinuities at an early stage and are therefore capable, with varying degrees of success, of preserving sharp changes in disparity (Belhumeur and Mumford, 1992; Cox et al., 1996; Fua, 1991; Geiger et al., 1995; Intille and Bobick, 1994; Jones and Malik, 1992; Luo and Burkhardt, 1995). Our approach builds on this line of work, resulting in crisper, more accurate depth discontinuities on a wider range of images, and with much less computation.

## 3. Matching Scanlines

We pose the stereo correspondence problem as the problem of matching pixels in one scanline to pixels in the corresponding scanline in the other image, assuming that the two cameras are rectified.[2] We have found that pixel resolution is sufficient to compute a rough disparity map and to detect the most prominent depth discontinuities.

Correspondence is encoded in a *match sequence*. Each *match* is an ordered pair $(x, y)$ of pixels signifying that the intensities $I_L(x)$ and $I_R(y)$ are images of the same scene point. (Throughout this paper, $x$ denotes a pixel in the left scanline while $y$ denotes a pixel in the right scanline.) Unmatched pixels are *occluded*, and a subsequence of adjacent occluded pixels that is bordered by two non-occluded pixels (or by a non-occluded pixel and the image boundary) is called an *occlusion*. (Our *occlusion*s roughly correspond to the *half-occluded region*s of Belhumeur and Mumford (1992).) An example of a match sequence on an extremely short scanline is shown in Fig. 1.

The *disparity* $\delta(x)$ of a pixel $x$ in the left scanline that matches some pixel $y$ in the right scanline is defined in the usual way as $x - y$, while the disparities of all the pixels in an occlusion are assigned the disparity of the farther of the two neighboring matches. The *depth-discontinuity pixels* are considered for now to be those pixels that border a change of at least one disparity level and lie on the far object. For postprocessing in Section 7 and for displaying the final results in Section 8, we use a threshold of two disparity levels, which enables the algorithm to handle slanted objects without explicitly detecting the slant.

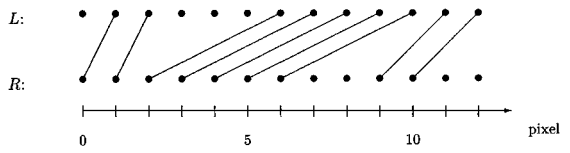


*Figure 1.* The match sequence $M = \langle(1,0), (2,1), (6,2), (7,3), (8,4), (9,5), (10,6), (11,9), (12,10)\rangle$. The five middle matches correspond to a near object.

### 3.1. Cost Function

With each match sequence $M$ we associate a cost $\gamma(M)$ that measures how unlikely it is that $M$ describes the true correspondence. (That is, the best match sequence has the lowest cost.) Instead of deriving a maximum a posteriori or maximum likelihood cost function from a Bayesian formulation (Belhumeur and Mumford, 1992; Cox et al., 1996; Geiger et al., 1995; Luo and Burkhardt, 1995), we propose a simple cost function which is based mainly on intuition and justified solely by empirical evidence.

The cost of a match sequence is defined by a constant penalty for each occlusion, a constant reward for each match, and a sum of the dissimilarities between the matched pixels:

$$\gamma(M) = N_{\text{occ}}\kappa_{\text{occ}} - N_{\text{m}}\kappa_{\text{r}} + \sum_{i=1}^{N_{\text{m}}} d(x_i, y_i), \quad (1)$$

where $\kappa_{\text{occ}}$ is the constant occlusion penalty, $\kappa_{\text{r}}$ is the constant match reward, $d(x_i, y_i)$ is the dissimilarity between pixels $x_i$ and $y_i$, and $N_{\text{occ}}$ and $N_{\text{m}}$ are the number of occlusions (not the number of occluded pixels) and matches, respectively, in $M$.

Because each change in disparity incurs an occlusion penalty, this cost function prefers piecewise-constant disparity maps. Thus, if possible, each object is assigned a single disparity, even if that object's depth actually varies (as in the case of a cylindrical surface). Although this behavior sacrifices accurate scene reconstruction, it facilitates the precise localization of depth discontinuities because it accentuates the change in disparity at the boundaries of those objects, like cylinders, whose disparity tapers at the ends. In addition, the simplicity of Eq. (1) makes our cost function easy to understand, implement, and evaluate.

***3.1.1. Occlusion Penalty and Match Reward.*** Intuitively, $\kappa_{\text{occ}}$ is interpreted as the amount of evidence (in terms of mismatched pixel intensities) that is necessary in order to declare a change in disparity, while $\kappa_{\text{r}}$

is interpreted as the maximum amount of pixel dissimilarity that is generally expected between two matching pixels. Together, the two terms function identically to an occlusion penalty that is linearly dependent on the length of the occlusion (Belhumeur and Mumford, 1992; Geiger et al., 1995). More precisely, let $l_i$ be the length (in terms of the number of pixels) of occlusion $i$. Then

$$\min\ \gamma(M) = \min\left\{ \sum_{i=1}^{N_{\text{occ}}} (\kappa_{\text{occ}} + \kappa_{\text{r}} l_i) + \sum_{i=1}^{N_{\text{m}}} d(x_i, y_i) \right\},$$

which is easily shown by noting that the number of matched pixels plus the number of occluded pixels equals the total number $n$ of pixels in the scanline, i.e.,

$$N_{\text{m}} + \sum_{i=1}^{N_{\text{occ}}} l_i = n.$$

We choose the former formulation because a constant occlusion penalty is essential to our method of pruning the search space, as described in Section 6.

***3.1.2. Pixel Dissimilarity.*** For now we define the dissimilarity simply as the absolute difference of intensity between the two pixels:

$$d(x_i, y_i) = |I_{\text{L}}(x_i) - I_{\text{R}}(y_i)|. \tag{2}$$

Because this definition is adversely affected by image sampling, as we will see in Section 5, we will later replace it with a more sophisticated measure of dissimilarity.

### *3.2. Hard Constraints*

In addition to measuring the likelihood of a match sequence by its cost, we impose hard constraints upon each match sequence. These constraints serve two purposes: (1) they facilitate a systematic search of the space of possible match sequences, and (2) they disallow certain types of unlikely match sequences. The first five constraints are as follows:

C1. $0 \le x_i - y_i \le \Delta, \quad i = 1, \ldots, N_{\text{m}}$
C2. $y_1 = 0$
C3. $x_{N_{\text{m}}} = n - 1$
C4. $x_i < x_j, \quad$ and
$y_i < y_j, \ 1 \le i < j \le N_{\text{m}}$
C5. $x_{i+1} = x_i + 1, \quad$ or
$y_{i+1} = y_i + 1, \quad i = 1, \ldots, N_{\text{m}} - 1$

To greatly reduce the size of the search space, C1 introduces $\Delta$ as the maximum allowed disparity and requires the disparities to be nonnegative because the cameras are rectified. The following two constraints force the left-most pixel in the right scanline and the right-most pixel in the left scanline ($n$ is the number of pixels in the scanline) to be matched to some pixel in the other scanline—notice that we do not specify which pixels they must match, as some other algorithms do. C4 is a combination of the uniqueness and ordering constraints (Belhumeur and Mumford, 1992; Faugeras, 1993), introduced to simplify dynamic programming, and C5 precludes simultaneous occlusions in the two images.[3]

In the next section we will add two additional constraints that allow the algorithm to handle untextured regions. Two stereo pairs, the left images of which are shown in Fig. 2, are used as running examples in the next few sections. As more constraints and processing

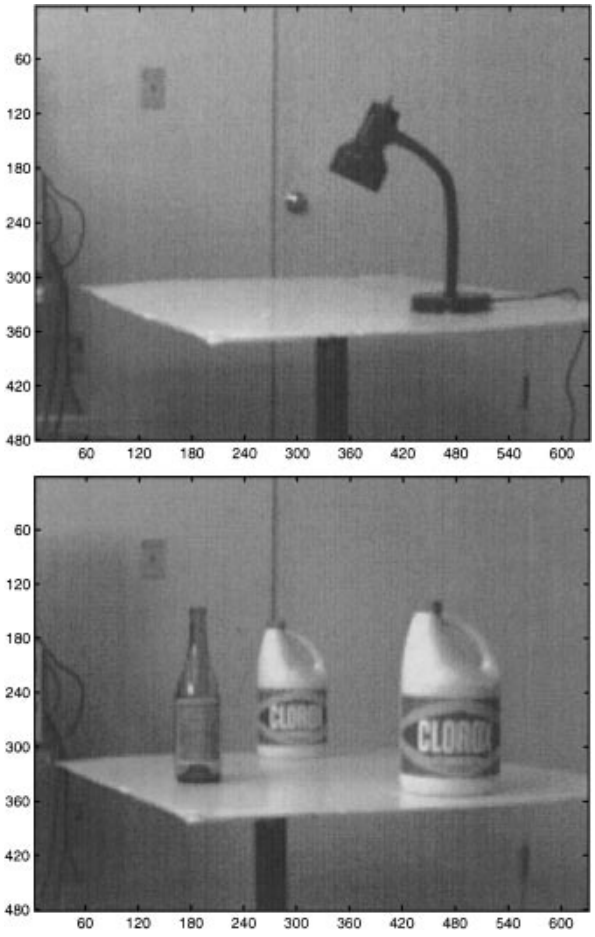


*Figure 2.* The lamp image and the Clorox image.

are introduced, the quality of the resulting depth maps is shown to improve.

## 4. Handling Untextured Regions

Because of the ambiguity inherent in untextured regions, many stereo algorithms require texture throughout the images. In the literature, scenes are often altered artificially by placing a textured background behind the objects of interest in order to make the scene more amenable to the stereo algorithm being tested. As we will demonstrate, however, untextured regions that are approximately fronto-parallel can be handled quite nicely as long as one assumption remains true, namely that intensity variation accompanies depth discontinuities.

Figures 3–5 show the depth maps for the two stereo pairs in Fig. 2 obtained by adding different constraints on the allowed location of depth discontinuities. In Fig. 3, no constraint is added to constraints C1–C5 discussed in Section 3.2. Since there is no texture in the background, noise determines the location of depth discontinuities.

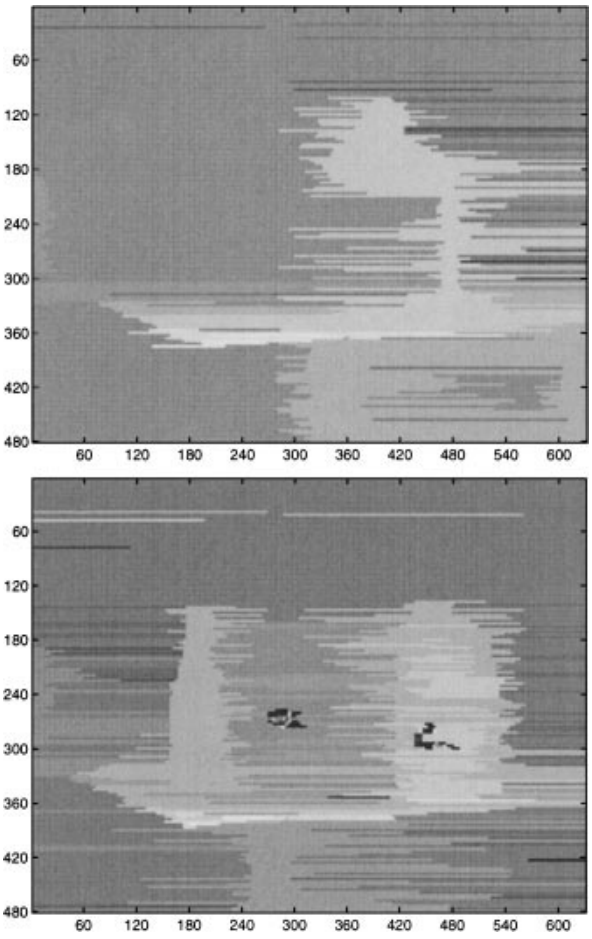


*Figure 3.* The disparity maps obtained without the intensity-variation constraints (and without our dissimilarity measure).

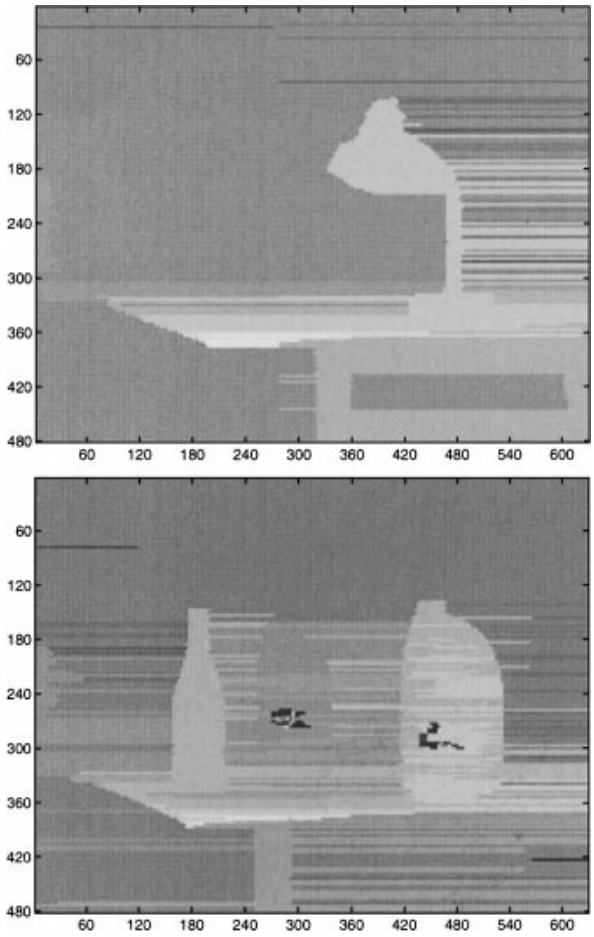


*Figure 4.* The disparity maps obtained with naive intensity-variation constraints (but without our dissimilarity measure).

Figure 4 includes the constraint that no disparity edge can be placed where the derivative of image intensity in the $x$ direction is below a rather low threshold. This threshold was kept fixed at 3 gray levels per pixel in all our experiments. Image locations with intensity changes above this threshold are declared to contain *intensity variations*. These are not edges, both because of the very modest value of the threshold and because edges are usually local maxima of intensity variation. Thus, the constraint used in Fig. 4 does not force depth discontinuities to lie along intensity edges (as done for instance in (Cochran and Medioni, 1992; Fua, 1991; Gamble, 1987)), but merely prevents the algorithm from placing them where any localization decision would inevitably be based on noise.

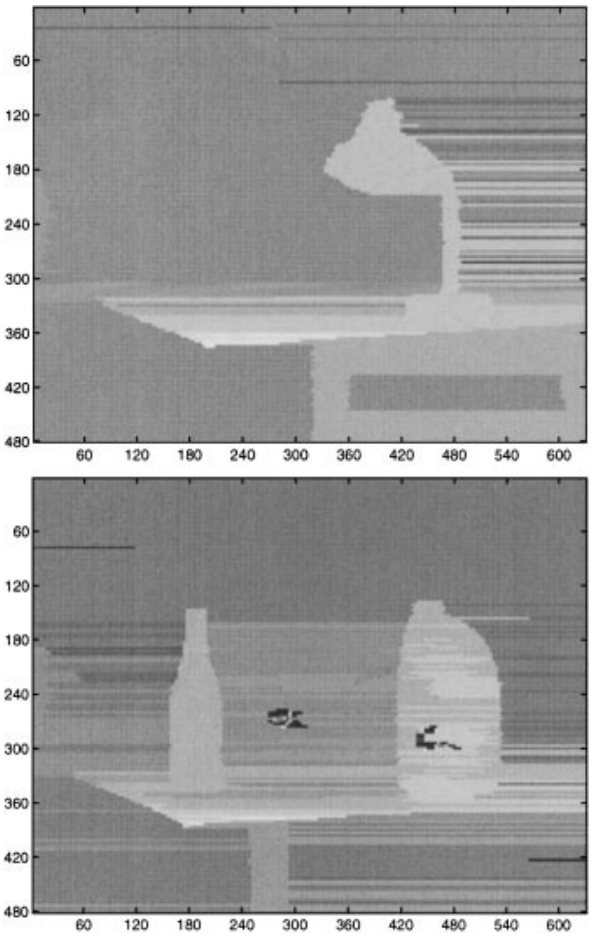


*Figure 5*. The disparity maps obtained with constraints C6 and C7 (but without our dissimilarity measure).

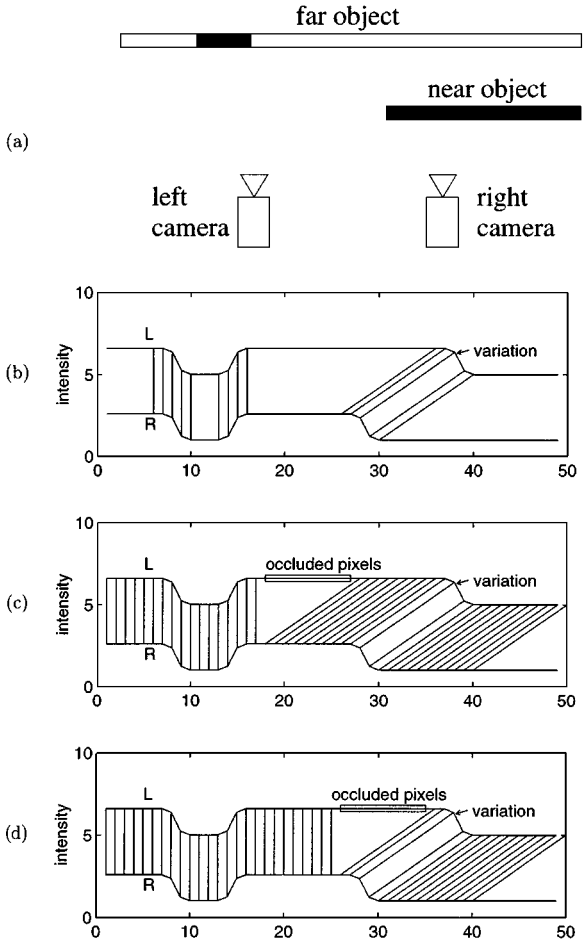


*Figure 6*. Given the assumption that there is a change in intensity along the boundary between the near and far objects, intensity variation must lie to the right of an occlusion in the left scanline. (a) A physical setup; (b) The resulting intensity functions, with the only matches evident from the data—the other matches must be hypothesized; (c) A match sequence that appears feasible but actually violates the assumption; (d) The match sequence that is consistent with the assumption.

Figure 5 further refines the intensity variation constraint by reasoning *on which side* of the intensity variation the depth discontinuity must lie. Specifically, it includes the following two constraints:

C6. If $\langle x_i, \ldots, x_j \rangle$ is an occlusion in the left scanline, then $x_j$ lies to the left of intensity variation, $1 \leq i \leq j < N_{\mathrm{m}}$

C7. If $\langle y_i, \ldots, y_j \rangle$ is an occlusion in the right scanline, then $y_i$ lies to the right of intensity variation, $1 < i \leq j \leq N_{\mathrm{m}}$.

To see the justification for these new constraints, note that intensity variation occuring at a depth discontinuity (as the result of an intensity difference between the near object and the far object) has the same disparity as the near object (see Fig. 6). This arises because the physical origin of the intensity variation is the boundary of the near object, regardless of the geometry of the far object. Therefore, as the camera moves laterally, the intensity variation moves with the projection of the near object.

Using Fig. 6 as an example, we notice that pixels in the left scanline are occluded when the far object's projection is to the left of the near object's. Since the occluded pixels come from the far object, and since the intensity variation is part of the near object, the occlusion must lie immediately to the left of the intensity variation. Likewise, occluded pixels in the right

scanline must lie immediately to the right of an intensity variation.

Adding these constraints to the algorithm yields the results in Fig. 5. The errors on the left edge of the table support have been fixed, as well as the region between the wine and Clorox bottles. The improvement from Fig. 4 to Fig. 5 is modest. However, this is mainly because dynamic programming, together with the other constraints, already does a relatively good job, considering that scanlines are processed individually so far. Different underlying algorithms can perform very differently with and without the considerations on the sidedness of depth discontinuities. As an illustration, Fig. 7 shows the results of a contrived algorithm in which constraints C1–C5 and the intensity variation constraint are satisfied, but not the "sided" version C6, C7. Specifically, the constraints used for Figs. 4 and 7 are exactly the same, except that for Fig. 7 constraints C6 and C7 are reversed to force the wrong sidedness.

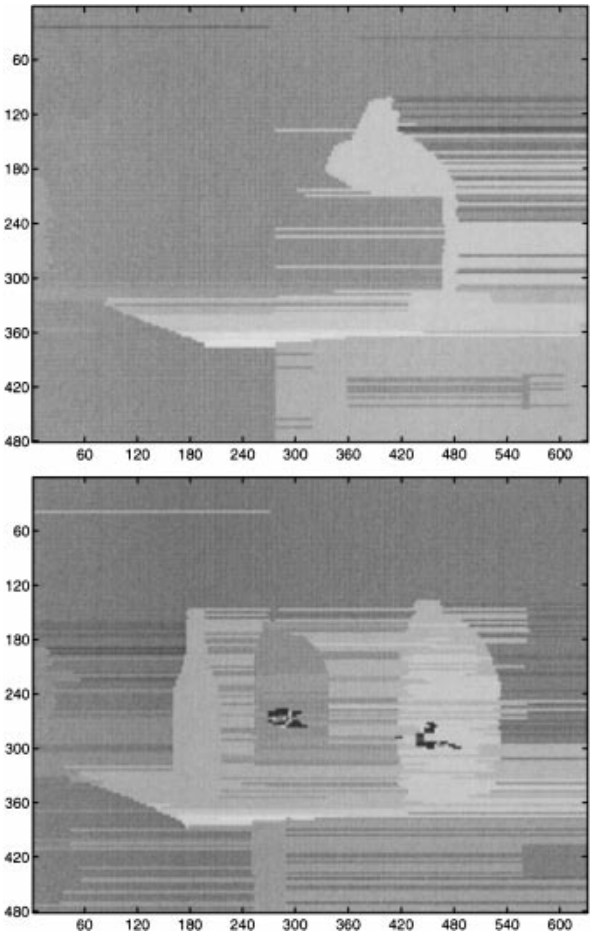


*Figure 7.* Hypothetical disparity maps that satisfy a naive intensity-variation constraint.

In summary, comparison of Figs. 3 and 4 shows that the intensity variation constraint is crucial. Comparison of Figs. 7 and 5 shows that in principle sidedness can be very important as well, while comparison of Figs. 4 and 5 shows that, given our other constraints and the dynamic programming method we use, sidedness is moderately useful.

## 5. The Problem of Sampling

The term $d(x_i, y_i)$ measures how unlikely it is that $I_{\mathrm{L}}(x_i)$ and $I_{\mathrm{R}}(y_i)$ are images of the same scene point. For the experiments of the previous section, $d(x_i, y_i)$ was simply the absolute difference between the two intensities (Eq. (2)). However, this measure is inadequate for discrete images, because image sampling can cause this difference to be large wherever the intensity is changing rapidly and the disparity is not an integral number of pixels. For example, the dark splotches in Fig. 5 arise because the repetitive texture of the lettering on the Clorox bottles is locally ambiguous, and the false disparity is closer to an integral number of pixels than is the true disparity. Typically, the sampling problem is alleviated by working at subpixel resolution (Belhumeur and Mumford, 1992; Luo and Burkhardt, 1995), but this solution is computationally expensive for algorithms that explicitly search over all possible disparities. Therefore, we propose instead to use the linearly interpolated intensity functions surrounding two pixels to measure their dissimilarity, in a method that is provably insensitive to sampling. This technique increases the computing time by only about 10%, as opposed to subpixel resolution which can increase the time by as much as 1100% (Birchfield and Tomasi, 1998b).

To understand our dissimilarity measure in more detail, consult Fig. 8, which shows the intensity functions $I_{\mathrm{L}}$ and $I_{\mathrm{R}}$ incident upon two corresponding scanlines of the left and right cameras, respectively. The functions are sampled at discrete points by the image sensor; three such adjacent points (or pixels) are shown here in each scanline. In this discussion, $x_i$ and $y_i$ are the pixels whose dissimilarity is to be measured. We define $\hat{I}_{\mathrm{R}}$ as the linearly interpolated function between the sample points of the right scanline. Then we try to measure how well the intensity at $x_i$ fits into the linearly interpolated region surrounding $y_i$. That is, we

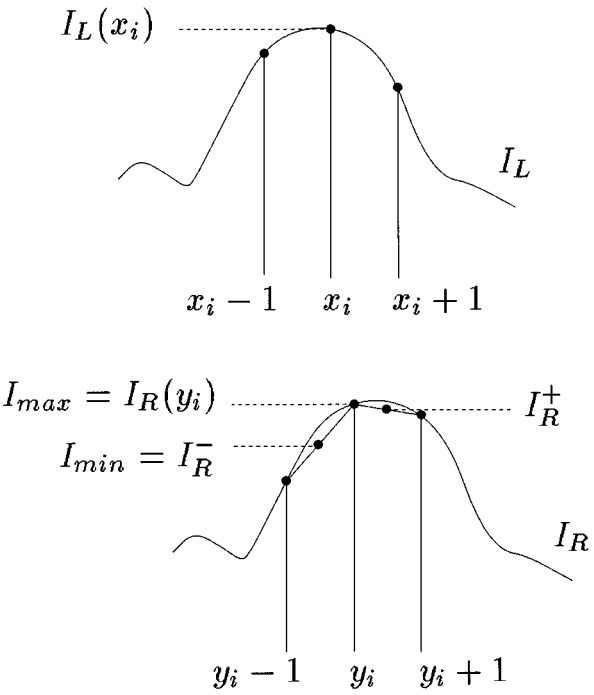


*Figure 8.* Definition and computation of $\bar{d}(x_i, y_i, I_L, I_R)$.

define the following quantity:

$$\bar{d}(x_i, y_i, I_L, I_R) = \min_{y_i - \frac{1}{2} \le y \le y_i + \frac{1}{2}} |I_L(x_i) - \hat{I}_R(y)|.$$

Then, the dissimilarity between the pixels is computed as the minimum of this quantity and its symmetric counterpart:

$$d(x_i, y_i) = \min\{\bar{d}(x_i, y_i, I_L, I_R), \bar{d}(y_i, x_i, I_R, I_L)\}.$$

Thus, the definition of $d$ is symmetrical.

Since the extreme points of a piecewise linear function must be its breakpoints, the computation of $d$ is rather straightforward. Again, see Fig. 8. First we compute $I_R^- \equiv \hat{I}_R(y_i - \frac{1}{2}) = \frac{1}{2}(I_R(y_i) + I_R(y_i - 1))$, the linearly interpolated intensity halfway between $y_i$ and its neighboring pixel to the left, and the analogous quantity $I_R^+ \equiv \hat{I}_R(y_i + \frac{1}{2}) = \frac{1}{2}(I_R(y_i) + I_R(y_i + 1))$. Then we let $I_{\min} = \min(I_R^-, I_R^+, I_R(y_i))$ and $I_{\max} = \max(I_R^-, I_R^+, I_R(y_i))$. With these quantities defined,

这个通过插值来计算代价有点意思

$$\bar{d}(x_i, y_i, I_L, I_R) = \max\{0, I_L(x_i) - I_{\max}, I_{\min} - I_L(x_i)\}.$$

This computation takes only a small, constant amount of time more than the absolute difference in intensities.

The quantity $d$ is insensitive to sampling in the sense that, without noise or other distortions, $d(x_i, y_i) = 0$ whenever $y_i$ is the closest sampling point to the $y$ value corresponding to $x_i$. The only restriction is that the continuous intensity function incident upon the sensor be either concave or convex in the vicinity of $x_i$ and $y_i$. In practice, inflection points cause no problem since the regions surrounding them are approximately linear—and linear functions are both concave and convex. Therefore, our cost function works well as long as the intensity function varies slowly compared to the pixel spacing on the sensor, i.e., as long as aliasing does not occur (see Birchfield and Tomasi, 1996, 1998b, for details). We slightly defocus the lens to ensure this condition.

Figure 9 contrasts our dissimilarity measure with the absolute difference in intensity, on portions of two

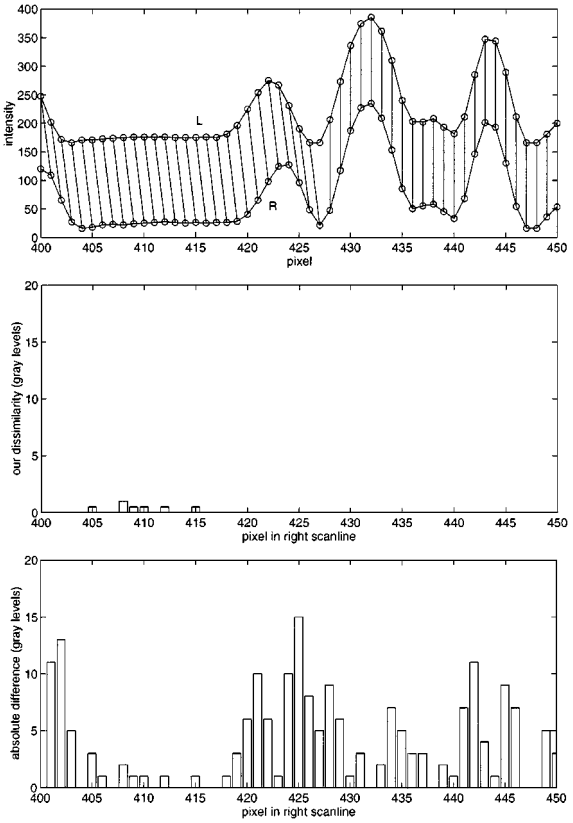


*Figure 9.* TOP: A portion of a match sequence. For viewing clarity, the left scanline is shifted up, and the right scanline is shifted to the right. MIDDLE: The dissimilarities between the matched pixels, as computed by our measure (Most of the values are zero). BOTTOM: The dissimilarities computed by taking the absolute value of the difference in intensity.

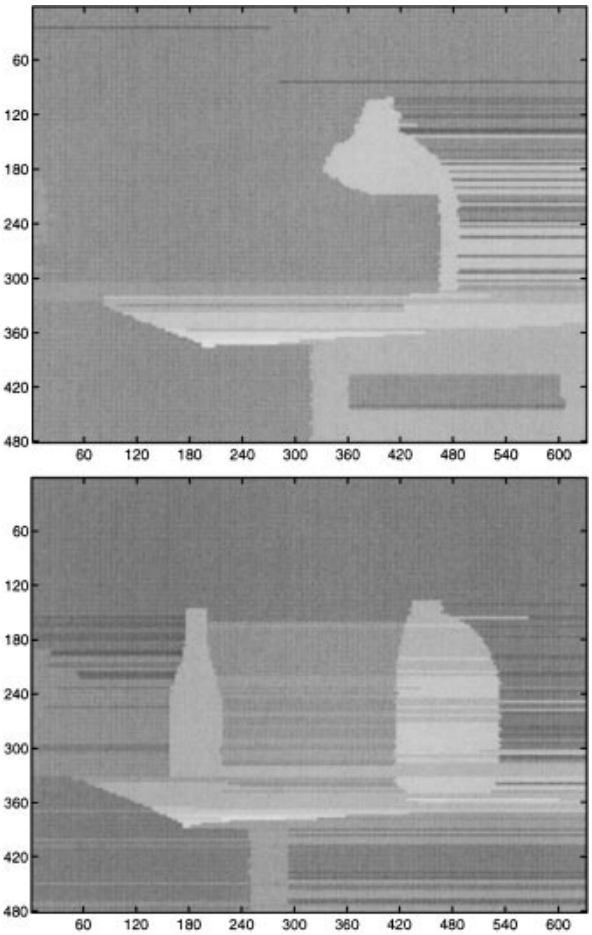


*Figure 10.* The disparity maps obtained with our dissimilarity measure (and with the constraints C6 and C7).

real scanlines. Wherever the intensity function is nearly constant, or wherever the disparity between the two scanlines is close to an integral number of pixels, the two approaches yield similar results, since sampling effects are negligible. In the remaining areas, however, the absolute difference can be large, while our measure remains well-behaved.

The disparity maps obtained by using our dissimilarity measure are shown in Fig. 10. Notice that the dark splotches have disappeared. Although these disparity maps are greatly improved, the streaks and the concavity of the lamp highlight an additional difficulty caused by untextured regions: disparity may not be discernible from a single scanline. Our postprocessing method described in Section 7 has been designed to solve this problem.

## 6. The Matching Algorithm

Thanks to the structure of the cost function in Eq. (1), the technique of dynamic programming (also used in (Baker and Binford, 1981; Belhumeur and Mumford, 1992; Cox et al., 1996; Geiger et al., 1995; Intille and Bobick, 1994; Ohta and Kanade, 1985)) can be used to find the optimal match sequence.

Figure 11(a) illustrates the search grid for two scanlines having 10 pixels each, using a maximum disparity of three pixels (i.e., $\Delta = 3$). Cell $(x, y)$ represents a possible match between pixel $x$ in the left image and pixel $y$ in the right image. Because of the disparity limit, many of the cells in the grid are disallowed; these are shown as black cells. Our algorithm searches for the best possible path[4] stretching from the first column to the last row. As an example, the match sequence $\langle(1, 0), (2, 1), (3, 2), (5, 3), (6, 4), (7, 5), (8, 7), (9, 8)\rangle$ is shown by the cells marked with $\times$. Any column or row that does not contain an $\times$ corresponds to an occluded pixel.

For any match $(x_i, y_i)$, the matches which can possibly be chosen as its immediately preceding match $(x_{i-1}, y_{i-1})$ are the cells shown in Fig. 11(b), due to constraints C4 and C5. Similarly, the matches which can possibly be chosen as its immediately following match $(x_{i+1}, y_{i+1})$ are the cells shown in Fig. 11(c). Placing Figs. 11(b) and (c) onto the grid of Fig. 11(a) reveals that each match has $\Delta + 1$ possible candidates as its immediately preceding match and $\Delta + 1$ possible candidates as its immediately following match (unless it is near one of the image boundaries). (Recall that $\Delta$ is the maximum disparity, from Constraint C1.)

In order to make the diagram in Fig. 11(a) more compact, we shift each column up by an amount equal to the number of the column, which leads to the grid in Fig. 12(a). The vertical axis is now $\delta = x - y$, the disparity. In a similar manner, Figs. 11(b) and (c) become Figs. 12(b) and (c). For each cell of the shifted search grid, we record two pieces of information: $\varphi[\delta, y]$ is the cost of the best match sequence (so far) ending at match $(y+\delta, y)$, and $\pi[\delta, y]$ points to the immediately preceding match in that match sequence.

### 6.1. *Two Dual Optimal Algorithms*

Because of the duality between a match's preceding matches and its following matches, there are two dual algorithms for searching this space. The first algorithm

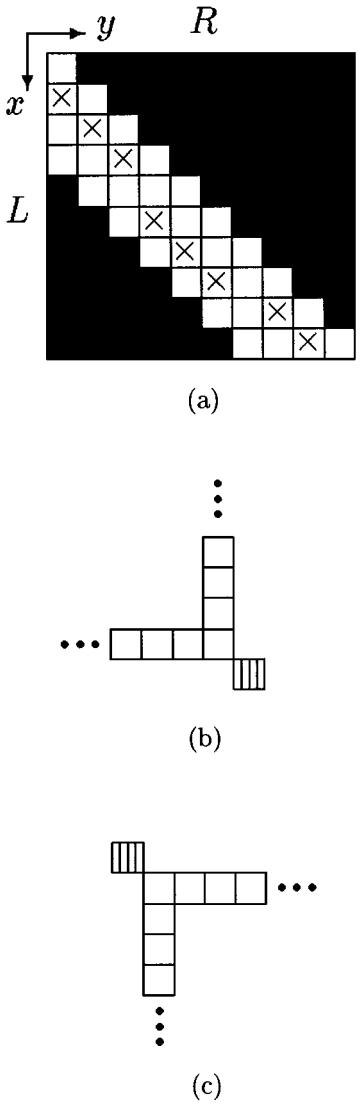


*Figure 11.* (a) The search grid and a match sequence ("×" cells); (b) The matches (white cells) that can immediately precede a match (striped cell); (c) The matches that can immediately follow a match.

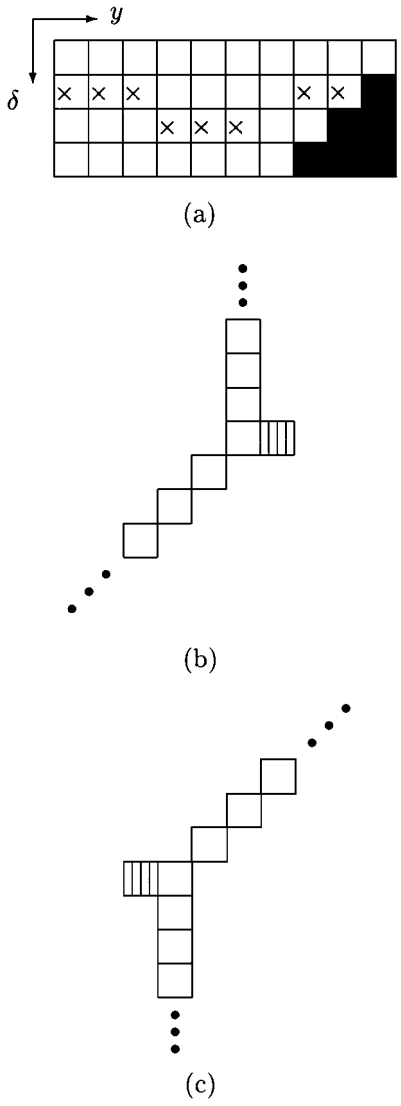


*Figure 12.* (a) The shifted search grid; (b) The immediately preceding matches (ignoring disparity bounds); (c) The immediately following matches.

is more intuitive and straightforward, but the second one provides us with a framework to speed the search by pruning bad cells, as we will see in Section 6.2. Letting $\gamma_0(x, y)$ refer to the cost of the best match sequence whose ending match is $(x, y)$, both algorithms traverse the $\varphi$ array from left to right, and from top to bottom, computing the cost of the best path to each cell:

$$\begin{aligned}\varphi[\delta, y] &= \gamma_0(y + \delta, y) \\ &= d(y + \delta, y) - \kappa_{\mathrm{r}} \\ &\quad + \min \left\{ \begin{array}{l} \varphi[\delta, y - 1], \\ \varphi[\delta - 1, y - 1] + \kappa_{\mathrm{occ}}, \ldots, \\ \quad \varphi[0, y - 1] + \kappa_{\mathrm{occ}}, \\ \varphi[\delta + 1, y - 2] + \kappa_{\mathrm{occ}}, \ldots, \\ \quad \varphi[\Delta, y + \delta - \Delta - 1] + \kappa_{\mathrm{occ}} \end{array} \right\}, \end{aligned} \tag{3}$$

where the minimum is taken over the costs of the best paths to the possible preceding matches of $(y + \delta, y)$: first the match preceding no occlusion, then the

matches preceding left occlusions, then the matches preceding right occlusions. Once the $\varphi$ array is filled, the lowest-cost cell which satisfies constraint C3 is selected as the ending match. Then, starting at this cell, $\pi$ is traced to find the optimal match sequence. Pseudocode for all three algorithms can be found in the appendix.

The Backward-Looking Algorithm iterates through all the cells $[\delta, y]$ of the shifted search grid, computing Eq. (3) for each cell. When a cell is encountered, all of its possible preceding matches $[\delta_p, y_p]$ are checked to determine which one lies on the best path to the cell. The algorithm gets its name from the fact that it looks backward to the preceding matches.

The Forward-Looking Algorithm splits the minimization of Eq. (3) so that $\varphi[\delta, y]$ is computed sequentially as follows: the minimum of the first two arguments, then the minimum of its current value and the third argument, then the minimum of its current value and the fourth argument, and so on. It iterates through the cells $[\delta_p, y_p]$ of the search grid, determining for each cell whether that cell lies on the best path to one of its possible following matches $[\delta, y]$. The path to the cell itself is not updated, since its best path has already been computed; rather, the paths to its possible following matches are updated. Therefore, it takes $\Delta + 1$ iterations (because each match has this number of possible preceding matches) before $\varphi[\delta, y]$ is equal to $\gamma_0(y + \delta, y)$. The algorithm gets its name from the fact that it looks forward to the following matches.

These two algorithms perform identical computations, and the running time of each is $O(n\Delta^2)$. Since the Backward-Looking Algorithm is slightly more intuitive, the advantage of the Forward-Looking Algorithm may not be obvious at first glance. The answer lies in the fact that, when a cell is encountered by the Forward-Looking Algorithm, the cost of its best path has already been computed. Therefore, we can determine *before* the cell is expanded whether or not it is likely to lie on the optimal path. By performing this test we can prune those cells with high costs, resulting in an algorithm with a greatly reduced running time. The justification and details of pruning are the subject of the next section.

### 6.2. *A Faster Algorithm*

In the interest of optimality, both of these algorithms perform a great deal of unnecessary computation because they compute the best paths to all the cells, even

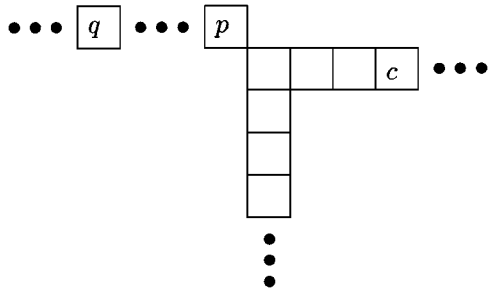


*Figure 13.* Optimality is retained when $p$ is not rightward expanded, assuming that $\gamma_0(q) < \gamma_0(p)$.

to bad ones. The Forward-Looking Algorithm provides a framework within which we can prune these bad cells to produce an algorithm with a greatly reduced running time. Although optimality is sacrificed in theory, the results of the algorithms are nearly identical in practice.[5] Consider a match $p$ with a possible following match $c$ such that there is a right occlusion between them, as shown in Fig. 13 with respect to the original search grid of Fig. 11. Now suppose that there is some match $q$ to the left of and on the same row as $p$ whose best path has a lower cost, i.e., $\gamma_0(q) < \gamma_0(p)$. Then $q$ is also a possible preceding match of $c$ (as is evident from Fig. 11(c)), and the best path to $c$ through $q$ is better than the best path to $c$ through $p$, since the occlusion penalty is constant. Therefore, there is no need for the Forward-Looking Algorithm to expand $p$ to $c$, or indeed to any of the matches on $c$'s row since $q$ is also a possible preceding match of each of them. By a similar argument, we conclude that it is fruitless to expand $p$ to any of the matches on its adjacent column if there is a lower-cost match above it.

In light of these observations we could, without sacrificing optimality, modify the Forward-Looking Algorithm so that it refuses to expand rightward any match with a lower-cost match to its left and refuses to expand downward any match with a lower-cost match above it. However, the running time would not be reduced, because determining whether there is a lower-cost match above or to the left of another match is a complex computation. Instead, we modify the algorithm so that it refuses to expand rightward any match with a lower-cost match in its row and refuses to expand downward any match with a lower-cost match in its column. We call the resulting algorithm the Faster Algorithm.

To see that optimality is lost, consider the situation shown in Fig. 14, in which a match $p$ has a possible following match $c$ such that there is a right occlusion between them, as we had before. Now suppose there

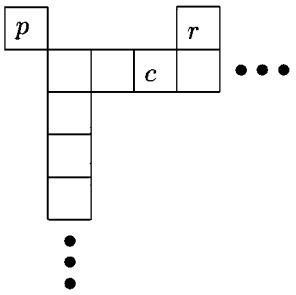


*Figure 14.* Optimality is lost when $p$ is not rightward expanded, assuming that $\gamma_0(r) < \gamma_0(p)$.

is a match $r$ on the same row as $p$ which has a lower-cost best path, i.e., $\gamma_0(r) < \gamma_0(p)$. Also suppose that, by the time $p$ is encountered, the best path to $r$ is better than the best path to $p$, i.e., $\varphi[r] < \varphi[p]$.[6] (Recall that $\varphi[p]$ is guaranteed to be equal to $\gamma_0(p)$ when $p$ is encountered, while $\varphi[r]$ will probably not be equal to $\gamma_0(r)$.) Then the Faster Algorithm will refuse to expand $p$ rightward, since there is a lower-cost cell on its row. Yet if there is no match to the left of $p$, or even to the left of $c$, whose best path has a lower cost than that of $p$, then the best path to $c$ might very well pass through $p$. Therefore, optimality is lost, although it is not surprising that the loss is small, given the large number of assumptions that have to be made.

The average running time of the Faster Algorithm is estimated empirically to be $O(n\Delta \log \Delta)$, as shown in Fig. 15. (Recall that $n$ is the number of pixels in the scanline.)

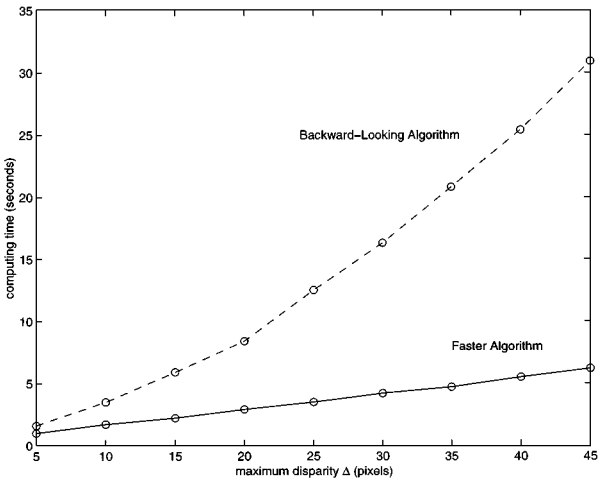


*Figure 15.* Computing time vs. $\Delta$ of the two algorithms on 630 × 480 images, using a 333 MHz Pentium II microprocessor.

## 7. Propagating Information Between Scanlines

For natural scenes, disparities in adjacent scanlines are not independent from each other. While processing scanlines independently is computationally attractive, it does not take full advantage of all the information in the images. Therefore, to produce a disparity map that is closer to the true solution, we seek a way to incorporate this information.

Although a few algorithms avoid incorporating this information altogether (Geiger et al., 1995; Intille and Bobick, 1994), a more common approach is to extend the one-dimensional cost function (such as Eq. (1)) to a two-dimensional cost function. Previous attempts to minimize such a function, however, have been computationally expensive, increasing the computing time by 800% or more in the extension from 1D to 2D (Belhumeur and Mumford, 1992; Belhumeur, 1993; Ohta and Kanade, 1985). Even with such a drain on resources, the global minimum is not guaranteed to be found, because the 2D cost function is minimized by local perturbations of an initial disparity map obtained by matching the scanlines independently (Belhumeur and Mumford, 1992; Belhumeur, 1993). Such local techniques would be often unsuccessful on poorly textured images because of the possible occurrence of large regions that are initially labeled with the wrong disparity. The concavity of the lamp or the area to the right of the lamp in Fig. 10 are cases in point.

In light of these observations, we have adopted a more pragmatic approach, in which disparities are modified in a postprocessing step only within each image column independently. Although one could contrive images in which this column-wise method would fail, we found it to be satisfactory in all our experiments, as shown in Section 8. We also found it useful to repeat this final postprocessing step once along the rows.

The heuristics we used in this stage are as follows:

1. Adjacent pixels of very similar intensities (see the definition of "intensity variation" in Section 4) can differ in disparity by at most one pixel. This heuristic is a standard limit on the gradient of disparity (Pollard et al., 1985).
2. When disparities must be changed to enforce the previous heuristic, very small regions are assumed to be the ones with the wrong disparity.
3. When large adjacent regions exist that are not separated by a sufficient intensity variation, small

disparities override large disparities. This is because foreground objects, even when poorly textured, are usually surrounded by visible boundaries. As discussed in Section 4, these boundaries are assigned the foreground disparity. As a consequence, large regions with the wrong disparity are usually in the background (smaller disparity). Examples of this situation are the regions inside the lamp's concavity, to the right of the lamp, in between the bottles, and below the table in Fig. 16.

Based on these heuristics, we have devised a method for postprocessing the disparity map by propagating reliable disparities into regions of unreliable disparities. This postprocessing is rather global in nature and is quite effective at propagating the background disparities into regions with little intensity variation.

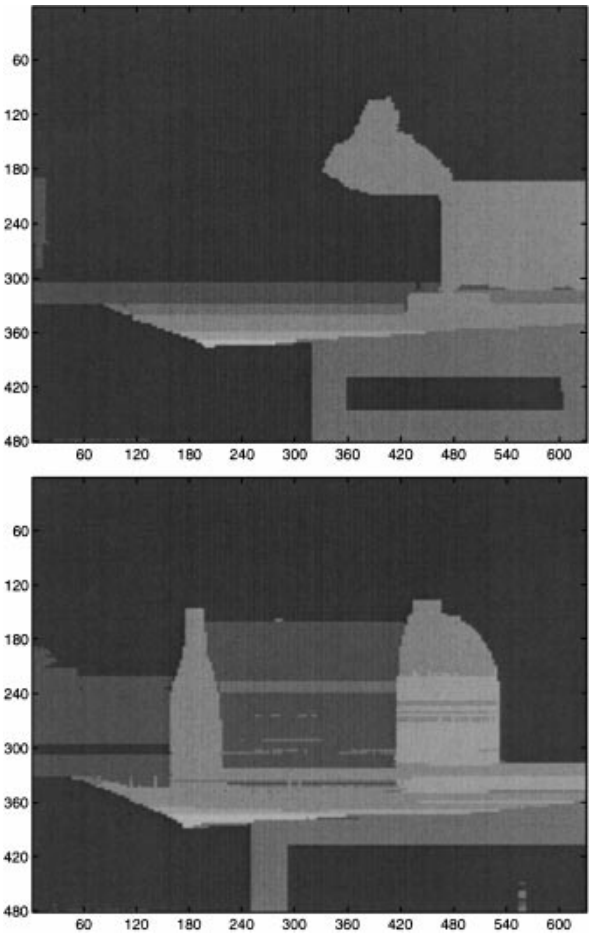


*Figure 16.* The disparity maps obtained after the first propagation step.

Moreover, it is fast, increasing the overall processing time by less than 30%.

Each pixel is assigned a level of *reliability*, which is determined by the number of contiguous pixels in the column agreeing on their disparity. As an example, if the disparities in a column are

$$[5\,7\,7\,7\,8\,8\,2\,7\,7\,7\,7\,7]^T,$$

then the reliabilities of the pixels are

$$[1\,3\,3\,3\,2\,2\,1\,5\,5\,5\,5\,5]^T.$$

(The superscript $T$ denotes transpose, indicating that these are column vectors.)

Each pixel is then categorized based upon its reliability, using a threshold $t_r$. Any pixel whose reliability is at least $(1+\alpha)t_r$ is said to be *reliable*, while any pixel whose reliability is less than $(1-\alpha)t_r$ is said to be *unreliable*, where the factors containing $\alpha$ create a small buffer zone to prevent unnecessary dependence on $t_r$. We can think of reliable pixels as being aggressive in propagating their values into neighboring regions, while unreliable pixels are defenseless in maintaining their values.

The first step of the postprocessor is rather straightforward. After cleaning "obvious" errors in the disparity map by coercing pixels which are surrounded on top and bottom by the same disparity, every pixel that is reliable propagates along its column, changing the disparities of any of the unreliable pixels it encounters, until it reaches an intensity variation (computed now in the vertical direction). This propagation step enforces heuristic 2, and is quite effective at removing the noisy disparities, as seen by the results of Fig. 16.

Heuristic 3 is then incorporated by propagating the background into the foreground, similar to the way we assigned the background disparity to the untextured pixels in Section 4.

We must be careful, though, not to destroy the untextured surfaces that are slanted in the $y$ direction, which give rise to genuine changes in disparity without any accompanying intensity variation (see the table in the lamp image, for example). Heuristic 1, then, forbids propagation when two adjacent regions differ by only one disparity level.

In summary, the second postprocessing step propagates reliable regions into their neighbors until an intensity variation is found, as long as the disparity of the neighbor is greater than the disparity of the propagating region by at least two levels. The results of this second step are shown in Fig. 17, where we see that the

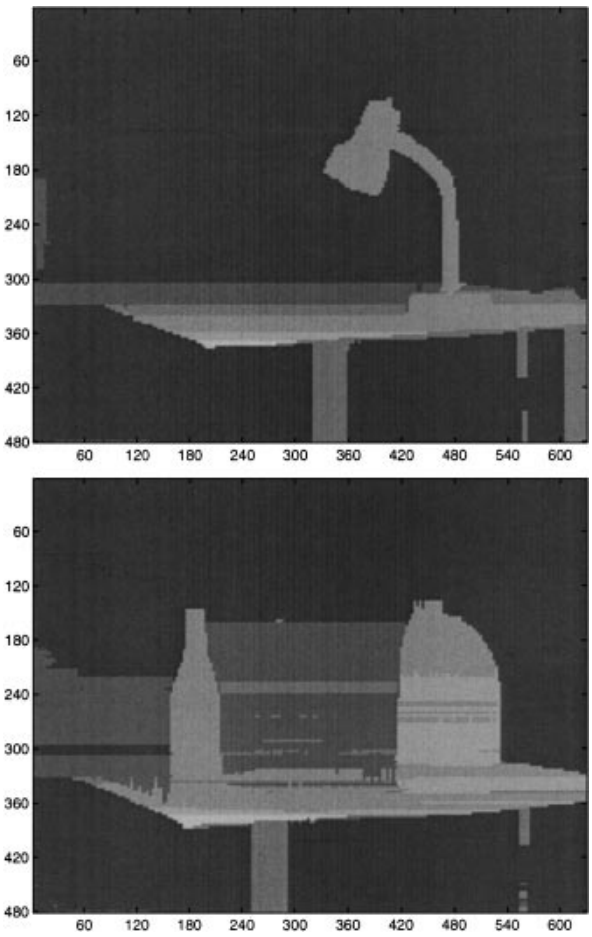


*Figure 17.* The disparity maps obtained after the second propagation step.

background has effectively propagated in many places while preserving the slant of the table. In between the bottles, however, some legitimate propagation was prevented because the background and the middle bottle differ by only one disparity level.

Below the table, we also see that the intensity variation on the door hinge prevented some untextured regions from being overridden, and the cord below the table is falsely declared to be vertical. So, the two propagation steps are repeated, this time along the rows. That is, reliability is determined by the number of contiguous pixels in a row agreeing on their disparity, and disparities are then propagated horizontally. This propagation has less theoretical justification than that done by columns, but it helps to fill in some of the remaining gaps, as seen in the figures of the next section. As a final step, the disparity map is cleaned by mode filtering. The final results on our two running examples are shown in Figs. 18 and 22 in Section 8, which presents several other sets of results as well.

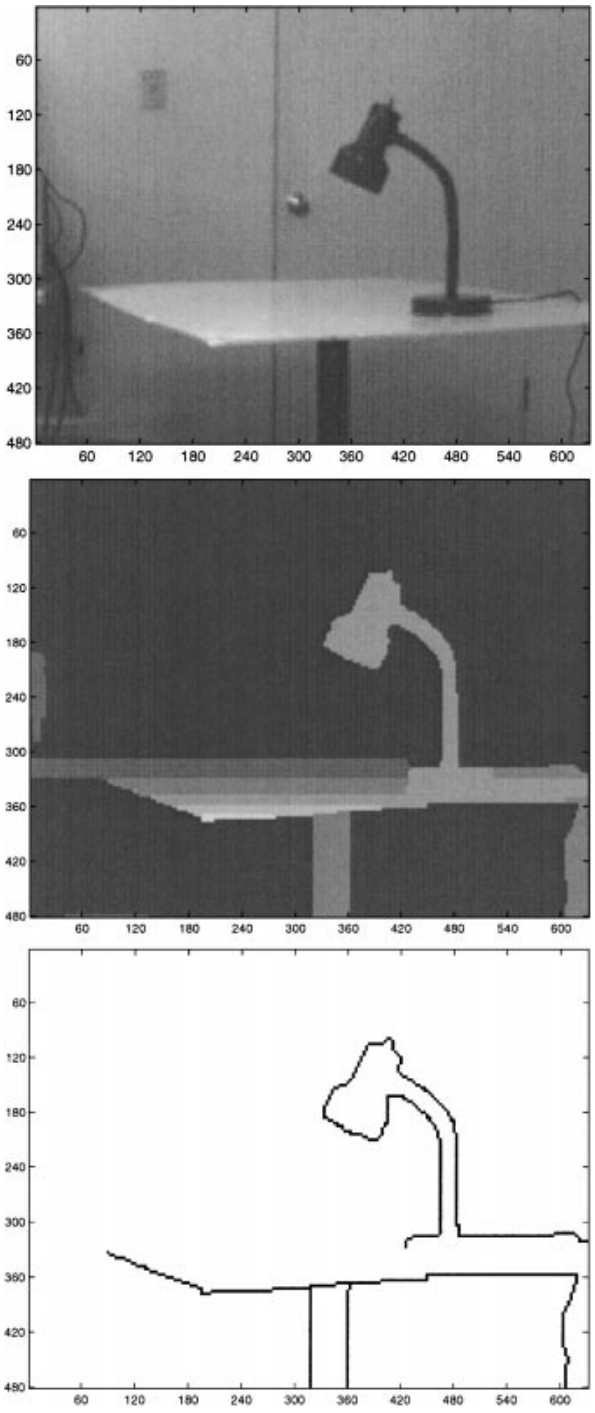


*Figure 18.* The lamp images.

## 8. Experimental Results

In this section we present the results of our algorithm on nine different stereo image pairs. The first six were taken in our laboratory with a single Pulnix camera which was translated along a baseline of 10 mm;

there was little geometric distortion due to the long focal length (16 mm) of the Nikon lens, which was slightly defocused to remove aliasing. The last three are from other laboratories. These nine image pairs represent a wide variety of situations, such as indoor and outdoor scenes, textured and untextured objects, textured and untextured backgrounds, curved and planar surfaces, specular and matte surfaces, and frontoparallel and slanted surfaces (in the $x$ and $y$ directions). Our algorithm's output—both disparity maps and depth discontinuities—are shown in Figs. 18–25 with respect to the left image. The depth discontinuities are defined as those pixels that border a change of at least two disparity levels and that lie on the far object.

The output on these images demonstrates the effectiveness of the algorithm at reconstructing a rough disparity map and, more importantly for our purposes, accurate depth discontinuities. The algorithm correctly places the depth discontinuities along the object contours, even when the background wall has roughly constant intensity. Moreover, even though the discontinuities are represented as binary maps with no linking, they form contiguous one-pixel-thick chains around the objects.

In Fig. 18, the depth discontinuities are nearly perfect. Notice that the depth discontinuities are correctly placed along the edges of the table support and the lamp cord, even though the only texture between the two is a little door hinge. Also, the table is recovered as a series of constant-disparity strips whose disparity decreases as the table recedes.

It is instructive to imagine how other stereo algorithms would handle these lamp images. By doing so, we will discover that this pair of images, although perhaps appearing easy at first glance, actually presents a challenge to existing techniques. Algorithms such as those by Geiger et al. (1995), Intille and Bobick (1994), Belhumeur and Mumford (1992), Belhumeur (1993), and Cox et al. (1996), which match the scanlines independently but have no mechanism for preferring to place discontinuities near intensity variation (or intensity edges) would not place the discontinuities along the contour of the lamp. Moreover, the concavity of the lamp would not be filled in, because none of their postprocessing methods use intensity variation, and in fact the first two methods do not postprocess the disparity map at all. Similarly, the algorithms of Luo and Burkhardt (1995) and of Jones and Malik (1992) would fail to find the boundary between the lamp and the background because they also have no mechanism for preferring intensity variation. Although they use windows to accumulate support, the windows would invariably be smaller than our untextured regions. More promising are the algorithms of Fua (1991) and of Cochran and Medioni (1992), which try to align the depth discontinuities with the intensity edges, but it is not clear how well they would perform

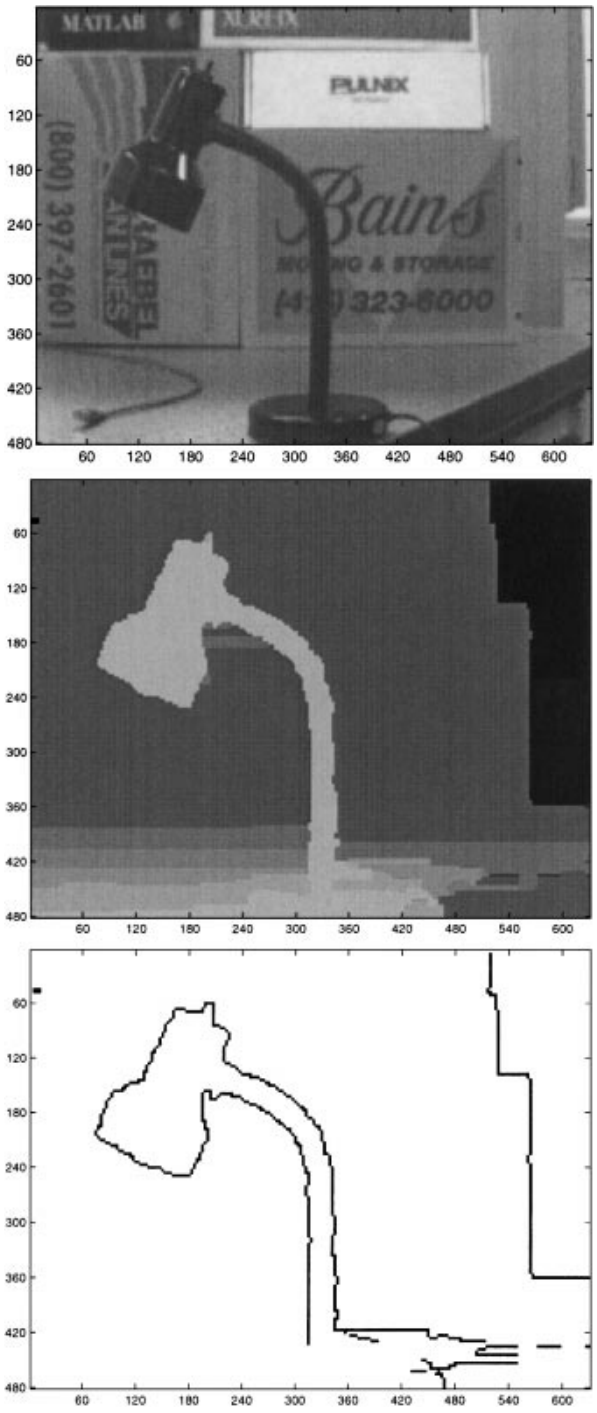


*Figure 19.* The textured lamp images.

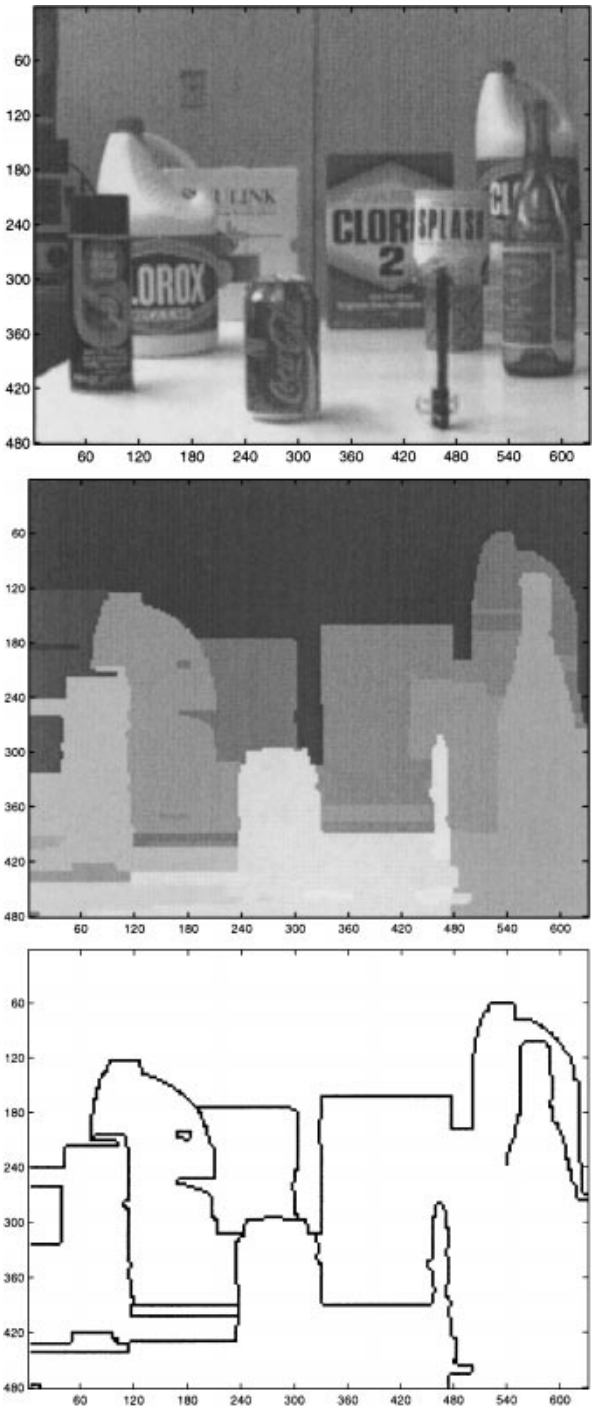


*Figure 20*. The random objects images.

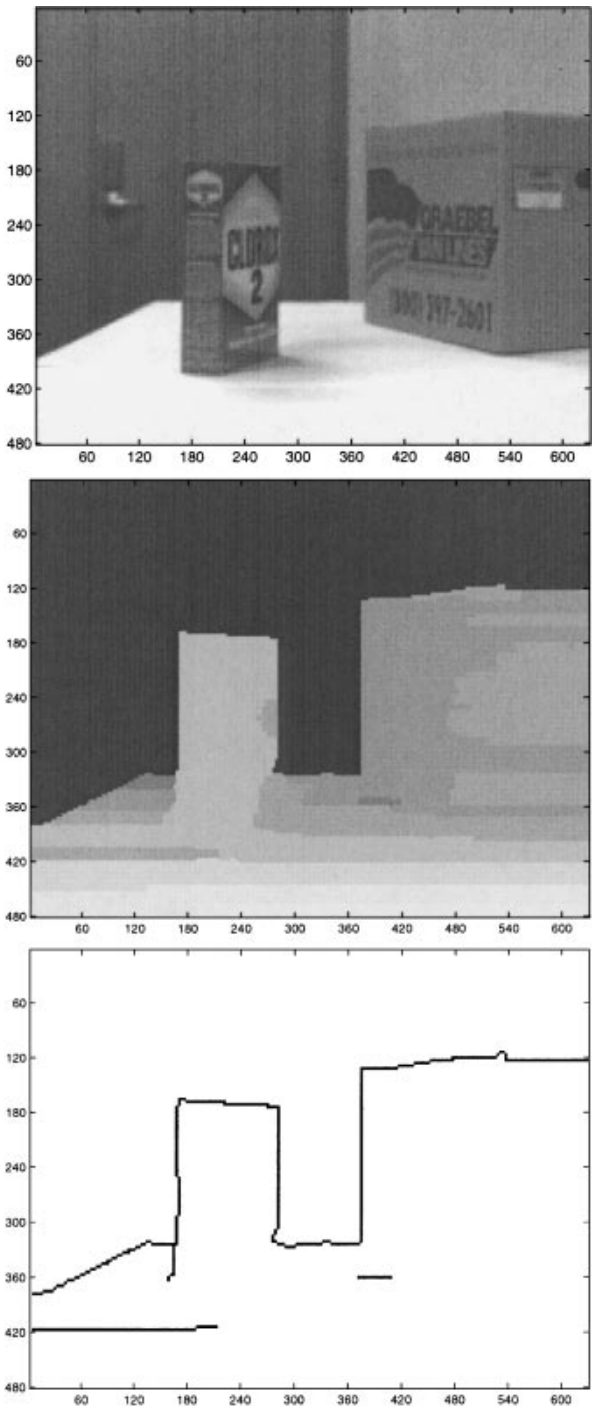


*Figure 21*. The slant images.

on these images because the initial disparity map would be so far from the true solution. As another approach, feature-based algorithms (Baker and Binford, 1981; Ohta and Kanade, 1985; Grimson, 1985) are good candidates for handling untextured regions, but they require smart interpolation schemes. Moreover, they use large thresholds for declaring intensity edges (as opposed to our small threshold for declaring intensity

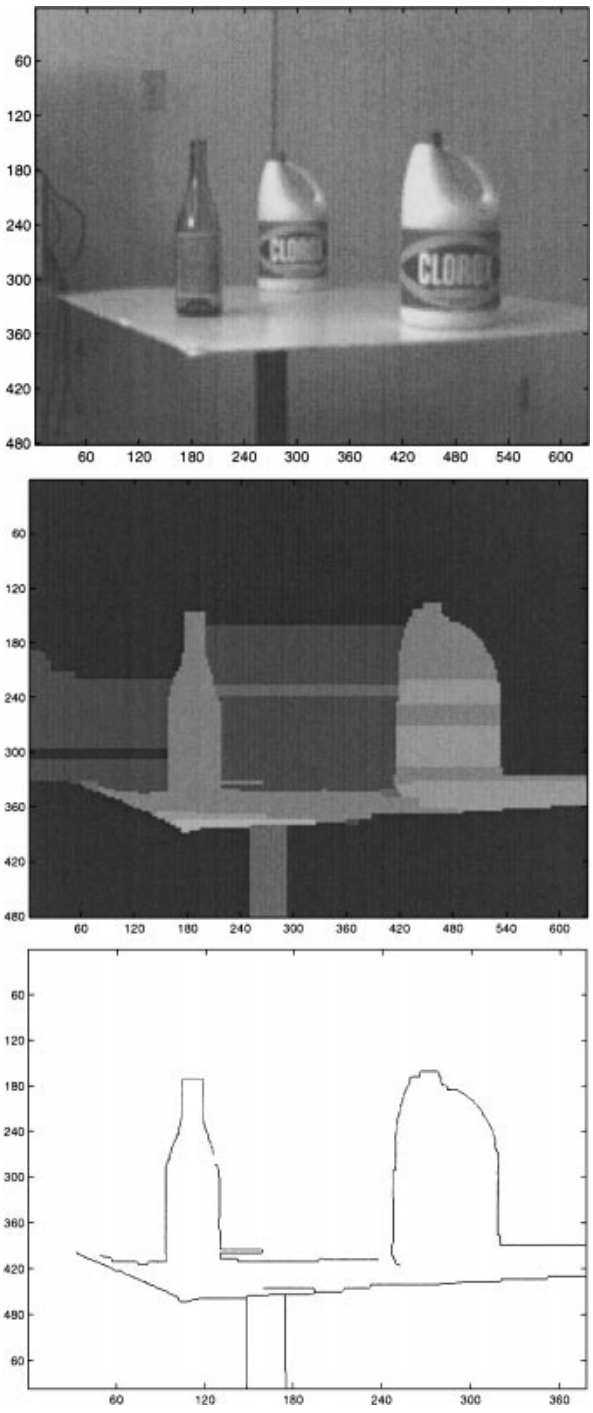

*Figure 22*. The Clorox images.

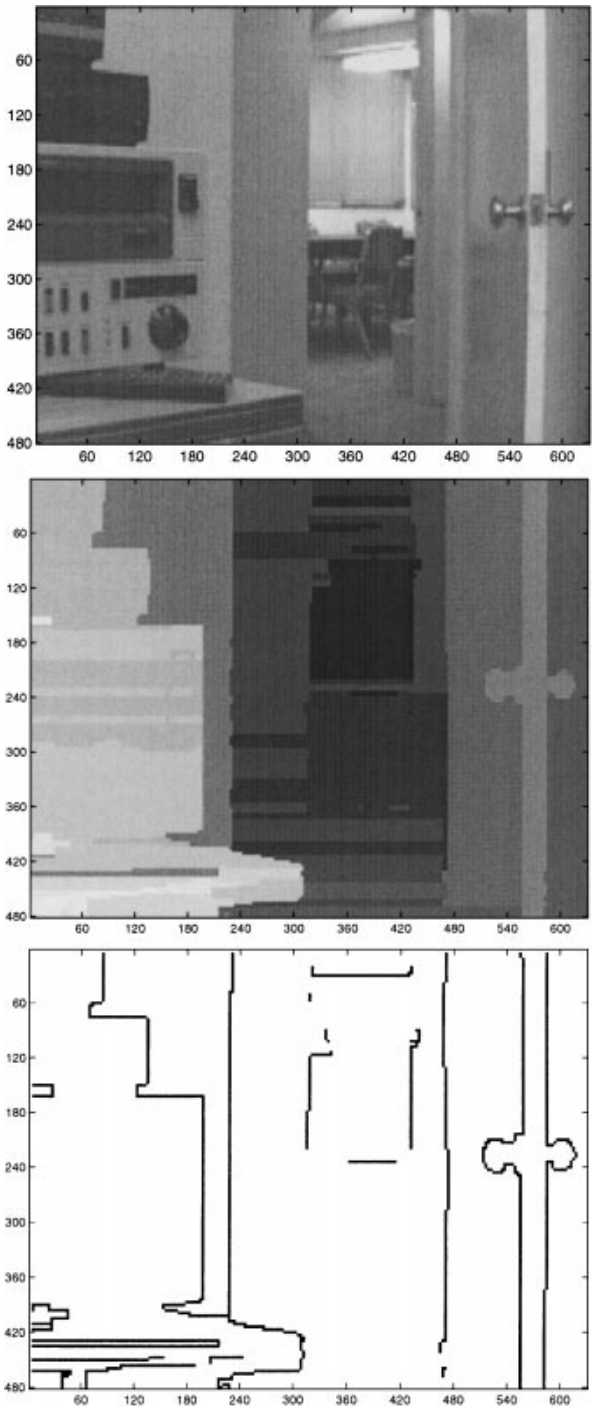

*Figure 23*. The doorway images.

variation), and thus eliminate much of the information in the image.

Returning to our algorithm, we note that it does not *require* untextured regions—as long as some intensity variation still accompanies the depth discontinuities. For example, Fig. 19 shows similar performance in the case of a textured background, although the depth discontinuities around the lamp are more jagged (Notice

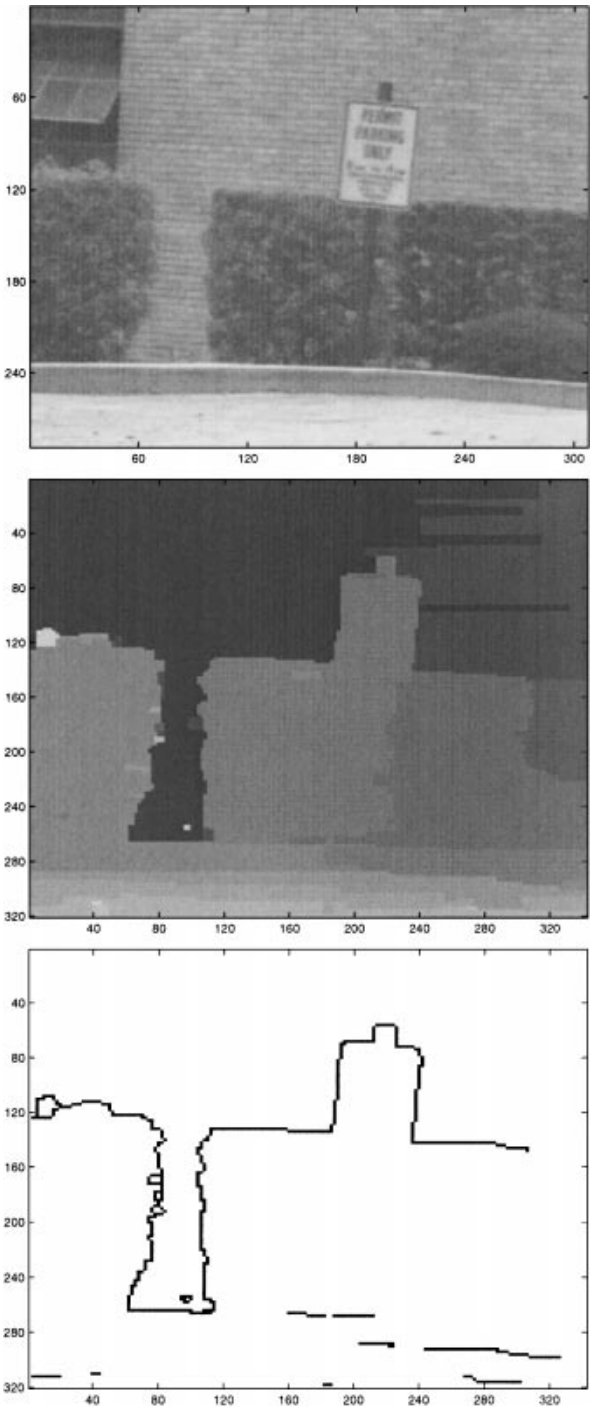


*Figure 24*. The shrub images.

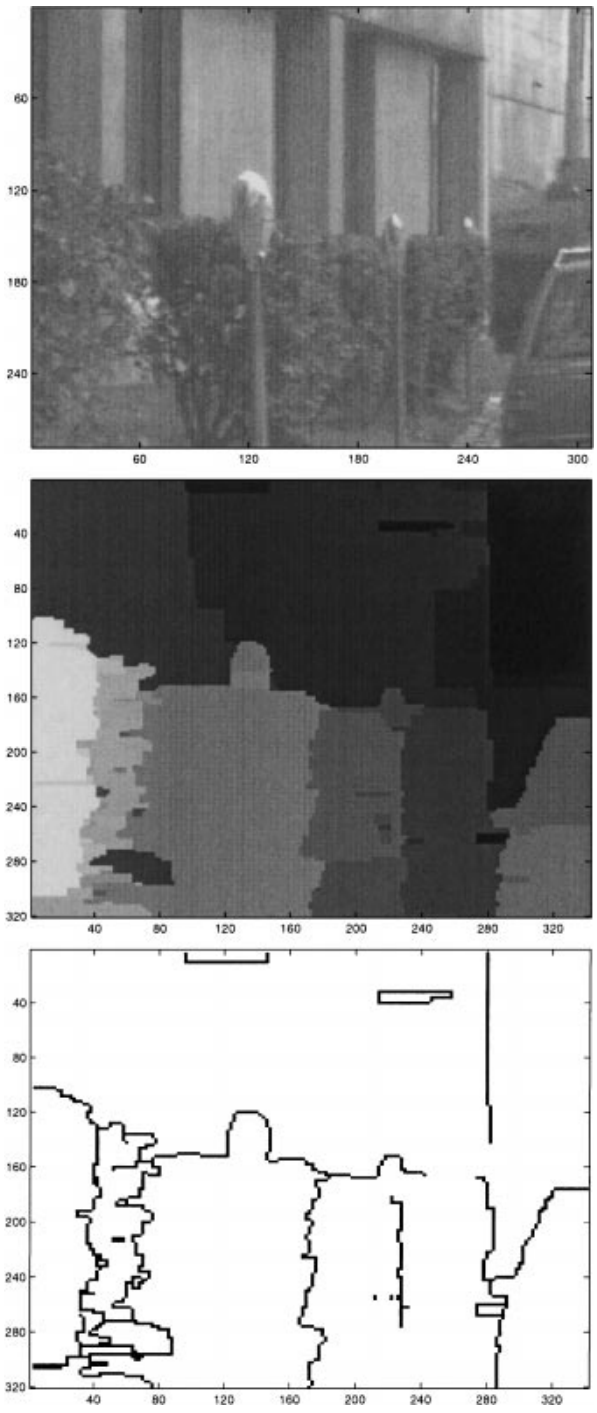


*Figure 25*. The meter images.

the accurate detection of the boundary between the boxes and the background). As another example, the top section of the wine bottle in Fig. 20 is correctly recovered, as well as the right edge of the spray paint can (column 110). The algorithm does not need *strong* intensity edges, as is evident from the barely discernible edge of the recorder in Fig. 23 (column 190), although without any intensity variation the algorithm will fail

(see the triangular wedge cut out of the left Clorox bottle in Fig. 20).

The results of Fig. 21 are especially worth noting. Even though the algorithm generally assumes fronto-parallel surfaces and has no explicit representation of a slanted surface, the depth discontinuities are recovered in the presence of both horizontal and vertical slant. Of course, the disparity map is only approximate (and yet the receding disparity strips of the table are preserved), but the contours around the objects are crisp.

Both the power and the drawback of ignoring one-level disparity transitions in the labelling of depth discontinuities are evident from these images. On the one hand, many false transitions are ignored, such as those on the slanted table and the boxes of Fig. 21. At the same time, however, some true transitions are improperly missed, such as the back edge of the table in Fig. 18 and the middle Clorox bottle in Fig. 22. It is important to note that, even in principle, this problem of using a threshold can never be eliminated completely, because it is impossible to determine the discontinuities of a continuous function from discrete data.

Figure 23 shows the algorithm's output in a different type of scene—one that a mobile robot might encounter as it navigates through a doorway. In these images, the extreme slant of the door on the right causes the algorithm to mistakenly label the door edge as a depth discontinuity when in reality it is a surface-normal discontinuity. In addition, the errors in the center of the image are caused by a lack of information along scanlines to differentiate a disparity of two from a disparity of three, and once several adjacent scanlines have incorrectly labeled the disparity, our postprocessor cannot recover. Nevertheless, the outline of the recorder, table, doorway, door, and distant background are all accurately recovered.

Figures 24 and 25 show the algorithm's effectiveness on two standard outdoor image pairs from the well-known JISCT data set (Bolles et al., 1993). In Fig. 24 the boundary between the wall of the building and the foreground is, for the most part, accurately recovered (the errors between the shrub and the wall are due to the wind blowing the leaves while the camera was being moved). Notice that the slope of the ground plane is also correctly preserved. In Fig. 25, although the discontinuities around the bush on the left are very jagged and two discontinuities are incorrectly labeled near the top of the image, many of the true discontinuities are found. For example, the boundary between the wall of the building and the foreground objects, the boundaries between the individual bushes, and one side of the middle parking meter are all approximately detected (the left meter cannot be separated from the bush behind it without subpixel resolution). Moreover, the vertical boundary between the two buildings and the outline of the car are accurately recovered (the error just below the mirror is due to a photometric difference between the two images—perhaps a shadow appeared while the camera was being moved).

In Fig. 26 we show the results of our algorithm on a stereo pair of images from the University of Tsukuba Multiview Image Database,[7] for which ground truth is available. The basic structure of the scene, including sharp discontinuities, is preserved, although significant errors occur in the top-right corner of the image, the video camera and table, and the neck of the lamp. According to ground truth, 80% of the pixels in the disparity map have the correct value, and over 96% have a disparity within one level of the correct value (See Fig. 27). This latter value is useful because it allows for potential discretization errors in the ground truth. From Table 1, whose entries come from Boykov et al. (1998), we see that our algorithm outperforms every other method except for their multiway-cut algorithm (GPM-MRF), which takes at least two orders of magnitude as much computing time as ours.

Probably the main drawback to our algorithm is its brittleness. Because of its emphasis on speed and on preserving sharp changes in disparity, the algorithm is heavily dependent upon local information. If a boundary has no accompanying intensity variation for several scanlines in a row, then that boundary will not be found. Similarly, if an object has no texture (imagine, for example, if the door hinge in the lower-right section of Fig. 18 were not present), then disparities are not carried from another, disjointed region. Moreover, the postprocessing steps can produce unexpected results

*Table 1*. Comparison with other algorithms on the images from the University of Tsukuba. Algorithm names are from (Boykov et al., 1998).

| Algorithm | Total errors (%) | Errors $> \pm 1$ (%) |
|---|---|---|
| GPM-MRF (Boykov et al., 1998) | 8.6 | 2.8 |
| Pixel-to-pixel (this paper) | 19.0 | 5.7 |
| LOG-filtered $L_1$ | 19.9 | 9.0 |
| Normalized correlation | 24.7 | 10.0 |
| MLMHV (Cox et al., 1996) | 24.5 | 11.0 |

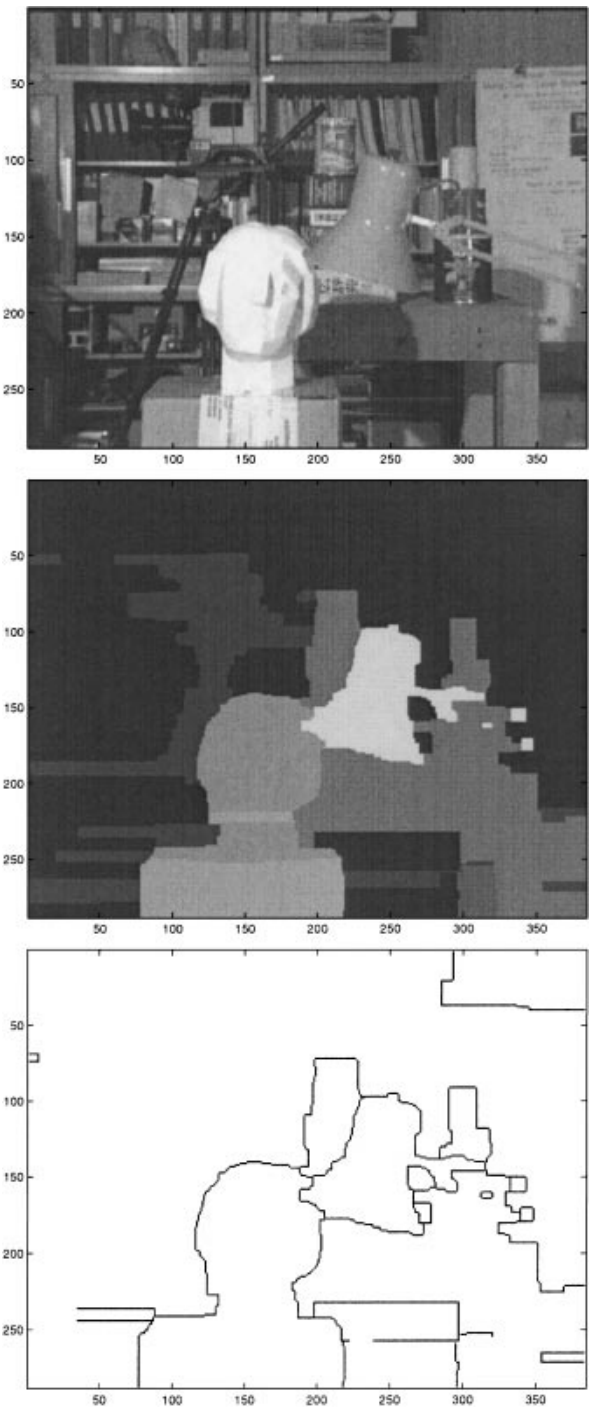


*Figure 26.* The images from the University of Tsukuba.

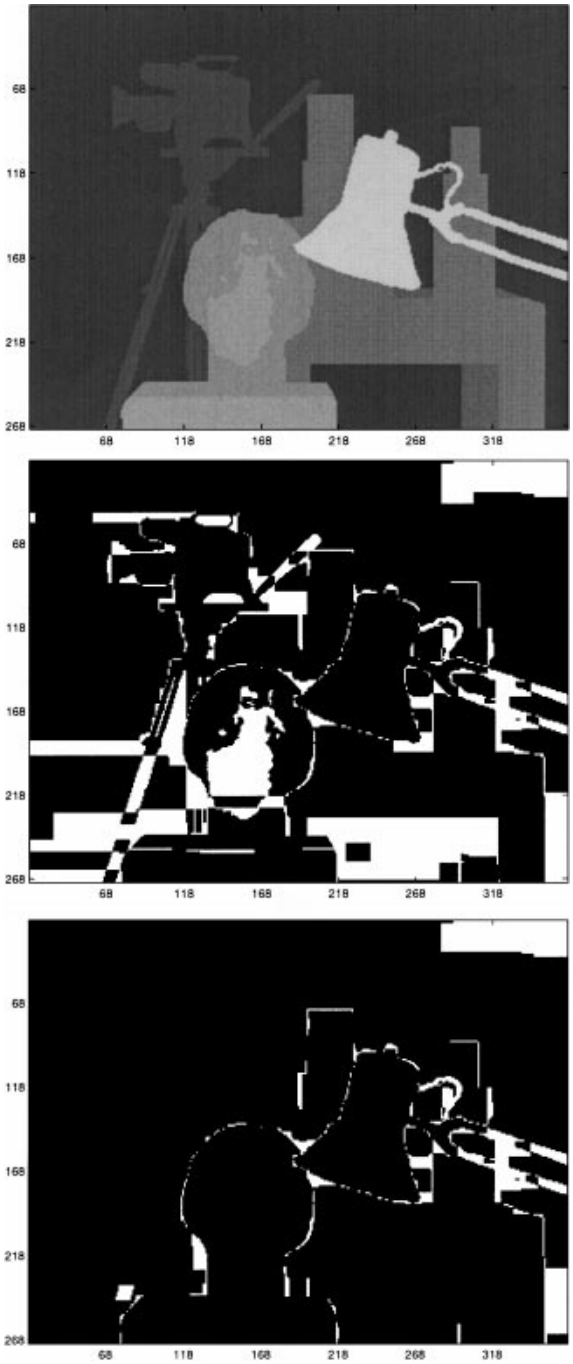


*Figure 27.* Top: Ground truth disparity map; Middle: White pixels have incorrect disparity in our map; Bottom: White pixels have a disparity error greater than one level.

on slanted surfaces, such as the hashing of the interior of the right box of Fig. 21.

Because the algorithm matches pixel intensities directly without windows, it requires high-quality images. The cameras must be aligned, and their gains and biases calibrated. We have noticed a substantial decrease in performance when aliasing is present, the images are subsampled, or the baseline is increased,

all of which cause the intensities of objects to look less similar in the two images.

### *8.1. Sensitivity to Parameters*

For all the results presented in this section, the parameters were set to the values in Table 2. To determine the algorithm's sensitivity to these parameters, we varied each one in turn while keeping the others fixed at their chosen values.

We found the algorithm to be entirely insensitive to the choice of $\Delta$ as long as it is larger than the actual maximum disparity: As $\Delta$ varies from 20 to 50, almost none (fewer than 0.3%) of the pixels in the disparity map change value. To a lesser extent, the algorithm is also fairly insensitive to the other parameters: Fewer than 3% of the pixels change value as $\kappa_{\text{occ}}$ varies $\pm 10\%$, $\kappa_{\text{r}}$ varies $\pm 40\%$, $t_{\text{r}}$ varies $\pm 20\%$, and $\alpha$ varies $\pm 50\%$.

Subjectively, this 3% window of change corresponds to no noticeable degradation of the disparity map. The only exception occurs when $(1-\alpha)t_{\text{r}}$ becomes so small that the disparity strips on the table of the lamp image are labeled as unreliable, which causes the farthest strip of pixels (having the smallest disparity) to propagate into the other strips, thus yielding a table with a single disparity. The general rule of thumb is that the parameter $(1-\alpha)t_{\text{r}}$ restricts the maximum allowable vertical slant of an untextured object in the scene; any object that slants more than the parameter allows will be labeled with a single disparity unless it contains enough texture to prevent propagation.

### *8.2. Computing Time*

Compared with other stereo algorithms, our method is fast. A personal computer equipped with a 333 MHz Pentium II microprocessor needed less than four seconds—2.9 for scanline matching and 0.8 for post-processing—to compute disparity maps and depth discontinuities from these $630 \times 480$ images, with $\Delta = 20$. The results of Fig. 18 can be produced in just over two seconds by setting $\Delta = 9$.

*Table 2.* The parameter settings.

| | | |
|---|---|---|
| Maximum disparity | $\Delta$ | 20 |
| Occlusion penalty | $\kappa_{\text{occ}}$ | 25 |
| Match reward | $\kappa_{\text{r}}$ | 5 |
| Reliability threshold | $t_{\text{r}}$ | 14 |
| Reliability buffer factor | $\alpha$ | 0.15 |

## 9. Necessity of Our Dissimilarity Measure

As explained in Section 5, the absolute difference in intensity is often adequate for comparing two pixels in stereo matching. In fact, when comparing the absolute difference with our dissimilarity measure, we found that fewer than 10% of the pixels in the disparity maps changed on our six images (Birchfield and Tomasi, 1998b). However, wherever the intensity function was changing rapidly and the disparity was not an integral number of pixels, our measure was crucial to recovering accurate disparities. In this section we will show that the improvements were not due to the choice of algorithm parameters. Specifically, we will demonstrate that there is no possible choice of parameter values that will enable the absolute difference measure to perform as well as our measure.

As the parameter values $\kappa_{\text{occ}}$ and $\kappa_{\text{r}}$ are varied, the disparity maps transition between two error modes: (1) if the overall occlusion penalty is too small, then false disparity changes are declared, or (2) if the overall penalty is too large, then true disparity changes are ignored. The key to producing a reasonable disparity map is to find a set of parameter values that is somewhere between these two error modes.

We first examined the region on the right Clorox bottle shown in Fig. 28(a). Since the correct disparity is either seven or eight pixels, we counted the number of pixels in the region with a disparity not equal to either seven or eight, as we varied $\kappa_{\text{occ}}$ and $\kappa_{\text{r}}$. The results are shown in Table 3(a), where the dots represent values that are at least as good as the one obtained with our measure (zero). Notice that to remove the errors on the Clorox bottle, the match reward $\kappa_{\text{r}}$ must be increased dramatically from 5 to 25.

Then we examined the region surrounding the neck of the lamp, as shown in Fig. 28(b). As we varied the parameters, we again counted the number of pixels with incorrect disparity (The correct disparity was either eleven or twelve pixels). The results are shown in Table 3(b). As the overall occlusion penalty is increased, it becomes more and more difficult to declare the occlusions necessary to distinguish the lamp from the boxes behind it. To get a feel for what these

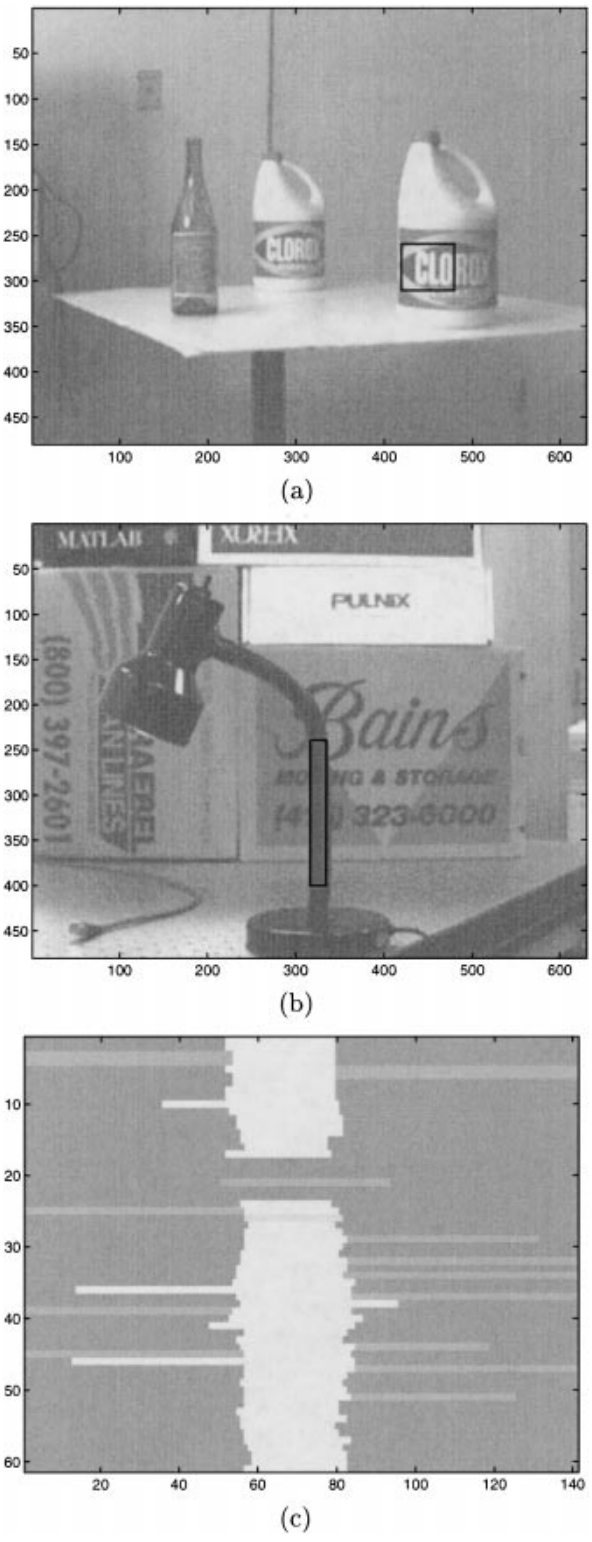


*Figure 28.* (a) and (b) The regions examined; (c) The disparity map around the lamp neck when $\kappa_{\text{occ}} = 20$ and $\kappa_{\text{r}} = 14$ (268 wrong pixels), using absolute difference.

*Table 3.* (a) The number of pixels in the window on the Clorox bottle with incorrect disparity, using absolute differences. The dots represent values that are at least as good as with our measure of dissimilarity (we had zero errors); (b) The number of pixels in the window on the lamp neck with incorrect disparity using absolute differences. The dots represent values that are at least as good as the value obtained using our measure (173).

(a)

| | $\kappa_{\text{occ}}$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\kappa_{\text{r}}$ | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
| 2 | 1386 | 1284 | 1213 | 1099 | 973 | 850 | 686 | 556 |
| 4 | 1056 | 967 | 881 | 748 | 557 | 472 | 368 | 282 |
| 5 | 906 | 784 | 660 | 562 | 435 | 376 | 284 | 251 |
| 6 | 746 | 618 | 552 | 447 | 347 | 296 | 267 | 222 |
| 8 | 577 | 465 | 382 | 316 | 271 | 246 | 198 | 198 |
| 10 | 385 | 345 | 276 | 268 | 216 | 198 | 189 | 173 |
| 12 | 291 | 254 | 254 | 197 | 189 | 189 | 173 | 129 |
| 14 | 237 | 215 | 189 | 189 | 181 | 129 | 129 | 111 |
| 15 | 187 | 189 | 189 | 173 | 129 | 111 | 111 | 87 |
| 20 | 110 | 111 | 87 | 87 | 87 | 72 | 48 | 24 |
| 25 | 48 | 48 | 24 | • | • | • | • | • |

(b)

| | $\kappa_{\text{occ}}$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\kappa_{\text{r}}$ | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
| 2 | • | • | • | • | • | • | • | • |
| 4 | • | • | • | • | • | • | • | 211 |
| 5 | • | • | • | • | • | • | • | 230 |
| 6 | • | • | • | • | • | • | 211 | 249 |
| 8 | • | • | • | • | • | 230 | 268 | 382 |
| 10 | • | • | • | 192 | 249 | 287 | 363 | 401 |
| 12 | • | • | 192 | 249 | 287 | 325 | 401 | 401 |
| 14 | 212 | 230 | 268 | 306 | 363 | 420 | 420 | 456 |
| 16 | 250 | 268 | 344 | 382 | 439 | 439 | 457 | 551 |
| 20 | 458 | 477 | 496 | 515 | 533 | 570 | 627 | 684 |
| 25 | 515 | 534 | 552 | 589 | 627 | 684 | 684 | 703 |

numbers mean, consider Fig. 28(c) which shows one of the disparity maps with 268 incorrect pixels. Notice that there are six consecutive scanlines with incorrect disparity on the lamp neck.

The intersection of the dots in Tables 3(a) and (b) is null: There is no choice of parameters that finds all of the lamp neck without declaring spurious discontinuities on the Clorox bottle.

```
 1  for δ ← 0 to Δ
 2      φ[δ, 0] ← d(δ, 0)
 3  for y ← 1 to n − 1
 4      for δ ← 0 to Δ
 5          φ̂ ← ∞
 6          for δ_p ← 0 to Δ
 7              y_p ← y − max(1, δ_p − δ + 1)
 8              if (δ = δ_p) or (δ > δ_p and v_L[y + δ − 1])
                            or (δ < δ_p and v_R[y_p + 1]) then
 9                  φ' ← φ[δ_p, y_p] + κ_occ * (δ ≠ δ_p)
10                  if φ' < φ̂ then
11                      φ̂ ← φ'
12                      π̂ ← [δ_p, y_p]
13          φ[δ, y] ← φ̂ + d(y + δ, y) − κ_r
14          π[δ, y] ← π̂
```

*Figure 29.* Backward-Looking Algorithm.

```
 1  for δ ← 0 to Δ
 2      φ[δ, 0] ← d(δ, 0)
 3  for y ← 1 to n − 1
 4      for δ ← 0 to Δ
 5          φ[δ, y] ← ∞
 6  for y_p ← 0 to n − 2
 7      for δ_p ← 0 to Δ
 8          for δ ← 0 to Δ
 9              y ← y_p + max(1, δ_p − δ + 1)
10              if (δ = δ_p) or (δ > δ_p and v_L[y + δ − 1])
                            or (δ < δ_p and v_R[y_p + 1]) then
11                  φ' ← φ[δ_p, y_p] + d(y + δ, y) − κ_r + κ_occ * (δ ≠ δ_p)
12                  if φ' < φ[δ, y] then
13                      φ[δ, y] ← φ'
14                      π[δ, y] ← [δ_p, y_p]
```

*Figure 30.* Forward-Looking Algorithm.

```
 1  for δ ← 0 to Δ
 2      φ[δ, 0] ← d(δ, 0)
 3  for y ← 1 to n − 1
 4      for δ ← 0 to Δ
 5          φ[δ, y] ← ∞
 6  for x ← 0 to n − 1
 7      m_x[x] ← ∞
 8  for y_p ← 0 to n − 2
 9      m_y ← min{φ[0, y_p], φ[1, y_p], ..., φ[Δ, y_p]}
10      for δ_p ← 0 to Δ
11          update(δ_p, y_p, δ_p, y_p + 1)
12          if φ[δ_p, y_p] ≤ m_y then
13              y ← y_p + 1
14              for δ ← δ_p + 1 to Δ
15                  if v_L[y + δ − 1] then update(δ_p, y_p, δ, y)
16          if φ[δ_p, y_p] ≤ m_x[δ_p + y_p] and v_R[y_p + 1] then
17              for δ ← 0 to δ_p − 1
18                  y ← y_p + δ_p − δ + 1
19                  update(δ_p, y_p, δ, y)

update(δ_p, y_p, δ, y)
20  φ' ← φ[δ_p, y_p] + d(y + δ, y) − κ_r + κ_occ * (δ ≠ δ_p)
21  if φ' < φ[δ, y] then
22      φ[δ, y] ← φ'
23      π[δ, y] ← [δ_p, y_p]
24      m_x[y + δ] ← min{m_x[y + δ], φ'}
```

*Figure 31.* Faster Algorithm.

## 10. Conclusion

Detecting depth discontinuities is an important problem that is rarely emphasized in stereo matching. We have presented an algorithm that sacrifices the usual goal of accurate scene depth in favor of crisp discontinuities. Our algorithm matches the pixel intensities directly in the two images, without windows, thus earning the title, "pixel-to-pixel stereo." The algorithm introduces four novelties: (1) a method for handling large untextured regions, (2) a measure of pixel dissimilarity that is insensitive to sampling effects, (3) a technique for pruning the search space to reduce the running time of dynamic programming, and (4) a postprocessor that quickly propagates disparity information between scanlines to produce a cleaner disparity map.

The algorithm is fast, computes disparities and depth discontinuities in situations where previous algorithms would fail, and yields results that are fairly independent of the amount of texture in the image. As always, however, limitations remain. The algorithm is rather brittle in the sense that the violation of an assumption has the potential of creating large errors in the output. The detected depth discontinuities contain obvious errors, such as boundaries terminating freely in space, jagged boundaries, and tiny, disconnected boundaries. These errors may possibly be removed by incorporating other constraints, such as the smoothness of boundaries (i.e., the "continuity of discontinuities" (Marr and Poggio, 1979)) or a minimum length on boundaries. Finally, the somewhat ad hoc nature of the postprocessor points to the need for a more principled approach without sacrificing speed.

## A. Algorithms

In this appendix we present pseudocode for the three algorithms of Section 6. In addition to the arrays $\varphi[\delta, y]$ and $\pi[\delta, y]$, two arrays are used: $v_L[x]$ is `TRUE` if the pixel $x$ in the left scanline lies to the left of intensity variation, and is `FALSE` otherwise; similarly, $v_R[y]$ is `TRUE` if the pixel $y$ in the right scanline lies to the right of intensity variation, and is `FALSE` otherwise.

### *A.1. Backward-Looking Algorithm*

For simplicity, we have omitted the test `if` $y_p > 0$, which is necessary to ensure that array indexing is in bounds. Line 7 is simply a compact formula to

express the position of the possible preceding matches (Fig. 12(b)). Line 8 prevents $\varphi'$ from being updated if the match being considered would cause a depth discontinuity without intensity variation. Specifically, the test $\delta > \delta_p$ is true when there is a left occlusion, in which case the depth-discontinuity pixel $y + \delta - 1$ must lie to the left of intensity variation; similarly the test $\delta < \delta_p$ is true when there is a right occlusion, in which case the depth-discontinuity pixel $y_p + 1$ must lie to the right of intensity variation. Line 9 updates $\varphi'$ by adding $\kappa_{\text{occ}}$ whenever there is an occlusion. (We are using the convention that $\delta \neq \delta_p$ is equal to 1 if there is an occlusion and zero otherwise; $*$ denotes ordinary multiplication.) After the `for` loop of lines 6–12, $\hat{\pi}$ points to the best preceding match, and $\hat{\varphi}$ holds its value, plus any occlusion penalty. The last two lines use this information to update the $\varphi$ and $\pi$ arrays.

### *A.2. Forward-Looking Algorithm*

Lines 1–5 initialize the $\varphi$ array. The equation in line 9 expresses the position of the possible following matches (Fig. 12(c)). The variable $\varphi'$ contains the cost of the best path (so far) to $[\delta, y]$ through $[\delta_p, y_p]$. If this path is better than all the paths which have already been examined, then the best path to $[\delta, y]$ is updated accordingly.

### *A.3. Faster Algorithm*

The Faster Algorithm utilizes two data structures for determining whether a cell should be expanded: $m_x[x]$ is the minimum cost of any cell in row $x$ (of the original search grid), while $m_y$ is the minimum cost of any cell in the current column. The subroutine `update` updates the path to the match $[\delta, y]$ if the path through $[\delta_p, y_p]$ is better than any path seen previously; it also maintains $m_x$.

Lines 1–7 initialize the $\varphi$ and $m_x$ arrays, while $m_y$ is initialized in line 9. The heart of the algorithm is executed in lines 11–19, which are repeated for every cell $[\delta_p, y_p]$ in the $\varphi$ array. Each cell is expanded to its adjacent following match at the same disparity in line 11. Then, if the cell is one of the best in its column, it is expanded in lines 13–15 to all the cells following a left occlusion, provided that the depth-discontinuity pixel $y + \delta - 1$ lies to the left of intensity variation. Finally, if the cell has the lowest cost among all the cells in its row, and if the depth-discontinuity pixel $y_p + 1$ lies to the right of intensity variation, then the cell is expanded in lines 17–19 to all the cells following a right occlusion. The formula in line 18 is easily understood if one notices that a right occlusion occurs when $x = x_p + 1$, and that $x = y + \delta$ and $x_p = y_p + \delta_p$.

## Acknowledgments

This research was supported by the U.S. National Science Foundation under contract IRI-9506064, and by the Department of Defense under MURI contract DAAH04-96-1-0007 monitored by ARO.

## Notes

1. The *forbidden zone* of a match is the hourglass region defined by the lines of sight from the two cameras to the scene point corresponding to the match. As long as the ordering constraint is valid, the forbidden zone contains no matches (Faugeras, 1993).
2. Two cameras are said to be *rectified* if their image planes are coplanar and their scanlines are parallel to the baseline (Nalwa, 1993).
3. The nomenclature here remains in debate. Some researchers call the ordering constraint by other names, such as the monotonic-ordering constraint (Nalwa, 1993), the monotonicity constraint (Baker and Binford, 1981; Intille and Bobick, 1994), or the non-reversal constraint (Ohta and Kanade, 1985). Geiger et al. (1995) distinguish the ordering constraint from the monotonicity constraint in requiring the latter also to preclude simultaneous occlusions, but this distinction is confusing because the term "monotonicity" simply means that a function is non-decreasing (or non-increasing), which is exactly what the ordering constraint ensures.
4. Informally, we use the terms *path* and *match sequence* interchangeably, as well as the terms *cell* and *match*.
5. Our experiments show that the resulting disparity maps disagree in fewer than 0.7% of their values, for $\Delta$ ranging from 14 to 40 pixels.
6. The meaning here should be clear, despite the abuse of notation.
7. Courtesy of Y. Ohta and Y. Nakamura at the University of Tsukuba, Japan.

## References

Attneave, F. 1954. Some informational aspects of visual perception. *Psychological Review*, 61(3):183–193.

Baker, H.H. and Binford, T.O. 1981. Depth from edge and intensity based stereo. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pp. 631–636.

Belhumeur, P.N. 1993. A binocular stereo algorithm for reconstructing sloping, creased, and broken surfaces in the presence of half-occlusion. In *Proceedings of the 4th International Conference on Computer Vision*, pp. 431–438.

Belhumeur, P.N. and Mumford, D. 1992. A Bayesian treatment of the stereo correspondence problem using half-occluded regions.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 506–512.

Birchfield, S. and Tomasi, C. 1996. Depth discontinuities by pixel-to-pixel stereo. Technical Report STAN-CS-TR-96-1573, Stanford University.

Birchfield, S. and Tomasi, C. 1998a. Depth discontinuities by pixel-to-pixel stereo. In *Proceedings of the 6th International Conference on Computer Vision*, pp. 1073–1080.

Birchfield, S. and Tomasi, C. 1998b. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Black, M.J. and Anandan, P. 1990. Constraints for the early detection of discontinuity from motion. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, Vol. 2, pp. 1060–1066.

Bolles, R.C., Baker, H.H., and Hannah, M.J. 1993. The JISCT stereo evaluation. In *ARPA Image Understanding Workshop*, pp. 263–274.

Boykov, Y., Veksler, O., and Zabih, R. 1998. Markov random fields with efficient approximations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–655.

Chen, L.-H. and Lin, W.-C. 1997. Visual surface segmentation from stereo. *Image and Vision Computing*, 15:95–106.

Cochran, S.D. and Medioni, G. 1992. 3-D surface description from binocular stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):981–994.

Cox, I.J., Hingorani, S.L., Rao, S.B., and Maggs, B.M. 1996. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567.

Faugeras, O. 1993. *Three-Dimensional Computer Vision*. MIT Press: Cambridge, MA.

Fua, P. 1991. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pp. 1292–1298.

Gamble, E.B., Geiger, D., Poggio, T., and Weinshall, D. 1989. Integration of vision modules and labeling of surface discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):1576–1581.

Gamble, E.B. and Poggio, T. 1987. Visual integration and detection of discontinuities: The key role of intensity edges. A.I. Memo No. 970, MIT-AI.

Geiger, D., Ladendorf, B., and Yuille, A. 1995. Occlusions and binocular stereo. *International Journal of Computer Vision*, 14(3):211–226.

Grimson, W.E.L. 1985. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):17–34.

Intille, S.S. and Bobick, A.F. 1994. Disparity-space images and large occlusion stereo. In *Proceedings of the 3rd European Conference on Computer Vision*, pp. 179–186.

Jones, D.G. and Malik, J. 1992. Computational framework for determining stereo correspondence from a set of linear spatial filters. *Image and Vision Computing*, 10(10):699–708.

Kanade, T. and Okutomi, M. 1994. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932.

Little, J.J. and Gillett, W.E. 1990. Direct evidence for occlusion in stereo and motion. *Image and Vision Computing*, 8(4):328–340.

Luo, A. and Burkhardt, H. 1995. An intensity-based cooperative bidirectional stereo matching with simultaneous detection of discontinuities and occlusions. *International Journal of Computer Vision*, 15(3):171–188.

Malik, J. and Perona, P. 1990. Finding boundaries in images. In *Proceedings of the 24th Asilomar Conference on Signals, Systems and Computers*, pp. 800–804.

Marr, D. and Poggio, T. 1976. Cooperative computation of stereo disparity. *Science*, 194:283–287.

Marr, D. and Poggio, T. 1979. A computational theory of human stereo vision. *Proceedings of the Royal Society of London, Series B*, 204:301–328.

Nalwa, V.S. 1993. *A Guided Tour of Computer Vision*. Addison-Wesley: Reading, MA.

Ohta, Y. and Kanade, T. 1985. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154.

Poggio, T., Gamble, E., and Little, J.J. 1988. Parallel integration of vision modules. *Science*, 242(4877):436–440.

Pollard, S.B., Mayhew, J.E.W., and Frisby, J.P. 1985. A stereo correspondence algorithm using a disparity gradient constraint. *Perception*, 14:449–470.

Spoerri, A. and Ullman, S. 1987. The early detection of motion boundaries. In *Proceedings of the 1st International Conference on Computer Vision*, pp. 209–218.

Toh, P.-S. and Forrest, A.K. 1990. Occlusion detection in early vision. In *Proceedings of the 3rd International Conference on Computer Vision*, Osaka, Japan, pp. 126–132.

Wang, J.Y.A. and Adelson, E.H. 1994. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638.

Wixson, L.E. 1993. Detecting occluding edges without computing dense correspondence. In *Proceedings of the DARPA Image Understanding Workshop*.