Depth Discontinuities by Pixel-to-Pixel Stereo

Stan Birchfield   Carlo Tomasi

Computer Science Department
Stanford University
Stanford, California 94305
[birchfield, tomasi]@cs.stanford.edu

This research was supported by the National Science Foundation under a Graduate Research Fellowship and under contract IRI-9506064, and by the Department of Defense under MURI contract DAAH04-96-1-0007 monitored by ARO and under a subcontract of STTR contract F49620-95-C-0078 monitored by AFOSR.

Abstract

This report describes a two-pass binocular stereo algorithm that is specifically geared towards the detection of depth discontinuities. In the first pass, introduced in part I of the report, stereo matching is performed independently on each epipolar pair for maximum efficiency. In the second pass, described in part II, disparity information is propagated between the scanlines.

Part I. Our stereo algorithm explicitly matches the pixels in the two images, leaving occluded pixels unpaired. Matching is based upon intensity alone without utilizing windows. Since the algorithm prefers piecewise constant disparity maps, it sacrifices depth accuracy for the sake of crisp boundaries, leading to precise localization of the depth discontinuities. Three features of the algorithm are worth noting: (1) unlike most stereo algorithms, it does not require texture throughout the images, making it useful in unmodified indoor settings, (2) it uses a measure of pixel dissimilarity that is provably insensitive to sampling, and (3) it prunes bad nodes during the search, resulting in a running time that is faster than that of standard dynamic programming.

Part II. After the scanlines are processed independently, the disparity map is postprocessed, leading to more accurate disparities and depth discontinuities. Both the algorithm and the postprocessor are fast, producing a dense disparity map in about 1.5 microseconds per pixel per disparity on a workstation. Results on five stereo pairs are given.

Contents

Part I

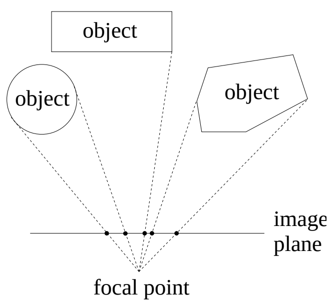
Processing the Scanlines Independently

Chapter 1

Introduction

Many powerful heuristics often fail at depth discontinuities: surfaces in the world are not continuous from the current viewpoint; the motion field often is not continuous; a window that straddles a depth discontinuity does not contain points from a single object; and scene points that are visible in one frame may not be visible in the next. So, on the one hand, depth discontinuities restrict the use of common assumptions and are therefore regarded by many image analysis methods as a hindrance. These methods see their performance degrade along depth discontinuities, where different disparities or image motions abut. Yet, from an image understanding point of view, depth discontinuities contain essential information about the scene: they contain information about the depth ordering of objects, are often strongly correlated to object shape, and facilitate figure-ground segregation. Therefore, many applications would benefit from a knowledge of the depth discontinuities, applications such as object tracking, interpolating between frames of an image sequence, merging different views of a scene, and planning camera motion to investigate areas of the scene which are not currently visible.

One appealing aspect of depth discontinuities is that they have a simple, purely geometric definition: A depth discontinuity is a point on the image plane whose projection ray, assuming a pinhole camera model, grazes a surface in the world. Thus, in the scene below, the black dots mark the depth discontinuities.



The goal of the approach described in this report is to detect depth discontinuities from a stereo pair of images. The distinctive flavor of our algorithm is that it is two-dimensional in nature. That is, it tries to locate the boundaries of objects (i.e., the depth discontinuities) rather than to recover an accurate depth map. To this end, it explicitly matches the pixels in the two images, preferring to assign a constant disparity to each object even if that object's depth is not constant. The pixels which are unmatched are assumed to be occluded, and the respective ends of the occluded regions are depth discontinuities. Localization of depth discontinuities is sharp because no windows are used for matching, but only individual pixels.

Because our algorithm establishes full correspondence, it is a stereo system in its own right, despite its specialized purpose. As such, it is similar to other stereo algorithms that incorporate depth discontinuities and occluded regions [2, 7, 6, 11, 8, 9]. Three aspects of the algorithm are worth noting. First, it uses a measure of pixel dissimilarity that looks at the linearly interpolated intensity functions surrounding the two pixels, and thus is provably insensitive to sampling effects. Sampling is an important phenomenon that greatly influences the values of the pixels that are near large variations in intensity. We have found that discrepancies between the sampling grids in the two images translate into intensity discrepancies as large as fifteen out of 256 gray levels (see Figure 3.7 on page 14). Our method handles these differences correctly. This is crucial in stereo matching, because it is precisely the pixels near intensity variations that have the most information. In contrast to the computationally expensive solution of matching at subpixel resolution or over windows, our dissimilarity measure solves the sampling problem with only a negligible increase in computation.

Most stereo algorithms require texture[1] throughout the image (but see [6], [8], and [1]), the assumption presumably being that untextured regions have no information for matching. However, images quite often contain large untextured regions, particularly in indoor settings. Our algorithm is able to handle these regions by making the simple but powerful assumption that depth discontinuities are always accompanied by intensity gradients. Based on this assumption, and on the observation that an intensity gradient between a near and a far object has the same disparity as the near object, the algorithm places depth discontinuities on the proper side of untextured regions. In our experience, this assumption is usually valid because our threshold for declaring intensity gradients is very low. In other words, our intensity gradients are not located just at strong intensity edges.

The final noteworthy point of the algorithm is that it prunes bad nodes during search, resulting in an average running time proportional to $n\Delta \log \Delta$ rather than the more common $n\Delta^2$ of standard dynamic programming. (n is the number of pixels in the scanline and Δ is the number of disparity levels.) On a high-end workstation, it takes about five seconds to process a pair of 630×480 images, using 15 disparity levels. (Note that the postprocessing described in

[1]By the term *texture*, we simply mean some variation in intensity.

Part II of this report takes an additional two seconds.)

We have strived for simplicity in our investigation. Thus the algorithm described here uses a simple cost function and processes each scanline independently. Not only has this simplicity facilitated our research by allowing us to isolate different phenomena, but the algorithm works surprisingly well with such little computational effort.

In the following chapter, we explicitly document the assumptions that the algorithm makes about the world, so that the reader can better evaluate the approach and the experimental results. Then, in Chaper 3, we formulate the correspondence problem, which is followed by descriptions of the cost function, the pixel dissimilarity measure, and the manner in which we handle untextured regions. In Chapter 4 we present our algorithm. In Chapter 5 we show the results of the algorithm on five pairs of images, and finally we conclude in Chapter 9.

Chapter 2

Assumptions

Every stereo algorithm makes assumptions about the world. Knowledge of these assumptions is important when evaluating an algorithm, because they nearly always affect the applicability of the algorithm. Below we provide an explicit list of our algorithm's assumptions:

A1. Each point in the scene looks identical in the two images. In other words, all surfaces are Lambertian, and the two image sensors are ideal samplers. The obvious violator is specular reflection. Another important problem arises with aliasing, which we alleviate by slightly defocusing the lens. Other sources of error are camera noise, lens distortion, lens blur (near a depth discontinuity where there is a nearby intensity gradient on the farther object), misalignment of the scanlines (where there is a large intensity gradient perpendicular to the scanlines), and different imaging parameters for the two cameras.

A2. In the absence of photometric information, the depth of the scene is piecewise constant in the direction of the scanlines. Although this is stronger than the usual assumption of piecewise continuity, some variation in depth is allowed, since the resolution of the disparity map is one pixel. Since our goal is to detect depth discontinuities rather than to compute an accurate depth map, we prefer to assign a constant disparity to each object, even if the depth of that object varies slightly (as in the case of a cylinder).

A3. At every depth discontinuity, there is an intensity gradient between the near object and the far object. This assumption (which is similar to the one made in [6]) allows us to prohibit depth discontinuities in regions of nearly constant intensity (i.e., untextured regions).

A4. Within each scanline, every object contains at least a modest amount of intensity variation. Therefore, we are not hindered by ambiguity in separating two adjacent objects. Because an intensity gradient between a near object and a far object belongs to the near object (as explained in Section

3.2.1), this assumption is automatically satisfied, by means of the previous assumption, for all objects except for those that are never adjacent to a farther object. We call these objects *background objects*. Therefore, this assumption is equivalent to assuming that background objects contain at least a modest amount of intensity variation.

A5. No scene point is viewed from both sides of another scene point. This is the monotonic ordering assumption.

A6. Each scene point is at least a certain distance from the camera. Because of this assumption, the search can be restricted to a small subset of all possible disparities.

A7. In any scanline in which an object is partially visible to one camera, that object is also partially visible to the other camera. Therefore, the depths of the objects can be determined by triangulation.

Just how restrictive are these assumptions? Overall, we have found A1 to be fairly reasonable (similar to the observation made in [2]), even though real-world objects are often not Lambertian [13]. Specular reflections cause little problem in our images since the baseline is small (but see [4] for one approach to handling them). Our experiments suggest that the assumption becomes less valid as the baseline is increased. To remove the effects of different imaging parameters, we obtained our stereo images by translating a single camera in a static scene; using two different cameras may require calibration. Also, we reduced distortion by using a lens with a long focal length. Surprisingly, A2 may be less restrictive than one might expect, because if a slanted object straddles two different disparities, the independent scanlines will place the discontinuities at somewhat random locations within the object; therefore the true discontinuity boundaries are still recognizable as coherent curves of depth discontinuities. (The experimental results of Figure 5.7 seem to justify this claim.) In our experience, A3 is not restrictive: an intensity gradient nearly always accompanies a depth discontinuity, because our threshold for declaring intensity gradients is low (see Section 3.2.1 for details). On the other hand, A4 is actually quite restrictive: although it allows for the untextured background walls which are prevalent in indoor settings, between any two foreground objects it requires the wall to have some intensity variation (such as a light switch or a door jamb). Propagating information between scanlines would generalize this assumption. Assumption A5 seems reasonable, since it is violated only when a thin object close to the viewer occludes a distant surface. In addition, it is worth noting that it can be proved to hold wherever the scene contains a unique surface of nonzero thickness [5]. Assumption A6 is easily enforced by maintaining a modest clearance in front of the cameras, and A7 is violated only in the presence of deep and narrow holes.

Chapter 3

Match Sequences

We pose the stereo correspondence problem as the problem of matching pixels in one scanline to pixels in the other scanline, assuming that the two cameras are aligned. Although our formulation could easily be extended to subpixel resolution, pixel resolution is sufficient to compute a rough depth map and to detect most depth discontinuities. Other pixel-resolution stereo algorithms are described in [7], [9], [10], and [8].

Correspondence is encoded in a *match sequence*, which is a sequence of matches. 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. 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*. We define *depth-discontinuity pixels* as those pixels that border a change in disparity and that lie on the far object. See Figure 3.1 for an illustration of these definitions on a pair of unusually short scanlines and Figure 3.2 for an actual match sequence found by the algorithm.

To formalize this framework, we let L and R denote the left and right scanlines, respectively, of a rectified stereo pair of images. $I_L(x)$ denotes the in-

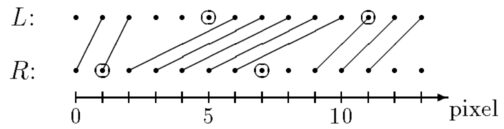


Figure 3.1: The match sequence $M = \langle(1,0),(2,1),(6,2),(7,3),(8,4),(9,5),(10,6),(11,9),(12,10),(13,11)\rangle$. The slope of a line is directly related to the disparity of its pixels; thus the five matches in the middle correspond to a near object. The left occlusions are $\langle 0\rangle$ and $\langle 3,4,5\rangle$, while the right occlusions are $\langle 7,8\rangle$ and $\langle 12,13\rangle$. The depth-discontinuity pixels are circled; note that the occlusions at the ends of the scanlines are caused by the borders of the image plane rather than by depth discontinuities.

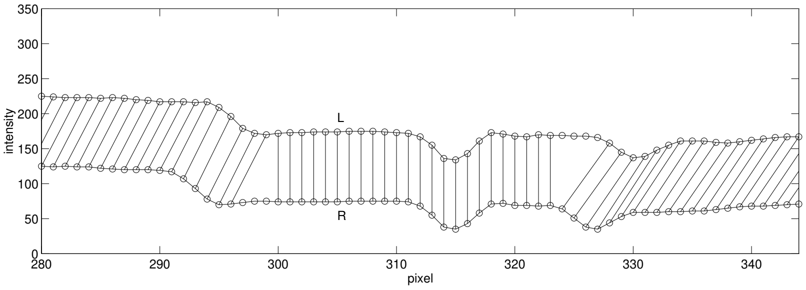


Figure 3.2: Part of the match sequence found by the algorithm on scanline 198 of the images in Figure 5.3. For viewing clarity, the left scanline (the top one) is shifted up, while the right scanline (the bottom one) is shifted to the right. In other words, the vertical axis is valid for R, while the horizontal axis is valid for L. The matches correspond, from left to right, to the Simulink box, the background wall, and the Clorox box.

tensity of pixel x in the left scanline, while $I_R(y)$ denotes the intensity of pixel y in the right scanline. We assume that each scanline is of length n, so that $x, y \in \{0, 1, \ldots, n-1\}$.

The *disparity* $\delta(x)$ of a pixel x in L that matches some pixel y in R 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 objects. In other words, if $\langle x, x+1, \ldots, x+k \rangle$ is a left occlusion,[1] then all of the pixels $x, x+1, \ldots, x+k$ are assigned a disparity of $\min\{\delta(x-1), \delta(x+k+1)\}$. Since disparity is inversely proportional to depth, at every change in disparity the side of the change with the smaller disparity belongs to the far object, while the side of the change with the larger disparity belongs to the near object. The depth-discontinuity pixels are labelled as those pixels that border a change in disparity and that lie on the far object. Thus, x is a depth-discontinuity pixel if either $\delta(x) < \delta(x-1)$ or $\delta(x) < \delta(x+1)$. It is worth noting that the right-most pixel in each left occlusion is a depth-discontinuity pixel, as well as the left-most pixel in each right occlusion (see Figure 3.1).

This definition of depth-discontinuity pixels does not allow for any disparity variation within an object. Thus, depth-discontinuity pixels will sometimes be declared where there are no actual depth discontinuities, such as within a slanted object. We show in Section 5, and in particular Figure 5.7, that the problems associated with this definition are not as severe as one might expect. And, since slanted objects often require a change in disparity of only one pixel, slightly modifying the above definition to require a disparity change of at least two pixels would alleviate this problem, at the expense of losing shallow depth discontinuities.

With each match sequence M we associate a cost $\gamma(M)$ that measures how

[1] A *left occlusion* is an occlusion in the left scanline, as shown in Figure 3.1.

unlikely it is that M describes the true correspondence. (That is, the best match sequence has the lowest cost.) In addition, we impose hard constraints to disallow certain types of unlikely match sequences. The cost function and hard constraints are the subjects of the next two sections.

3.1 Cost function

Instead of deriving a maximum a posteriori cost function from a Bayesian formulation (as is done in [2], [7], and [11]), we propose a simple cost function based mainly on intuition and justified solely by empirical evidence.

We define the cost γ of a match sequence M as a constant penalty for each occlusion, a constant reward for each match, and a sum of the dissimilarities between the matched pixels:

$$\gamma(M) = N_{occ}\kappa_{occ} - N_m\kappa_r + \sum_{i=1}^{N_m} d(x_i, y_i), \tag{3.1}$$

where κ_{occ} is the constant occlusion penalty, κ_r is the constant match reward, $d(x_i, y_i)$ is the dissimilarity between pixels x_i and y_i, and N_{occ} and N_m are the number of occlusions and matches, respectively, in M.

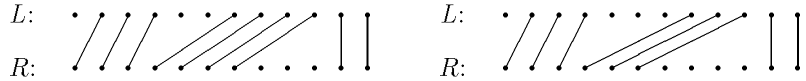
This cost function prefers piecewise-constant disparity maps. Thus, if possible, each object is assigned a single disparity. Although this behavior sacrifices accurate depth reconstruction, it is desirable for detecting depth discontinuities. In addition, the simplicity of Equation 3.1 makes our cost function easy to implement and to evaluate.

When an object is slanted, or otherwise straddles two disparities, more than one disparity is assigned to the object. This change in disparity is interpreted as an occlusion and hence a depth discontinuity. However, as we argued in the previous section, this error is not as severe as one might expect, because these false depth-discontinuity pixels will exhibit little coherence between adjacent scanlines (see Figure 5.7), and because these changes in disparity are often only one pixel, while true depth discontinuities usually cause changes of several pixels or more.

3.1.1 Occlusion penalty and match reward

Technically, κ_{occ} is interpreted as the amount of evidence (in terms of mismatched pixel intensities) that is necessary to declare a change in disparity, while κ_r is interpreted as the maximum amount of pixel dissimilarity that is generally expected between two matching pixels. Together, the two terms function like an occlusion penalty that is dependent on the length of the occlusion. (Figure 3.3 demonstrates the interaction of the two terms.) Nevertheless, we keep the terms separate because a constant occlusion penalty is central to our method of pruning the search space, as described in Chapter 4.2.

In our implementation, we set κ_{occ} to 25 and κ_r to 5, leading to an effective occlusion penalty function $\kappa_{eff}(l)$ as shown in Figure 3.4. As long as we are



(a) (b)

Figure 3.3: The match reward causes occlusions to be penalized based on their lengths. In fact, although the number of occlusions is the same (three) in both (a) and (b), the lengths of the occlusions affect the number of matches. Assuming that all of the dissimilarities between matched pixels are zero, the cost function evaluates to: (a) $3\kappa_{occ} - 9\kappa_r$ and (b) $3\kappa_{occ} - 8\kappa_r$.

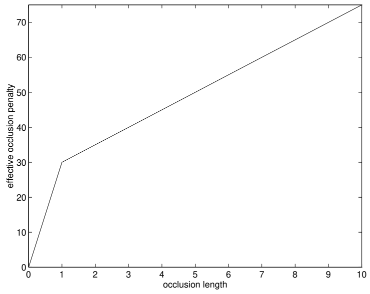


Figure 3.4: The effective penalty κ_{eff} of an occlusion vs. the length of the occlusion. κ_{eff} is a combination of κ_{occ} and κ_r.

using Equation 3.1, this function must satisfy two conditions. First, the function must increase with the length of an occlusion, or else the matching algorithm will skip over regions of the images in which the intensities of matching pixels differ significantly due to noise or other phenomena. In other words, if a large occlusion has a relatively small penalty, then the algorithm will declare a large occlusion in the left (or right) scanline followed by a single match followed by a large occlusion in the right (or left) scanline to avoid matching the pixels affected by noise. Other researchers have encountered difficulty in using a cost function that adds only a constant occlusion penalty to pixel dissimilarities [8]. Second, as explained in [7], the function must satisfy $\kappa_{eff}(l_1) + \kappa_{eff}(l_2) \geq \kappa_{eff}(l_1 + l_2)$ for all lengths l_1 and l_2. Otherwise, the algorithm would prefer to declare two small occlusions instead of one large occlusion, leading to a staircase-like disparity function. Making the left-hand side of this inequality significantly greater than the right-hand side causes each object to be assigned a constant disparity, even if the object's depth is not constant (as in the case of a cylinder). The result is a piecewise-constant disparity map.

3.1.2 Pixel dissimilarity

The term $d(x_i, y_i)$ measures how unlikely it is that $I_L(x_i)$ and $I_R(y_i)$ are images of the same scene point. This dissimilarity cannot be measured by simply taking

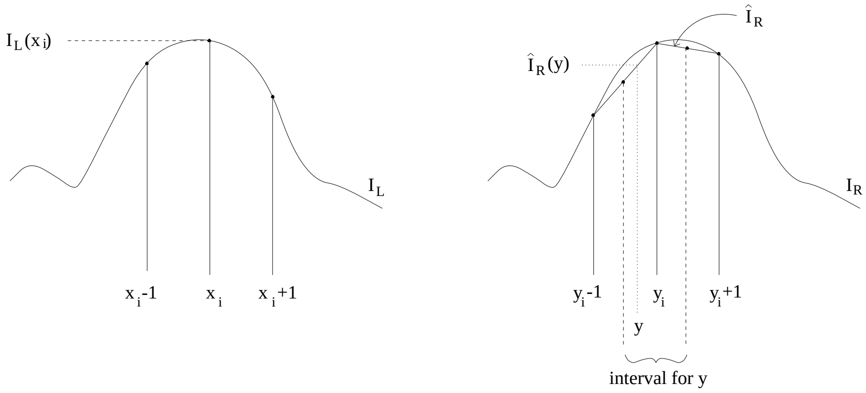


Figure 3.5: Definition of $\bar{d}(x_i, y_i, I_L, I_R)$.

the difference between $I_L(x_i)$ and $I_R(y_i)$, as is often done, because these intensities are affected by image sensor sampling. In fact, this difference in intensities varies depending upon how much the intensity of the image is changing in the vicinity. We know of no one who has explicitly addressed this problem. Typically, the problem is alleviated either by working at subpixel resolution [2, 11] or by adding robustness through window-based matching [7, 9, 6]. Instead, we propose to use the linearly interpolated intensity functions surrounding two pixels to measure their dissimilarity, in a method that is provably insensitive to sampling. For this method to work well, the intensity functions must vary predictably between pixels; therefore, we slightly defocus the lens to remove aliasing.

More precisely, as shown in Figure 3.5, we first define the minimum difference $\bar{d}$ between $I_L(x_i)$ and $\hat{I}_R(y)$, where $\hat{I}_R$ is the function obtained by linearly interpolating I_R and where $|y - y_i| \leq \frac{1}{2}$:

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

Then we define the minimum difference in the other direction, and we take the minimum of the two: $d(x_i, y_i) = \min\{\bar{d}(x_i, y_i, I_L, I_R),\ \bar{d}(y_i, x_i, I_R, I_L)\}$. 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. See Figure 3.6. 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,

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

This computation takes only a constant amount of time more than taking the

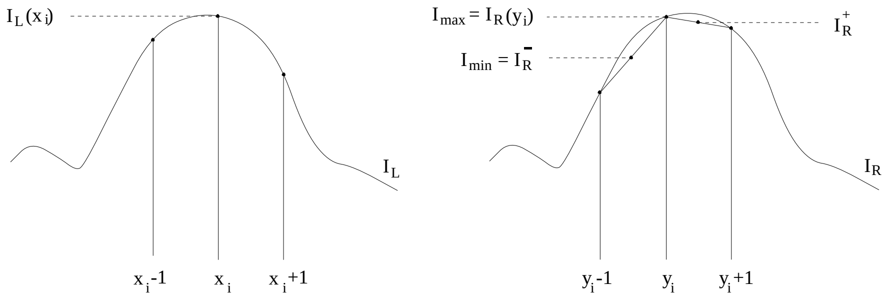


Figure 3.6: Computation of $\bar{d}(x_i, y_i, I_L, I_R)$.

absolute difference in intensities, yet it saves us the expensive task of having to compute matches at subpixel resolution or within windows.

It is shown in Appendix A that 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. As mentioned above, we slightly defocus the lens to ensure this condition.

Figure 3.7 contrasts our dissimilarity measure with the absolute difference in intensity. (We will call the latter method SAD, although there is no sum involved.) Wherever the intensity function is nearly constant, or wherever the disparity between the two scanlines is close to an integer number of pixels, the two approaches yield similar results, since sampling effects are negligible. However, where the intensity function changes rapidly and the disparity is not an integer number of pixels, the SAD between two matching pixels is large, while our dissimilarity measure remains well-behaved. Close examination of the pixel intensities verifies that the erratic plot of Figure 3.7(c) is due to sampling rather than the choice of matched pixels.

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 disallow certain types of unlikely match sequences, and (2) they facilitate a systematic search of the space of possible match sequences. Of the seven constraints described below, they all serve the first purpose, but only the last five serve the second purpose.

In the following treatment, we assume that we have a match sequence $M =$

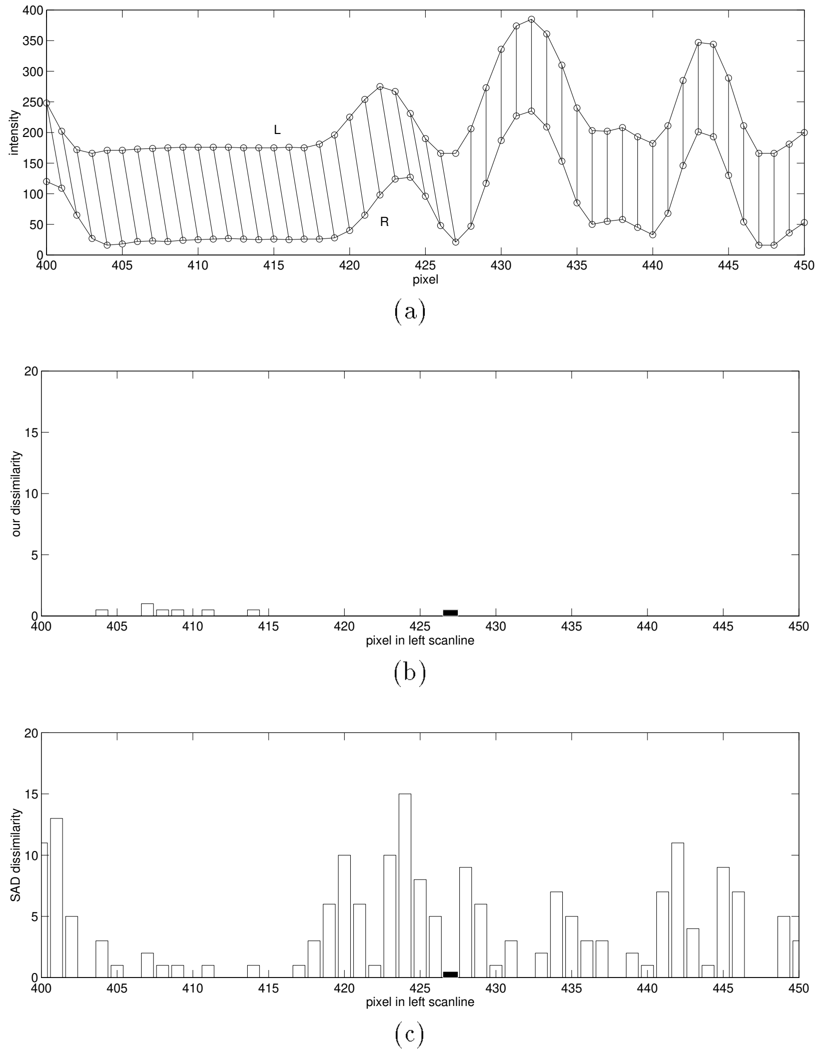


Figure 3.7: (a) A portion of the match sequence computed for scanline 240 of the images in Figure 5.3. For viewing clarity, the left scanline is shifted up, while the right scanline is shifted to the right. (b) The dissimilarities between the matched pixels, as computed by our measure. Most of the values are zero. The black rectangle indicates that the dissimilarity of the unmatched pixel 427 is undefined. (c) The dissimilarities computed by taking the absolute value of the difference (SAD) in intensity.

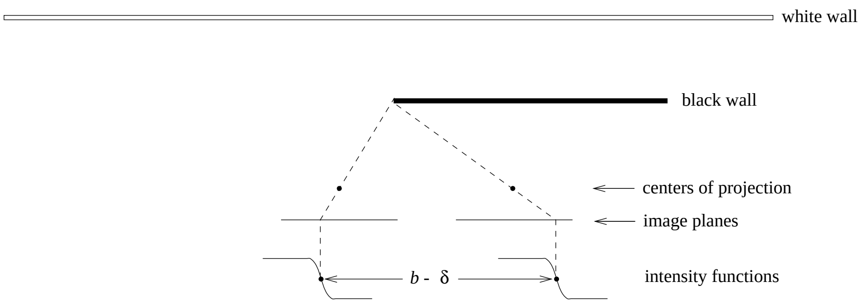


Figure 3.8: An intensity gradient due to a depth discontinuity has the same disparity as the near object. In this example, the near object is an untextured black wall, while the far object is an untextured white wall. Together, the objects produce a step edge in each image. The disparity δ between the step edges is identical to the disparity of any point on the black wall. (b is the baseline.)

$\langle(x_1, y_1), \ldots, (x_k, y_k)\rangle$, where $k = N_m$ is the number of matches in M.

3.2.1 Intensity gradients accompany depth discontinuities

Assume, as we did in Chapter 2, that every depth discontinuity is accompanied by an intensity gradient. Such a gradient will have the same disparity as the near object and therefore will be considered as part of the near object. (To be convinced of this important but non-obvious principle, consult Figure 3.8. An argument from real images is presented in Chapter 5, item A3.) An occlusion in the left scanline occurs because the near object is to the right of the far object. Since the intensity gradient is part of the near object, and since the occluded pixels come from the far object, the occlusion must lie immediately to the left of the intensity gradient. Likewise, a right occlusion must lie immediately to the right of an intensity gradient. Therefore, we impose the following two constraints:

C1. if $\langle x_i, \ldots, x_j\rangle$ is a left occlusion, then x_j lies to the left of an intensity gradient, $\quad 1 \leq i \leq j < k$

C2. if $\langle y_i, \ldots, y_j\rangle$ is a right occlusion, then y_i lies to the right of an intensity gradient, $\quad 1 < i \leq j \leq k$

Examples showing various allowed and non-allowed placements of an occlusion relative to its intensity gradient are shown in Figure 3.9.

A pixel x in the left scanline is said to lie to the left of an intensity gradient if the intensity variation between $x+1$, $x+2$, and $x+3$ is at least 5 gray levels. Likewise, a pixel y in the right scanline is said to lie to the right of an intensity gradient if the intensity variation between $y-1$, $y-2$, and $y-3$ is at least 5 gray levels. The size of the window is chosen to make the computation fairly

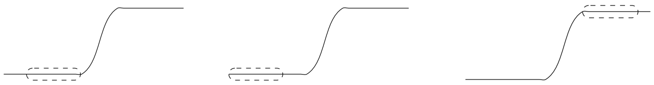


Figure 3.9: An occlusion in the left image must lie to the left of an intensity gradient. (a) Correct placement of occlusion. (b) Occlusion is too far from gradient. (c) Occlusion has no gradient to its right.

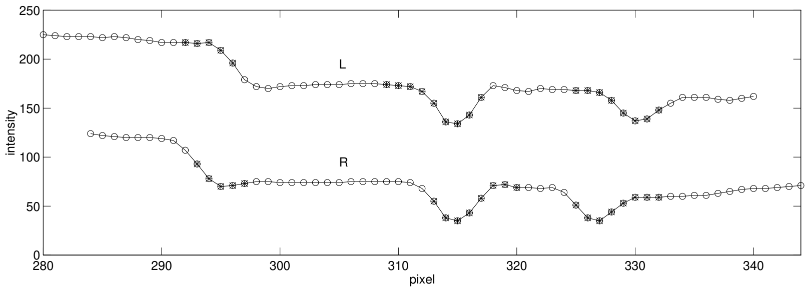


Figure 3.10: Darkened pixels lie (a) to the left and (b) to the right of intensity gradients. The scanlines are from Figure 3.2.

robust to noise while still being localized. The threshold of 5 is chosen to be just above the intensity variation in untextured regions for our camera. Since noise depends on intensity, calibrating the camera would yield a more accurate threshold. As an example of the output of this computation, Figure 3.10 shows the pixels which are determined to lie next to intensity gradients.

It is important to note that we are not trying to place depth discontinuities along strong intensity "edges", since our threshold for image intensity variation is so small. Rather, we are merely preventing the cost function from making a poor decision in a region where there is little information.

3.2.2 Constraints related to search

C3. $0 \leq x_i - y_i \leq \Delta, \quad i = 1, \ldots, k$

C4. $y_1 = 0$

C5. $x_k = n - 1$

C6. $x_i < x_j$, and
$y_i < y_j, \quad 1 \leq i < j \leq k$

C7. $x_{i+1} = x_i + 1, \quad \text{or}$
$y_{i+1} = y_i + 1, \quad i = 1, \ldots, k-1$

To greatly reduce the size of the search space, C3 introduces Δ as the maximum allowed disparity. This constraint is justified by the assumption that all objects are at least a certain depth from the cameras (Assumption A6 of Chapter 2); because the cameras are rectified, it also requires that the disparity be nonnegative. The next two constraints force the left-most pixel in the right scanline and the right-most pixel in the left scanline to be matched; this is due to a technical requirement of dynamic programming. Since the number of matched pixels far outweighs the number of occluded pixels in a typical image, it is reasonable to assume that these two pixels should be matched.

The final two constraints enable the use of dynamic programming to traverse the entire search space (although C7 is slightly more restrictive than necessary). While C6 follows naturally from the monotonic ordering assumption (A5), the justification of C7 may not be obvious at first glance. Still, it is not hard to show that it also follows from the assumptions of Chapter 2. Suppose that we have a match (x_i, y_i) which corresponds to a point on an object in the scene. Three situations are possible:

1. Neither x_i nor y_i is near a depth discontinuity. In this case, (x_{i+1}, y_{i+1}) will be a match corresponding to a point on the same object. So C7 is valid.

2. The pixel x_i is near a depth discontinuity in scanline L. This situation is symmetric to the following one and should be clear from the explanation there.

3. The pixel y_i is near a depth discontinuity in scanline R. To see that Constraint C7 is valid in this case, refer to Figure 3.11 which shows the projection rays from the two cameras landing on the two objects in the scene; these objects are piecewise constant in depth due to Assumption A2. In the drawing, the vertical line connecting the two planes at different depths signifies that no projection ray crossing that line can give rise to a match, due to the monotonic ordering assumption. If the projection rays for x_i and y_i meet on some object in the world, then the projection ray for y_{i+1} must terminate at some point p on the next object; it can't skip an object or else the mutual visibility assumption (A7) would be violated. It is clear from the monotonic ordering assumption that there can be no obstruction between p and the left focal point. Therefore, there must be some $j > i$ such that (x_j, y_{i+1}) is a match.

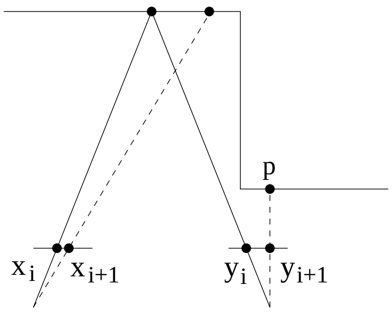


Figure 3.11: Justification of Constraint C7.

Chapter 4

Searching for the Optimal Match Sequence

Thanks to the structure of the cost function (Equation 3.1), the technique of dynamic programming (also used in [1], [12], [2], and [7]), can be used to find the optimal match sequence by searching over all the match sequences that satisfy the constraints given in Section 3.2.

Figure 4.1a illustrates the search grid for $n = 10$ and $\Delta = 3$. (Recall that n is the number of pixels in each scanline and Δ is the maximum disparity allowed.) Because of constraint C3, many of the cells in the grid are disallowed; these are shown as black cells. The algorithm searches for the best possible path[1] stretching from the left-hand side to the right-hand side. 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 $\times$. Notice that any column or row that does not contain an $\times$ corresponds

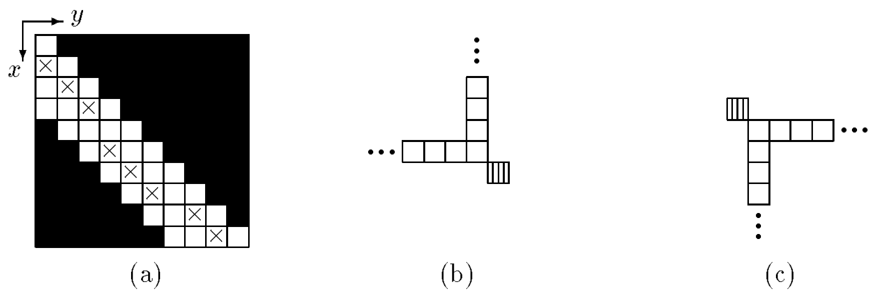
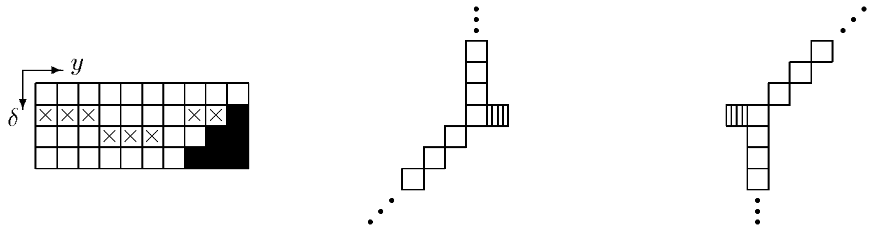


Figure 4.1: (a) The search grid and a match sequence ("$\times$" cells). x indexes the left scanline, while y indexes the right scanline. (b) The matches (white cells) that can immediately precede a match (striped cell), when the bounds on disparity are ignored. (c) The matches that can immediately follow a match.

[1] Informally, we will use the terms *cell* and *match* interchangeably, as well as the terms *path* and *match sequence*.



(a) (b) (c)

Figure 4.2: (a) The shifted search grid. (b) The immediately preceding matches (ignoring disparity bounds). (c) The immediately following matches.

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 Figure 4.1b, due to constraints C6 and C7. Similarly, the matches which can possibly be chosen as its immediately following match (x_{i+1}, y_{i+1}) are the cells shown in Figure 4.1c. Placing Figure 4.1b and Figure 4.1c onto the grid of Figure 4.1a 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 boundaries).

In order to make the diagram in Figure 4.1 more compact, we shift each column up by an amount equal to the number of the column, which leads to the grid in Figure 4.2. The vertical axis is now $\delta = x - y$, the disparity. In a similar manner, Figures 4.1b and 4.1c become Figures 4.2b and 4.2c.

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. Two additional arrays are used: $g_L[x]$ is `TRUE` if the pixel x in the left scanline lies to the left of an intensity gradient, and is `FALSE` otherwise; similarly, $g_R[y]$ is `TRUE` if the pixel y in the right scanline lies to the right of an intensity gradient, and is `FALSE` otherwise.

4.1 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 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 4.2. Both algorithms traverse the φ array from left to right, and from top to bottom, computing the cost of the best path to each cell:[2]

$$
\begin{aligned}
\varphi[\delta, y] &= \gamma_0(y+\delta, y) \\
&= d(y+\delta, y) - \kappa_r \qquad\qquad (4.1) \\
&\quad + \min \left\{ \begin{array}{l} \varphi[\delta, y-1], \\ \varphi[\delta-1, y-1] + \kappa_{occ}, \ldots, \varphi[0, y-1] + \kappa_{occ}, \\ \varphi[\delta+1, y-2] + \kappa_{occ}, \ldots, \varphi[\Delta, y+\delta-\Delta-1] + \kappa_{occ} \end{array} \right\},
\end{aligned}
$$

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 φ array is filled, the lowest-cost cell which satisfies constraint C5 is selected as the ending match m_k. Then, starting at this cell, π is traced to find the optimal match sequence.

4.1.1 Backward-Looking Algorithm

The Backward-Looking Algorithm, shown in Figure 4.3, iterates through all the cells $[\delta, y]$ of the shifted search grid, computing Equation 4.1 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.

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 (Figure 4.2b). Line `8` prevents φ' from being updated if the match being considered would cause a depth discontinuity without an intensity gradient. 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 an intensity gradient; 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 an intensity gradient. Line `9` updates φ' by adding κ_{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 φ and π arrays.

4.1.2 Forward-Looking Algorithm

The Forward-Looking Algorithm, shown in Figure 4.4, splits the minimization of Equation 4.1 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

[2] We introduce the notation $\gamma_0(x, y)$ to refer to the cost of the best match sequence whose ending match is (x, y). The ending match of a match sequence $\langle (x_1, y_1), \ldots, (x_k, y_k) \rangle$ is defined as (x_k, y_k).

```
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          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 g_L[y + δ − 1])
                            or (δ < δ_p and g_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 4.3: Backward-Looking Algorithm.

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.

Lines `1-5` initialize the array. The equation in line `9` expresses the position of the possible following matches (Figure 4.2c). The variable φ' 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.

4.1.3 Comparison

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 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

```
Forward-Looking Algorithm
for δ ← 0 to Δ
    φ[δ, 0] ← d(δ, 0)
for y ← 1 to n − 1
    for δ ← 0 to Δ
        φ[δ, y] ← ∞
for y_p ← 0 to n − 2
    for δ_p ← 0 to Δ
        for δ ← 0 to Δ
            y ← y_p + max(1, δ_p − δ + 1)
            if (δ = δ_p) or (δ > δ_p and g_L[y + δ − 1])
                         or (δ < δ_p and g_R[y_p + 1]) then
                φ' ← φ[δ_p, y_p] + d(y + δ, y) − κ_r + κ_occ ∗ (δ ≠ δ_p)
                if φ' < φ[δ, y] then
                    φ[δ, y] ← φ'
                    π[δ, y] ← [δ_p, y_p]
```

Figure 4.4: Forward-Looking Algorithm.

next section.

4.2 A Faster Algorithm

In the interest of optimality, both of the algorithms described in Section 4.1 perform a great deal of unnecessary computation because they compute the best paths to all the cells, even 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.[3]

Consider a match p with a possible following match c such that there is a right occlusion between them, as shown in Figure 4.5 with respect to the original search grid of Figure 4.1. 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 Figure 4.1c), 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.

[3] Our experiments show that the resulting disparity maps disagree in fewer than 0.7% of their values, for Δ ranging from 14 to 40 pixels.

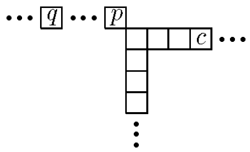


Figure 4.5: Optimality is retained when p is not rightward expanded, assuming that $\gamma_0(q) < \gamma_0(p)$.

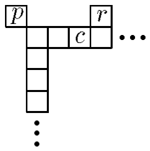


Figure 4.6: Optimality is lost when p is not rightward expanded, assuming that $\gamma_0(r) < \gamma_0(p)$.

In light of these observations we could, without sacrificing optimality, modify the Forward-Looking Algorithm so that it refuses to rightward expand any match with a lower-cost match to its left and refuses to downward expand any match with a lower-cost match above it. However, the running time would not be reduced because of the difficulty in determining whether there is a lower-cost match above or to the left of another match. Instead, we modify the algorithm so that it refuses to rightward expand any match with a lower-cost match in its row and refuses to downward expand 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 Figure 4.6, 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 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., $\varphi[r] < \varphi[p]$.[4] (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 rightward expand p, 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. Thus optimality is lost. However, it is not surprising that the loss of optimality is small, given the large number of assumptions that we had to make.

The average running time of the Faster Algorithm is estimated empirically to be proportional to $n\Delta \log \Delta$, as explained in Section 5.2. Justifying this ex-

[4] The meaning here should be clear, despite the abuse of notation.

pression analytically would require making assumptions about the distribution of the elements in the φ table.

The Faster Algorithm, shown in Figure 4.7, 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 φ 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 φ 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 an intensity gradient. 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 an intensity gradient, 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 $x_p = y_p + \delta_p$.

```
Faster Algorithm
for $\delta \leftarrow 0$ to $\Delta$
    $\varphi[\delta, 0] \leftarrow d(\delta, 0)$
for $y \leftarrow 1$ to $n-1$
    for $\delta \leftarrow 0$ to $\Delta$
        $\varphi[\delta, y] \leftarrow \infty$
for $x \leftarrow 0$ to $n-1$
    $m_x[x] \leftarrow \infty$
for $y_p \leftarrow 0$ to $n-2$
    $m_y \leftarrow \min\{\varphi[0, y_p], \varphi[1, y_p], \ldots, \varphi[\Delta, y_p]\}$
    for $\delta_p \leftarrow 0$ to $\Delta$
        update($\delta_p, y_p, \delta_p, y_p + 1$)
        if $\varphi[\delta_p, y_p] \leq m_y$ then
            $y \leftarrow y_p + 1$
            for $\delta \leftarrow \delta_p + 1$ to $\Delta$
                if $g_L[y + \delta - 1]$ then update($\delta_p, y_p, \delta, y$)
        if $\varphi[\delta_p, y_p] \leq m_x[\delta_p + y_p]$ and $g_R[y_p + 1]$ then
            for $\delta \leftarrow 0$ to $\delta_p - 1$
                $y \leftarrow y_p + \delta_p - \delta + 1$
                update($\delta_p, y_p, \delta, y$)

update($\delta_p, y_p, \delta, y$)
$\varphi' \leftarrow \varphi[\delta_p, y_p] + d(y + \delta, y) - \kappa_r + \kappa_{occ} * (\delta \neq \delta_p)$
if $\varphi' < \varphi[\delta, y]$ then
    $\varphi[\delta, y] \leftarrow \varphi'$
    $\pi[\delta, y] \leftarrow [\delta_p, y_p]$
    $m_x[y + \delta] \leftarrow \min\{m_x[y + \delta], \varphi'\}$
```

Figure 4.7: Faster Algorithm.

Chapter 5

Experimental Results

In this section we present the results of the Faster Algorithm on five different stereo image pairs. Our goal is to ascertain the algorithm's strengths and weaknesses by closely examining the validity of its output.

Figures 5.3 through 5.7 show five stereo image pairs, displayed for cross-eyed fusion, and the resulting disparity map and depth discontinuities of each. The latter two are displayed with respect to the left image. No postprocessing was performed on the output of the algorithm.

The output on these images demonstrates the effectiveness of the algorithm at reconstructing a piecewise disparity map and, more importantly, accurate depth discontinuities. The algorithm correctly places the depth discontinuities along the object contours, even when the background wall has roughly constant intensity. In particular, notice that near the bottom center of Figure 5.4(d) the depth discontinuities are placed along the edges of the table support rather than along the edge of the door, even though there is no texture on either the support or the door. Similarly, in Figure 5.6(d) the depth discontinuities are placed along the table support even though the only real intensity variation on the background is near the left and right ends of the scanlines.

One immediately obvious limitation of the algorithm is the fact that it processes the scanlines independently, which causes the horizontal streaks in the disparity map. The other modes of failure result from breakdowns in the assumptions of Chapter 2. We will examine each assumption in turn:

A1. *Each point looks identical in the two images.* To help ensure the validity of this assumption, we carefully took our stereo images by using a single camera and translating roughly in the direction of the scanlines. In addition, we slightly defocused the lens to eliminate aliasing.

The breakdown in this assumption can be readily seen in Figure 5.3, where specular reflections occur on the wine bottle, on the right Clorox bottle, and on the shiny surface of the table. Since the camera translated a small amount between frames, the disparities of the reflections are almost the same (within one pixel) as the disparities of the corresponding objects.

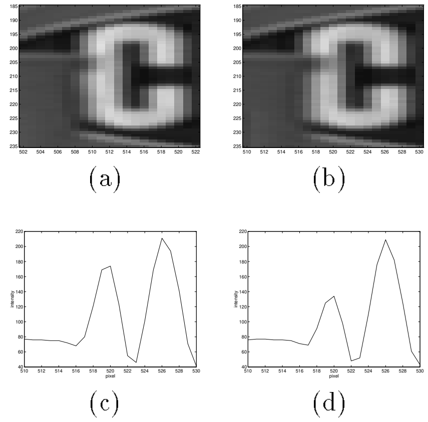


Figure 5.1: The blending of intensities across depth discontinuities. (a) Right image. (b) Left image. (c) and (d) Intensity plots of a scanline pair (row 205) containing the lid of the Splash can. Notice that the two smaller peaks differ from each other by 40 gray levels.

Therefore, the algorithm is able to assign a constant disparity to most objects in spite of the reflections.

Another breakdown occurs as a byproduct of defocusing the lens. Shown in Figure 5.1 is a blowup of the area surrounding the top right corner of the Splash can of Figure 5.3, which partially occludes the "C" on the right Clorox bottle. Defocusing the lens mixes the intensities of the Splash can lid and the "C" so that the intensities of the "C" are much higher in the right image than in the left, in the rows containing the lid. This type of situation, however, requires there to be a large intensity variation on the far object near a depth discontinuity and is therefore rare in our images.

Even when the image has little intensity variation, the dissimilarities between the pixels can sometimes guide the search to the correct disparity. For example, the bottom fifty scanlines of Figure 5.7 contain almost no intensity variation, and yet their disparities are nearly correct. Sometimes, as in the case of the region to the right of the lamp in Figure 5.4, the disparity is correctly changed from the object (in this case the lamp) to a lower value, although that value itself is incorrect. However, as is seen in the four streaks near the top of this same figure, sometimes the sum of the dissimilarities of the pixels over a large untextured region is not a minimum at the correct disparity, and is even large enough to declare an occlusion.

A2. *The depth of the scene is piecewise constant.* In the first four images, this assumption holds because the minor variation in depth within each object is below the resolution of the disparity map. One exception occurs on the door of Figure 5.5. The left door edge between the knobs is found to be a depth discontinuity, when in fact it is a surface-normal discontinuity. It could be argued that the algorithm's decision is not a complete failure, however, because a small amount of camera motion in the proper direction would make this edge a depth discontinuity. Thus, in some sense, the edge is "almost" a depth discontinuity.

Even when the assumption is intentionally violated, the algorithm performs surprisingly well. For example, in Figure 5.7, even though the Clorox box varies considerably in depth, the algorithm assigns a constant disparity to the object (in most of the scanlines) and therefore correctly detects many of the depth discontinuities along both of its edges. The algorithm also works well on the Graebel box, which straddles two disparities. Although depth discontinuities are falsely declared within the box, they are declared independently for each scanline and therefore exhibit no coherence between scanlines. (For a more principled approach to handling sloping surfaces, see [3].)

A3. *Intensity gradients accompany depth discontinuities.* In general, this assumption holds. One exception occurs in Figure 5.3 along the right edge of the left Clorox bottle, just above the lettering. The algorithm assumes that the triangular wedge belongs to the Simulink box instead. Another exception occurs along the left side of the cap of the spray paint can.

The power of this assumption is seen in Figure 5.4, where a reasonable disparity map is constructed from a pair of images that has sparse intensity variation. To see the reasoning of the algorithm, look at scanline 250. The edges of the lamp are easily determined to be one disparity, while the edge of the door (the dark vertical stripe) is easily determined to be a different disparity. Because of our assumption, we know that a depth discontinuity cannot occur along the bland wall between them. So either the door edge or the lamp edge is a depth discontinuity, which means that the wall is either part of the door or part of the lamp. But the wall cannot be part of the lamp, because if it were, then the lamp would extend to the edge of the door, which would cause the door edge to be at the same disparity as the lamp. Since the door edge is at a smaller disparity (implying larger depth), the bland wall must belong to the door. Therefore, the edge of the lamp is a depth discontinuity. So we see that high-level, powerful reasoning is achieved by a simple, local test for intensity variation.

A4. *Every object contains intensity variation.* This is an important assumption. It explains many of the algorithm's successes and failures on these images, since the background wall has little intensity variation. For example, consider the black table support near the bottom center of Figure 5.4(a). The depth discontinuities along the left edge of the support are correctly found, due to the dark vertical stripe caused by the left edge of the door. In contrast, the depth discontinuities along the right edge are correctly found only along the scanlines which contain the door hinge; along the other scanlines there is not enough intensity variation on the wall for the algorithm to correctly detect the disparity of the wall, and hence it cannot detect the depth discontinuities. As another example, the cavity of the lamp (where the lamp head meets the neck) in Figure 5.4 is not found because the region of the wall inside has nearly constant intensity. More strikingly, the middle Clorox bottle entirely disappears from the disparity map of Figure 5.6 because there is not enough intensity variation on the wall for the algorithm to realize that there is a wall between the three objects. Finally, notice in Figure 5.3(a) that the dark vertical stripe caused by the edge of the door is conveniently placed between the Simulink box and the Clorox box, with the result that the depth discontinuities along the edges of the two objects are correctly detected.

A5. *The monotonic ordering assumption.* In these images, this assumption is always valid.

A6. *The true disparity of any pixel is not larger than Δ.* After taking the images, we chose Δ so that this assumption would hold.

A7. *Every object can be seen by both cameras.* In these images (and in fact in most real images), this assumption is always valid.

5.1 Sensitivity to Parameters

We have found our algorithm to be insensitive to the choice of Δ, as long as it is larger than the actual maximum disparity. As Δ varies from 14 to 50, almost none (fewer than 0.3%) of the pixels in the disparity map change value. Of course, if Δ is below the actual maximum disparity (which in these images ranges from 8 to 13), then the algorithm's performance degrades substantially.

On the other hand, we have found the algorithm to be moderately sensitive to the occlusion penalty κ_{occ} and the match reward κ_r. As mentioned in Section 3.1, we empirically chose $\kappa_{occ} = 25$ and $\kappa_r = 5$ for all of the results presented in this report. Fewer than 5% of the pixels in the disparity map change as κ_{occ} varies from 18 to 35, and fewer than 1.5% of the pixels change as κ_r varies from 3 to 8. These results were obtained from the images of Figures 5.3 and 5.4, with Δ set to 14. If κ_r is less than 3, then performance degrades as Δ increases because the algorithm is allowed to declare large occlusions without incurring large penalties.

5.2 Computing Time

The algorithm was implemented in C and compiled using `cc` with maximum optimization. For the results presented in this report, we set Δ to 14, which yields a computing time of about nine seconds on these 630×480 images using a Silicon Graphics Indy workstation, which is an average machine. On a Silicon Graphics Indigo 2 Extreme workstation, which is a faster machine, the algorithm takes about five seconds.

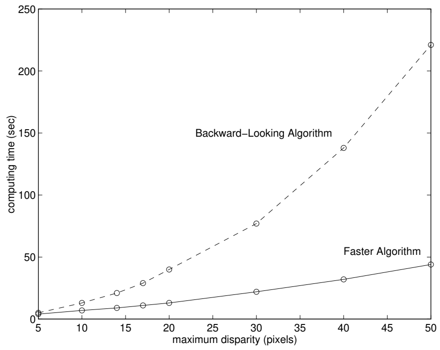


Figure 5.2: Computing time vs. Δ of the Backward-Looking (dashed) and Faster (solid) Algorithms, on a Silicon Graphics Indy workstation.

Of course, the computing time depends on the value of Δ. Figure 5.2 demonstrates that our algorithm is significantly faster than the standard dynamic programming algorithm, especially for large Δ. From the data in the figure, the running time of our algorithm seems to be proportional to $n\Delta \log \Delta$.

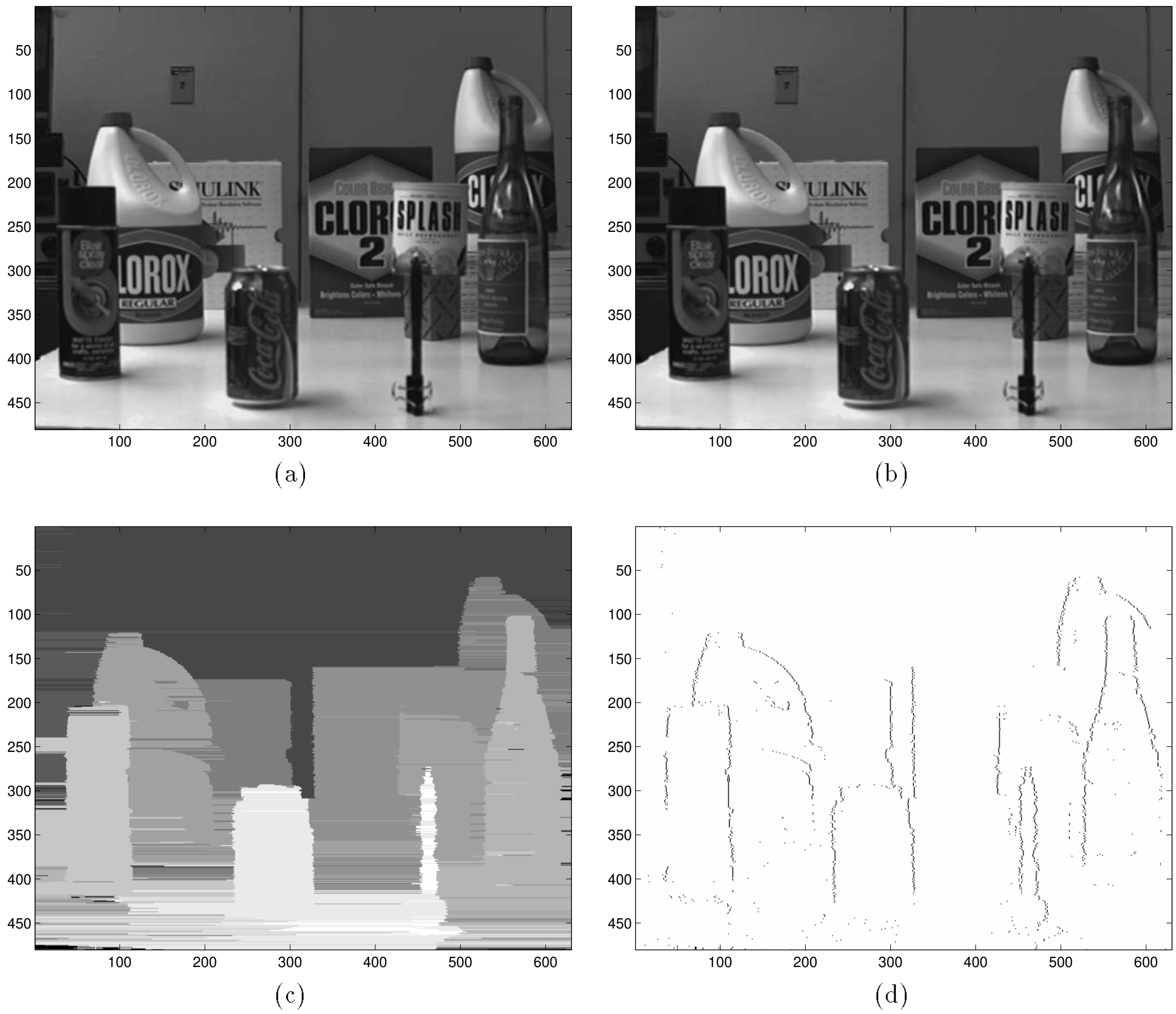


Figure 5.3: (a) Right image. (b) Left image. (c) Disparity map. (d) Depth discontinuities.

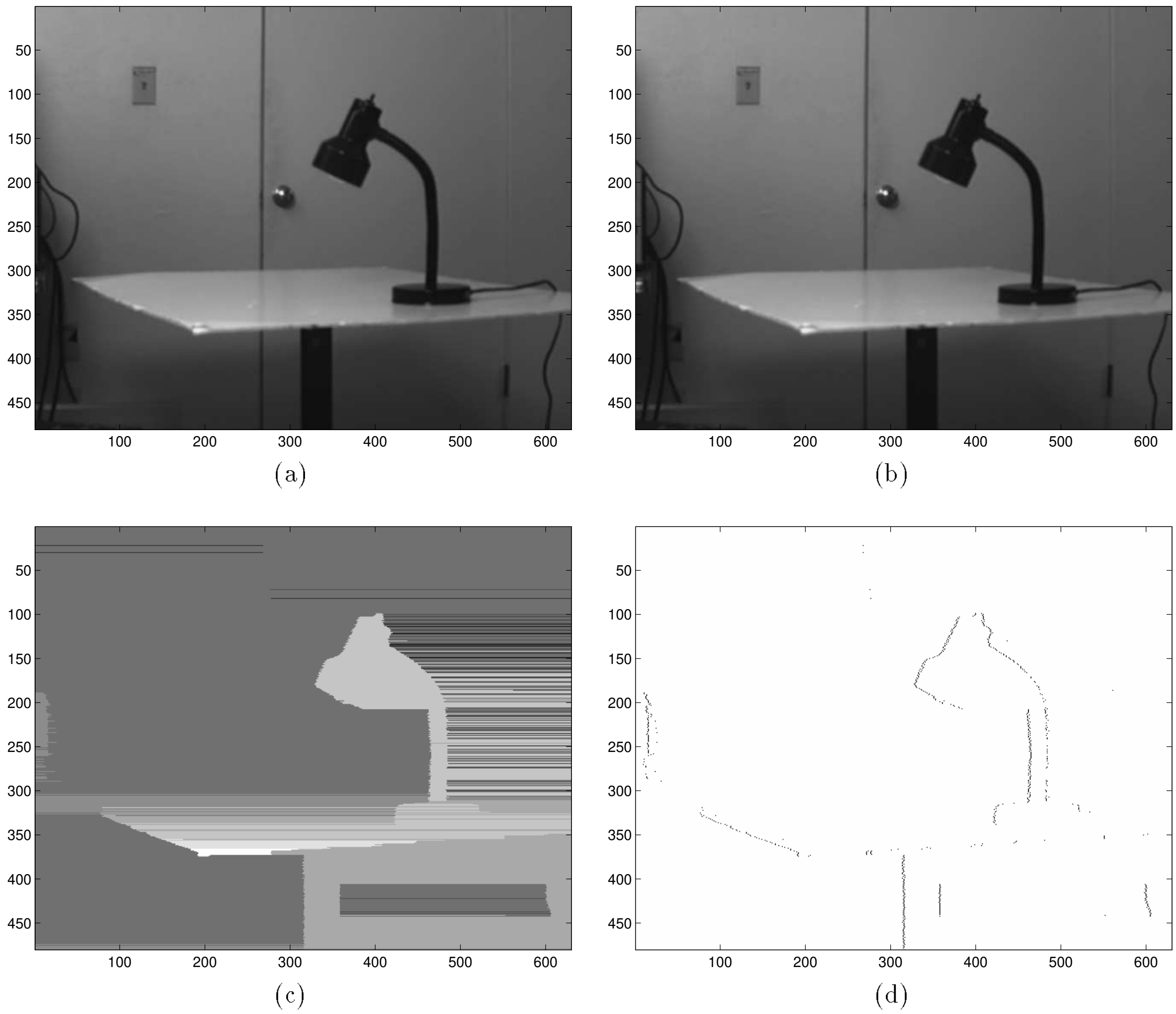


Figure 5.4: (a) Right image. (b) Left image. (c) Disparity map. (d) Depth discontinuities.

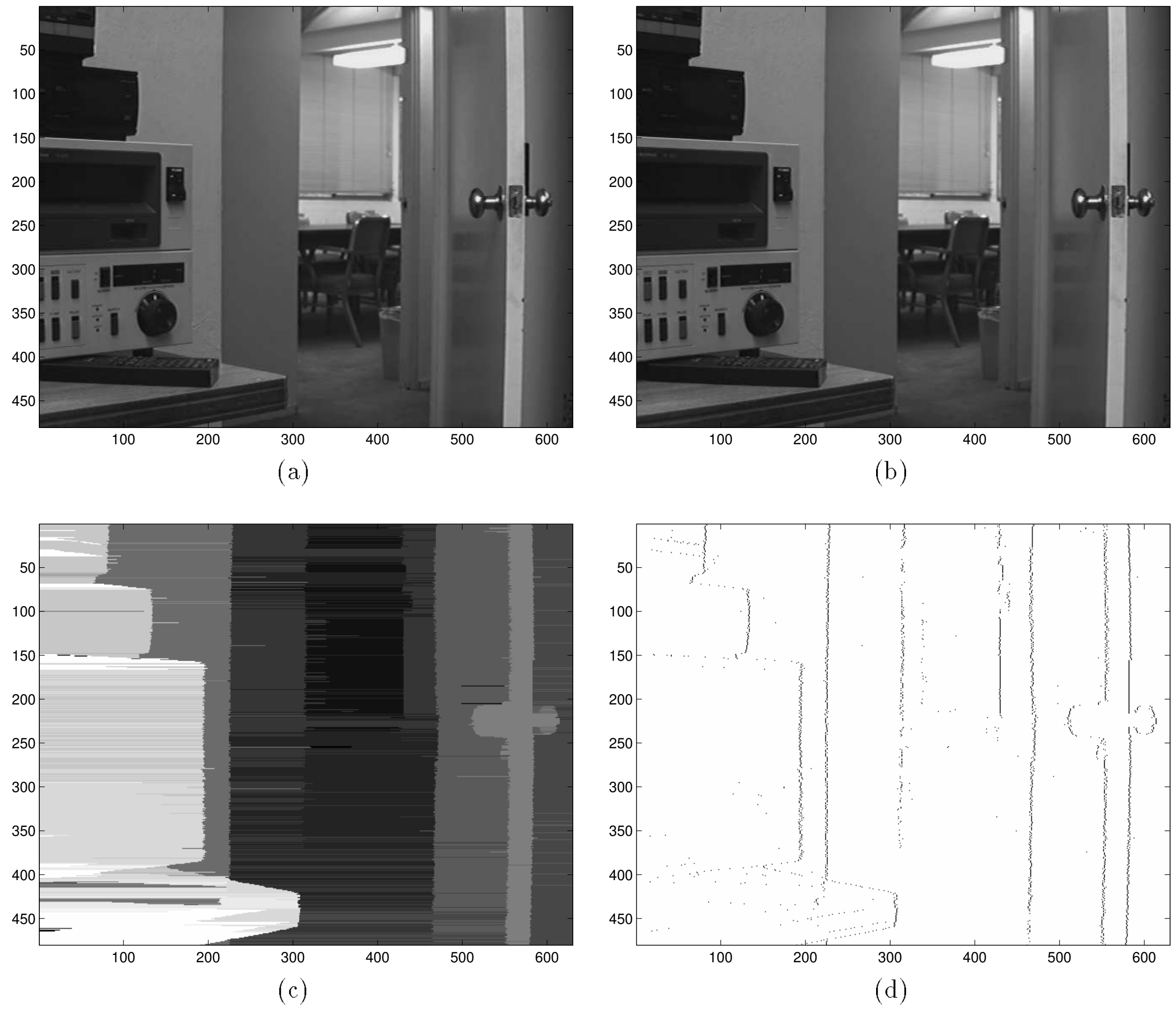


Figure 5.5: (a) Right image. (b) Left image. (c) Disparity map. (d) Depth discontinuities.

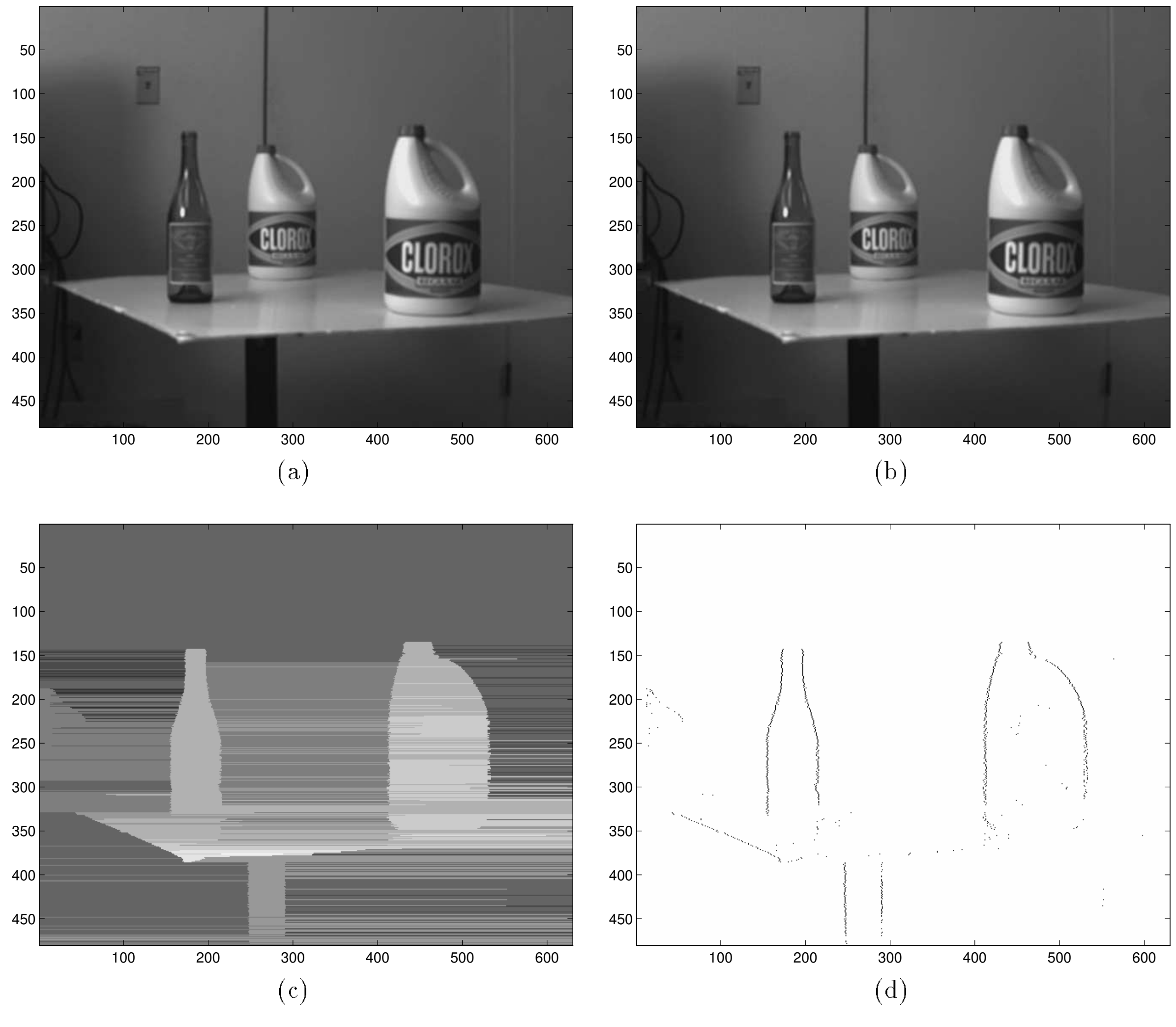


Figure 5.6: (a) Right image. (b) Left image. (c) Disparity map. (d) Depth discontinuities.

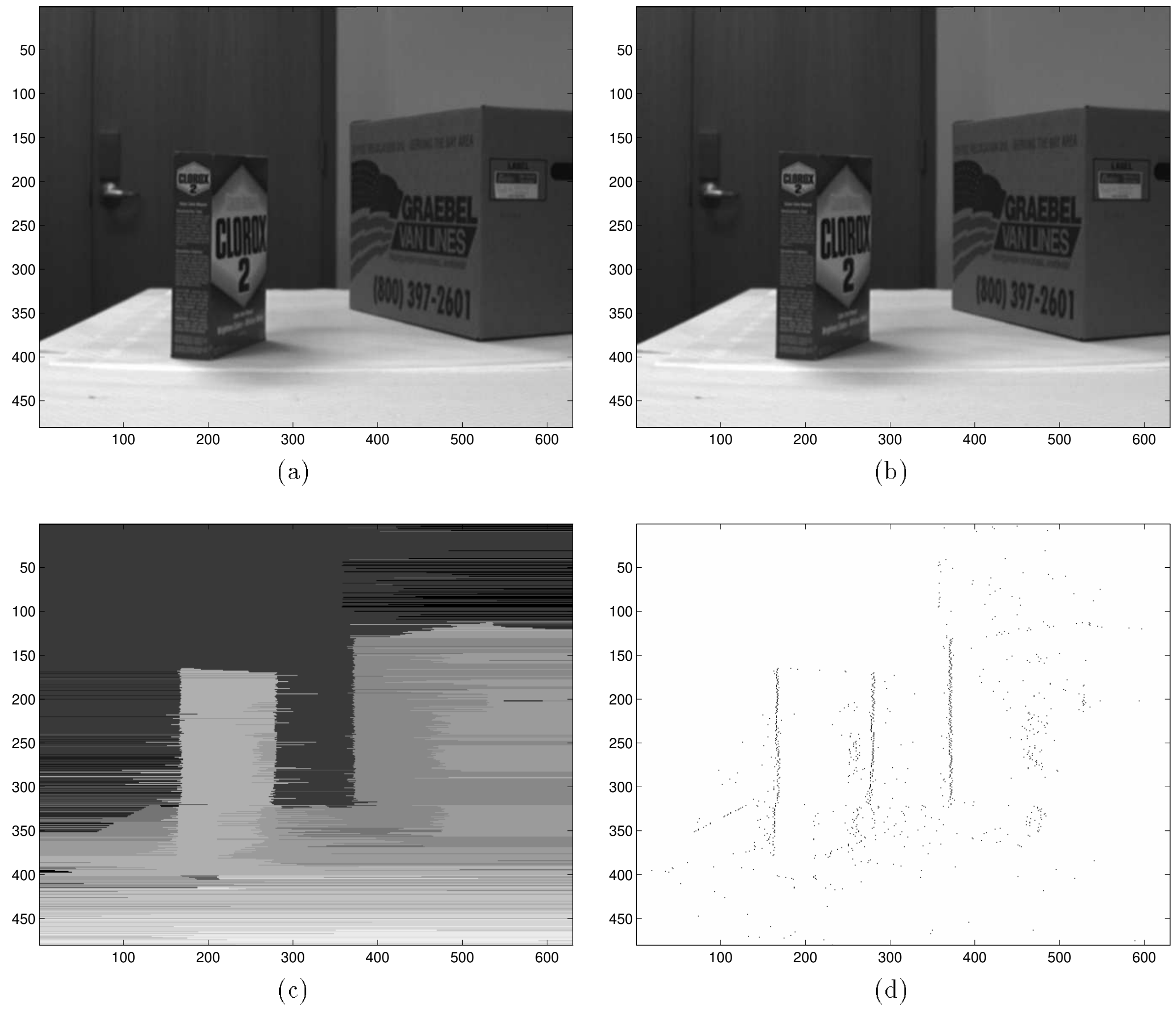


Figure 5.7: (a) Right image. (b) Left image. (c) Disparity map. (d) Depth discontinuities.

Part II

Propagating Information Between Scanlines

Chapter 6

Postprocessing the Disparity Map

In an image, the intensity values of pixels from different scanlines are not independent. Similarly, the true disparities of a pair of stereo images are not independent from one scanline to the next. Thus, while processing scanlines independently, as we did in Part I, is computationally attractive and straightforward to formulate, it does not take full advantage of all the information in the images. Thus, we seek a way to incorporate this information.

A common approach is to extend the one-dimensional cost function (such as Equation 3.1) to a two-dimensional cost function, which is then minimized. However, minimizing such a function in a computationally efficient manner is not a straightforward task. In the extension from 1D to 2D, it is not uncommon for the computing time to increase by 600% or more [2, 12]. As a result, some approaches avoid the extension altogether [7, 8]. Moreover, we cannot rely on techniques that look only at pairs [1] or triplets [2] of adjacent scanlines at a time, since our initial disparity maps contain errors over large regions. For example, incorrect disparities are found in Figure 5.4 in the concavity of the lamp, in the region to the right of the lamp, and in the region in the bottom-right-hand corner of the image.

In light of these observations, we have devised a method for postprocessing the disparity map by propagating reliable disparity values into regions of unreliable disparity values. This postprocessing is rather global in nature and is quite effective at propagating the background disparities into regions with little intensity variation. Moreover, it is fast, taking only a couple of seconds on a workstation.

Figure 6.1 shows a schematic of the information flow. The algorithm of Part I takes the two original images and two binary images of intensity gradients and computes a disparity map. The postprocessing uses this disparity map and the intensity gradients of the left image to compute a final disparity map. The original images are not available to the postprocessor.

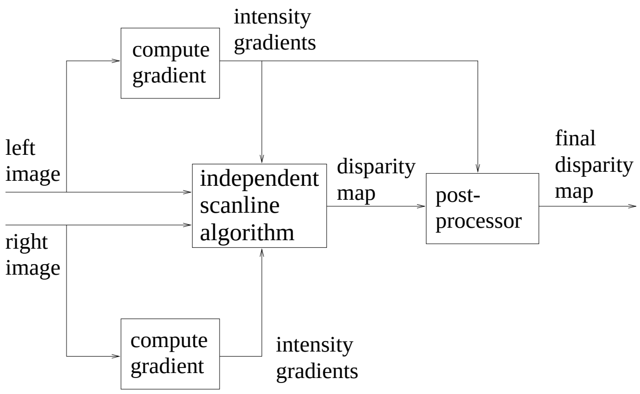


Figure 6.1: Information flow.

After postprocessing, depth discontinuity pixels are labelled as those pixels that lie on the far object and that border a change of at least two disparity levels. This threshold of two allows for some disparity variation within an object and helps to prevent depth discontinuities from being declared within slanted objects.

The speed of the postprocessing is due to the alternation between independent rows (scanlines) and independent columns. More specifically, after the algorithm of Part I processes the rows independently, the following steps are performed:[1]

S1. Obvious errors in the disparity map are removed.

S2. In the y direction:

   (a) Intensity gradients are computed.
   (b) "Isolated" intensity gradients are removed.
   (c) Stable regions are propagated.

S3. Step S2 is repeated in the x direction.

S4. The disparity map is mode-filtered in the y direction.

S5. The disparity map is mode-filtered in the x direction.

These steps are now explained in more detail:

S1. Isolated disparity values are removed by checking each pixel's two neighbors: the pixel above and the pixel below. If these neighbors have the same disparity as each other but a different disparity from the pixel itself, then the disparity of the pixel is changed to that of its neighbors. This step has little effect on the disparity map but corrects obvious errors.

[1]For the rest of this report, the term "x direction" refers to the direction along the rows, while the "y direction" is along the columns.

S2. In the y direction, the following steps are performed:

(a) A binary map of the intensity gradients is constructed by centering 3×1 vertical windows around every pixel in the image. If the intensity variation within a window is at least five gray levels, then all of the pixels within the window are labelled as intensity gradients.

(b) Since the intensity gradients are obtained by processing each column independently, we remove, along any particular row, every gradient for which three adjacent columns do not agree.

(c) This step is the heart of the postprocessing and is by far the most difficult to explain. The basic idea, however, is simply to propagate reliable regions into unreliable regions and into regions with higher disparity, until an intensity gradient is encountered. Reliability is defined for each pixel within a given column as the number of contiguous rows that agree on the pixel's disparity. For 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.)

We use three thresholds for reliability. A *highly reliable* region is a set of contiguous pixels whose reliability is at least t_h, a *moderately reliable* region is at least t_m, and a *slightly reliable* region is at least t_s. Since we enforce $t_h > t_m > t_s$, each of these definitions subsumes the previous one. A region whose reliability is less than t_s is called *unreliable*. In general, highly and moderately reliable regions are offensive in propagating their values into neighboring regions, while slightly reliable regions are defensive in maintaining their values.

A highly reliable region propagates its disparity along its column, changing the disparity of the neighboring pixels. Propagation stops when it reaches an intensity gradient or a slightly reliable region with a lower disparity. Regions with a higher disparity are overrun no matter what their reliability. The reason for this is that reliability is not a good indication that the disparities are correct when the background has little intensity variation, i.e., when Assumption A4 of Chapter 2 is violated. Thus, in the cavity of the lamp of Figure 5.4, all of the rows agree on the incorrect disparity because there is little intensity variation on the background.

A moderately reliable region acts exactly like a highly reliable region, with one exception: It cannot propagate into a slightly reliable region that is only one disparity level away. This rule enables some sloping surfaces to remain intact. For example, the farthest region of the table top of Figure 8.2 is prevented from propagating into the nearer regions.

Objects that have nearly horizontal borders cause a problem. Due to the independence of the scanlines resulting from the algorithm of Part I, a particular scanline that straddles the background and the object may assign the background disparity to the object. Then the background disparity will be propagated into the object, and possibly over a large portion of it. This is the case of the top edge of the table, near scanline 300 of Figure 5.4. Therefore, we make one exception to the propagation described in the previous two paragraphs. Before a region is allowed to propagate up, it checks whether the region just above its top pixel is slightly reliable, and whether there is an intensity gradient within a certain number (t_g) of pixels below its top pixel. If these two tests are true, then instead of the region propagating up, the slightly reliable region propagates down to the intensity gradient. A similar procedure is performed before a region is allowed to propagate down.

S3. Propagation occurs in the x direction, in a manner symmetric to that described in Step S2.

S4 and S5. These two steps perform mode filtering with a 5×1 vertical and a 1×5 horizontal filter, respectively. That is, all of the pixels within the filter are allowed to vote, and the middle pixel is labelled with the majority decision. By filtering in the two directions sequentially, we preserve corners.

We have now described in detail the postprocessing computation. Although some of the steps are rather ad hoc, it must be kept in mind that the computation is fast and achieves acceptable results for a set of five images. Moreover, these images were taken and selected before the postprocessing procedure was developed, and they contain a wide range of situations: textured and untextured objects, textured and untextured backgrounds, specular and matte surfaces, planar and curved surfaces, and fronto-parallel and slanted (in both the x and y directions) surfaces. The next chapter gives some indication of the generalizability of this computation to other images.

Chapter 7

Assumptions

To get a feel for what type of image is amenable to the postprocessing computation described in the previous chapter, we will attempt to list the assumptions that it makes, in a manner similar to that of Chapter 2. This list, however, is not nearly as rigorous, since the postprocessing computation consists of several alternating steps. The numbering system continues from Chapter 2:

A8. Intensity gradients accompany changes in disparity along columns and rows.

A9. In any row or column, if fewer than t_s contiguous pixels agree on a disparity, then the disparity of those pixels is completely unreliable.

A10. If a change in disparity is not accompanied by an intensity gradient, and if the region with the lower disparity contains at least t_m pixels, then the improper placement of the disparity change was due to a violation of Assumption A4 of Chapter 2, that is, the background along a particular row or column did not contain any intensity variation.

A11. Each object (including the background) falls into one of two categories:

   (a) *either* the object is nearly fronto-parallel, meaning that the object straddles at most two disparities,

   (b) *or* the object's slant is at least $\frac{1}{t_m}$ disparity levels per pixel but still small enough and textured enough that the interior of the object can be labelled with successive disparities.

A12. Every depth discontinuity is accompanied by a change of at least two disparity levels.

A13. Each background pixel is reachable from a moderately reliable region of the background by moving first in the y direction, then in the x direction.

Surprisingly, these assumptions do not appear to be very restrictive. Assumption A8 is similar to A3 except that intensity gradients now must lie in the y direction as well as the x direction. Also, the notion that the intensity gradient has the same disparity as the near object is no longer used. This assumption seems to be valid much of the time, but our current postprocessor does not easily recover from its violations – a more powerful approach is needed. Assumption A9 seems fairly reasonable, as long as the scene does not contain very small or thin objects. The assumption implicit in A10 is that A8 is never violated for more than $t_m - 1$ contiguous pixels. In other words, the disparity of a far object is not allowed to leak into a near object for more than $t_m - 1$ contiguous pixels. This assumption, which is invalid only in the images of Figure 5.5, leads to the powerful heuristic used in Step S2(c) that smaller disparities win over larger disparities. Assumption A11, coupled with A12, allows for a wide range of objects in the scene, at the expense of losing some true depth discontinuities; the threshold for declaring depth discontinuities is the smallest number available that still allows some variation in disparity within an object. Finally, Assumption A13 seems fairly safe and always holds in our images. It would be violated, for example, in the presence of an object shaped like  in which the interior of the object is only reachable by more complicated propagation. As another example, it would be violated in Figure 5.4 if the top of the lamp were near the top of the image, in which case the correct disparities from the vertical stripe on the wall could not propagate to the right side of the lamp.

Chapter 8

Experimental Results

Figures 8.1 through 8.5 show the results of postprocessing the disparity maps of Chapter 5. The resulting disparity maps and depth discontinuities are significantly cleaner and more accurate. Since depth discontinuities are detected in the vertical as well as the horizontal direction, they tend to outline the objects.

These results show the tradeoff that the algorithm and postprocessor make. The values of the disparity maps are only approximate, which is obvious from the fact that objects tend to be labelled with either a single disparity or with two disparities that alternate in somewhat random locations. See, for example, the bottles in Figure 8.4. However, because no windows were used for matching, and because no prefiltering (such as taking the Laplacian of Gaussian) was performed on the images, the boundaries of objects are crisp. Therefore, although there are gross errors, such as the triangular wedge that is cut out of the left Clorox bottle in Figure 8.1, the depth discontinuities are, for the most part, located precisely on the boundaries of the objects. Especially noteworthy are the results in Figures 8.2 and 8.5, in which the depth discontinuities are approaching perfect. Even the tiny light bulb inside the lamp is almost discernable.

For the results presented in this chapter, the thresholds were set as follows: $t_h = 25$, $t_m = 15$, $t_s = 12$, and $t_g = 10$. These values were used for all of the images.

The postprocessor takes about two seconds on a Silicon Graphics Indy workstation to postprocess these images, with Δ set to 14. The computation is at most linearly related to Δ and takes about four seconds when Δ is set to 50.

We will now examine each of the assumptions in turn:

A8. *Intensity gradients accompany depth discontinuities.* This assumption is valid much of the time and is crucial to the success of the approach. It allows us to remove a lot of the false disparities from the algorithm of Part I, and it allows us to deal with scanlines along which the background has little intensity variation.

Unfortunately, when this assumption is violated, performance degrades drastically. Thus the location of the object boundaries, or depth discontinuities, can be off by 75 pixels or more. This is evident in Figure 8.1 along the top of the left Clorox bottle, the triangular wedge of the same bottle, and the neck of the wine bottle. The problems on the Clorox top and the wine neck actually disappear after Step S2 but reappear after the values are propagated in the x direction in Step S3.

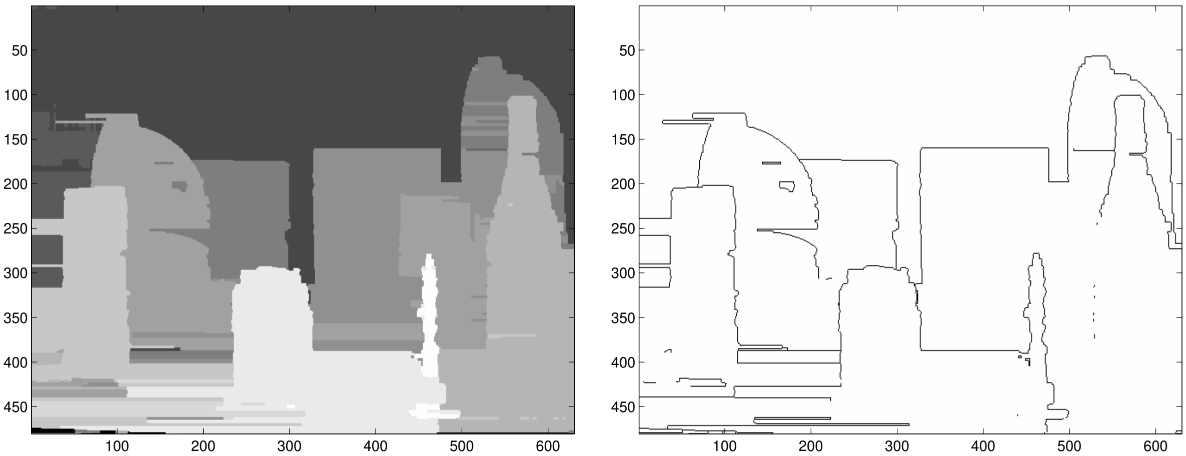
A9. *Small regions are unreliable.* This assumption holds in our images. Even the little pencil in Figure 8.1 is at least 12 pixels thick.

A10. *Smaller disparities win over larger disparities.* This assumption is extremely powerful and is responsible for the good results of Figure 8.2. However, when the assumption is violated, the consequences could potentially be far-reaching. In Figure 8.3, for example, near row 70 and column 250, the far object leaks into the near object in 20 consecutive rows. Were it not the case that the disparities of the objects differ by only one, the near object would be completely wiped out.

A11. *Objects are either fronto-parallel or significantly slanted.* Surprisingly, some objects that we consider slanted are not considered slanted by this assumption. For example, both of the boxes of Figure 8.5 straddle at most two disparity levels and are thus considered "fronto-parallel" by the algorithm. The second part of this assumption is crucial in preserving the slanted nature of the table in Figures 8.2, 8.4, and 8.5.

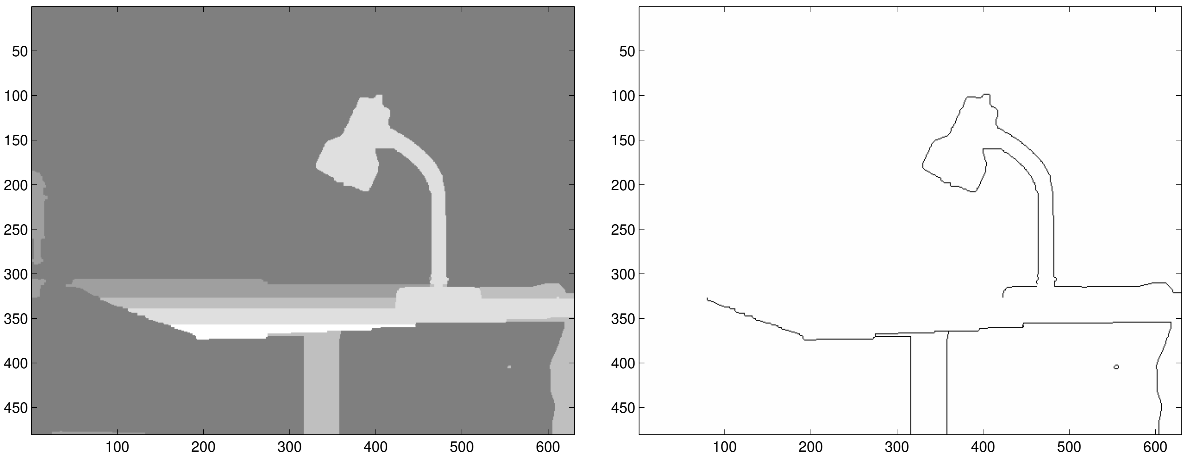
A12. *Depth discontinuities are accompanied by changes of at least two disparity levels.* This assumption is very powerful in that it allows us to handle slanted surfaces without explicitly detecting them or dealing with them. Some true depth discontinuities are lost, however, as can be seen from the Splash can of Figure 8.1, the back of the table in Figure 8.2, and the middle Clorox bottle of Figure 8.4.

A13. *Background pixels are reachable.* This assumption is valid in our images.



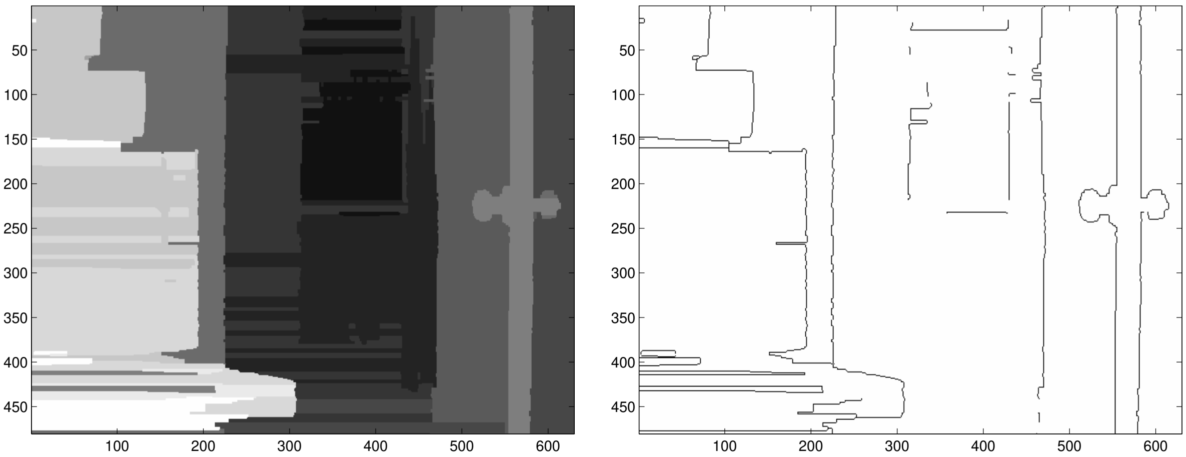
(a) (b)

Figure 8.1: (a) Disparity map. (b) Depth discontinuities.



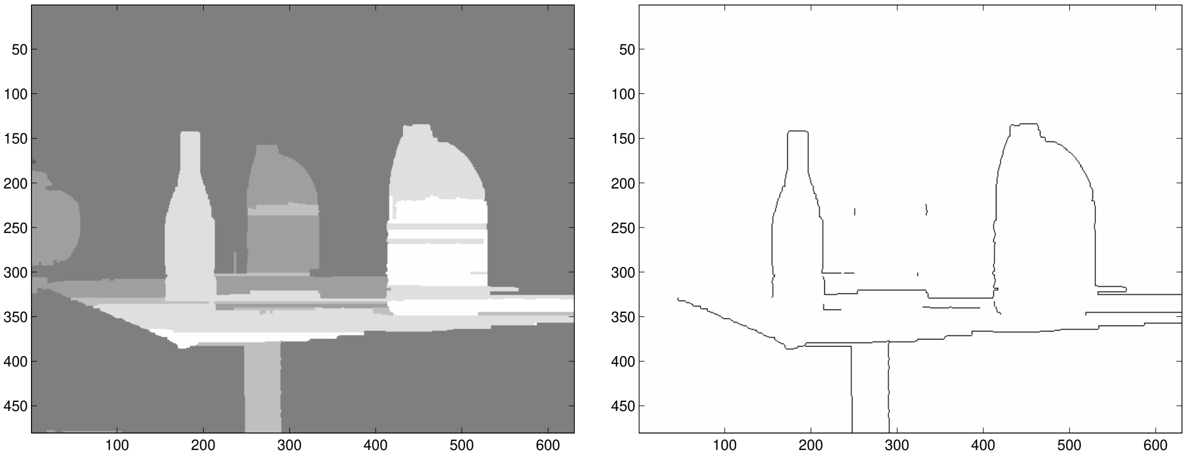
(a) (b)

Figure 8.2: (a) Disparity map. (b) Depth discontinuities.



(a) (b)

Figure 8.3: (a) Disparity map. (b) Depth discontinuities.



(a) (b)

Figure 8.4: (a) Disparity map. (b) Depth discontinuities.

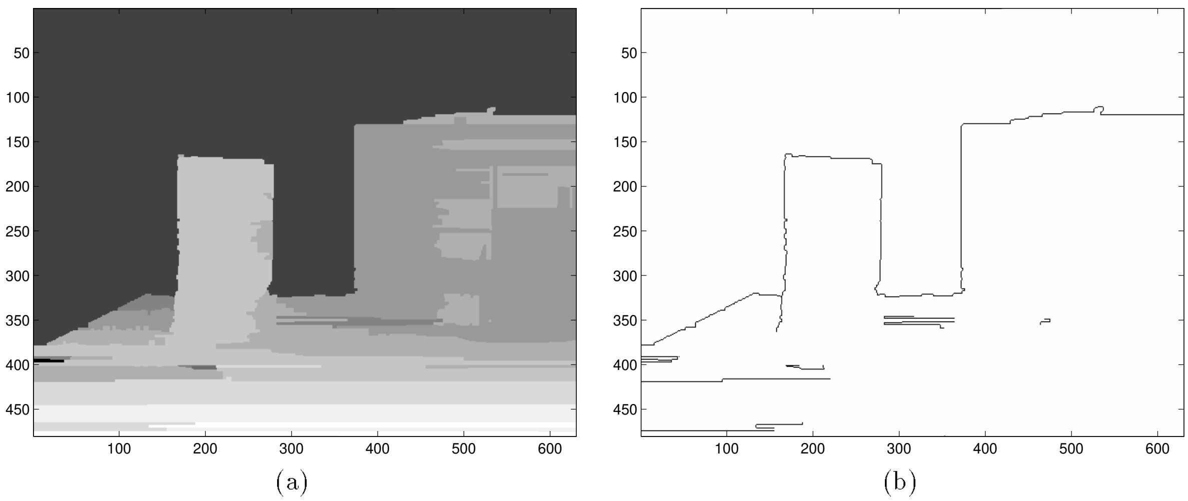


Figure 8.5: (a) Disparity map. (b) Depth discontinuities.

Chapter 9

Conclusion

A simple and fast stereo algorithm for the purpose of detecting depth discontinuities has been presented. The algorithm explicitly matches the pixels in one scanline with the pixels in the corresponding scanline while leaving occluded pixels unmatched. Depth discontinuities are precisely located, since the images are not prefiltered and since no windows are used for matching.

The contributions of the algorithm are threefold. First, it uses a measure of pixel dissimilarity that is provably insensitive to sampling. Thus we are spared the computationally expensive solution of matching at subpixel resolution or over windows. Second, it handles large untextured regions by making the assumption that intensity gradients always accompany depth discontinuities. This assumption seems to be valid much of the time, since our threshold for declaring intensity gradients is low. Finally, the algorithm is significantly faster than stereo algorithms that use standard dynamic programming, because it prunes bad nodes during the search.

After the scanlines are processed independently, a simple and fast postprocessing step propagates disparity information between scanlines to produce a more refined disparity map. Depth discontinuities are then detected in the horizontal and vertical directions as significant changes in the disparity map. Results on five pairs of stereo images show that, with little computational effort, a rough disparity map can be constructed, and many of the depth discontinuities can be accurately located. These images contain surfaces that are both fronto-parallel and slanted, textured and untextured, planar and curved, and matte and specular.

By explicitly laying down the assumptions behind both the algorithm and the postprocessor, we have tried to get a feel for how generally applicable they are. We suspect that they will work on many high-quality indoor images obtained with a slightly defocused lens and with cameras having a small baseline. In addition, for good performance the objects should be reasonably outlined by intensity gradients.

The postprocessor, while effective on these images, has limitations. For one, the approach seems to be rather brittle in the sense that the violation of an assumption has the potential of creating large errors in the output. The output of the postprocessor has obvious errors, such as boundaries terminating freely in space, jagged boundaries, and tiny, disconnected boundaries. Also, the postprocessor should take advantage of the information available in the original images; it could use heuristics such as the smoothness of boundaries; and it must be more robust to the absence of intensity gradients at boundaries. Finally, the postprocessor described in this report is somewhat ad hoc, contains several thresholds (but what vision algorithm does not?), and should be replaced by a more principled approach.

Beyond the immediate limitations there remain open questions, as always. How well would the method perform on a real stereo rig with two cameras taking images of a dynamic scene? Would the results be reasonable, or would the computation be too brittle? And could it be implemented in real time without specialized hardware? Looking beyond, is there a way to provide consistency between time instants, so that a sequence of stereo pairs can be processed faster and more accurately than if each pair is processed independently? How would we detect depth discontinuities from an unconstrained image sequence in which the epipolar geometry is not known? These are some of the questions that we feel future research should address.

Bibliography

[1] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. In *Proc. 7th IJCAI* , pp. 631-636, August 1981.

[2] P. N. Belhumeur and D. Mumford. A Bayesian treatment of the stereo correspondence problem using half-occluded regions. In *CVPR*, pp. 506-512, 1992.

[3] P. N. Belhumeur. A binocular stereo algorithm for reconstructing sloping, creased, and broken surfaces in the presence of half-occlusion. In *Proc. 4th ICCV*, pp. 431-438, May 1993.

[4] D. N. Bhat and S. K. Nayar. Stereo in the presence of specular reflection. In *Proc. 5th ICCV*, pp. 1086 - 1092, June 1995.

[5] O. Faugeras. *Three-Dimensional Computer Vision.* The MIT Press: Cambridge, Massachussetts, 1993, pp. 178-181.

[6] P. Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *Proc. 12th IJCAI*, pp. 1292-1298, 1991.

[7] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and binocular stereo. *IJCV*, 14(3): 211-226, April 1995.

[8] S. S. Intille and A. F. Bobick. Disparity-space images and large occlusion stereo. In *ECCV*, pp. 179-186, 1994.

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

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

[11] A. Luo and H. Burkhardt. An intensity-based cooperative bidirectional stereo matching with simultaneous detection of discontinuities and occlusions. *IJCV*, 15(3):171-188, July 1995.

[12] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. PAMI*, PAMI-7(2): 139-154, March 1985.

[13] M. Oren and S. K. Nayar. Generalization of the Lambertian model and implications for machine vision. *IJCV*, 14(3): 227-252, April 1995.

Appendix A

Theorems and Proofs of Pixel Dissimilarity Measure

In Section 3.1 we proposed the following measure for computing the dissimilarity d between a pixel x in the left scanline and a pixel y in the right scanline:

$$d(x, y) = \min\{\bar{d}(x, y, I_L, I_R),\ \bar{d}(y, x, I_R, I_L)\}, \tag{A.1}$$

where

$$\bar{d}(x, y, I_L, I_R) = \min_{y-\frac{1}{2} \leq y' \leq y+\frac{1}{2}} |I_L(x) - \hat{I}_R(y')|, \tag{A.2}$$

$I_L(x)$ is the intensity of pixel x in the left scanline, and $\hat{I}_L(x)$ is the linearly interpolated intensity at position x in the left scanline. Similar notation holds for the right scanline. Since x and y are measured in pixels, the expression $y - \frac{1}{2} \leq y' \leq y + \frac{1}{2}$ requires y' to be within half a pixel of y on either side.

Let i_L and i_R denote the continuous intensity functions incident upon the left and right scanlines, respectively, prior to sampling. We assume that our cameras are ideal samplers, and that there is neither photometric nor geometric distortion or shift between the two intensity functions, so that $i_L = i_R = i$. Keep in mind that this restriction is only required in a very small neighborhood surrounding each pixel. In the following two theorems, we show that the dissimilarity measure defined in Equation A.1 is insensitive to sampling if i satisfies a certain condition.

Theorem 1 *Let i be either convex or concave on an interval A, and let x and y be sufficiently inside A so that $[x - \frac{1}{2}, x + \frac{1}{2}] \subseteq A$ and $[y - \frac{1}{2}, y + \frac{1}{2}] \subseteq A$. If $|x - y| \leq \frac{1}{2}$, then $d(x, y) = 0$.*

This theorem states that if the sampling points x and y are closer to each other than they are to any other sampling points (thereby implying that correspondence should be established between them), then the dissimilarity of the two pixels is zero, as measured by Equation A.1. This theorem holds as long as the continuous intensity function is either concave or convex in the vicinity of the two pixels. By "convex" we mean "weakly convex", and similarly for "concave".

When the continuous intensity function is linear, it is by definition both convex and concave, and therefore satisfies Theorem 1. In addition, a stronger statement can be made in this case.

Theorem 2 *Let i be linear and have nonzero slope on an interval A, and let x and y be sufficiently inside A so that $[x-\frac{1}{2}, x+\frac{1}{2}] \subseteq A$ and $[y-\frac{1}{2}, y+\frac{1}{2}] \subseteq A$. Then $d(x,y) = 0$ if and only if $|x-y| \leq \frac{1}{2}$.*

These two theorems state that, using the dissimilarity measure of Equation A.1, sampling will not prevent two pixels from being matched whenever the continuous intensity function is convex or concave; in addition, sampling will not cause two pixels to be be incorrectly matched when the intensity function is linear with nonzero slope. Although, strictly speaking, these criteria do not hold near inflection points, in practice the regions surrounding inflection points look and behave almost like linear regions as long as the intensity function varies smoothly. Slightly defocusing the lens helps to ensure this condition.

In order to prove these two theorems, we will use the following lemma.

Lemma 1 *Consider a function g defined on an interval A of the real line. If g is convex and if $x, y, z \in A$ satisfy $x \leq y \leq z$, then either $g(y) \leq g(x)$ or $g(y) \leq g(z)$, or both.*

Proof Suppose, for the purpose of contradiction, that $g(y) > g(x)$ and $g(y) > g(z)$. Consider the case in which $g(x) \leq g(z)$; by multiplying both sides of this inequality by the number $\lambda = \frac{z-y}{z-x}$, adding $g(z)$ to both sides, and rearranging terms, we obtain the following:

$$g(z) \geq \lambda g(x) + (1-\lambda) g(z),$$

which, since $g(y) > g(z)$ and $y = \lambda x + (1-\lambda)z$, contradicts the statement that g is convex:

$$g(\lambda x + (1-\lambda)z) \leq \lambda g(x) + (1-\lambda) g(z) \qquad \forall \lambda \in [0,1].$$

(Note that $\lambda \in [0,1]$ because $x \leq y \leq z$.) The case in which $g(z) \leq g(x)$ leads us to a similar contradiction using the fact that $g(y) > g(x)$. Therefore, either $g(y) \leq g(x)$ or $g(y) \leq g(z)$. $\square$

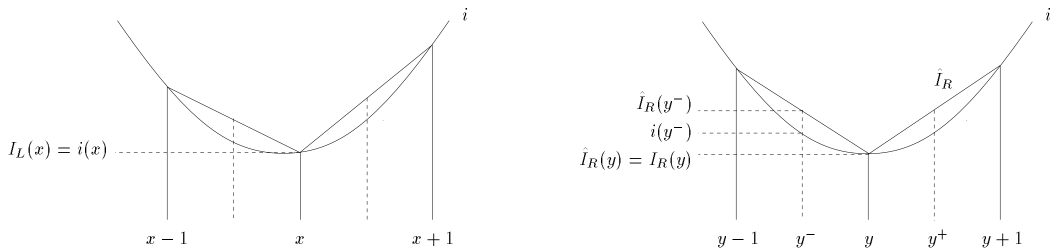


Figure A.1: The proof of Theorem 1.

Proof of Theorem 1 We will prove only the case in which i is convex; the other case can be proved similarly. In order to show that $d(x, y) = 0$, we need to show that either $\bar{d}(x, y) = 0$ or $\bar{d}(y, x) = 0$, since $\bar{d}$ is nonnegative.[1]

Let $y^- \equiv y - \frac{1}{2}$ and $y^+ \equiv y + \frac{1}{2}$, as shown in Figure A.1. Then $|x - y| \le \frac{1}{2}$ implies that either $i(x) \le i(y^-)$ or $i(x) \le i(y^+)$, by Lemma 1. Suppose for the moment that $i(x) \le i(y^-)$. By assuming that $I_R(y) \le I_L(x)$, we will show that $\bar{d}(x, y) = 0$. (If we instead assume that $I_L(x) \le I_R(y)$, we can use a similar procedure to show that $\bar{d}(y, x) = 0$.) With these assumptions, we have the following:

$$\hat{I}_R(y) = I_R(y) \le I_L(x) = i(x) \le i(y^-),$$

where we have used the fact that $i(w) = I_s(w) = \hat{I}_s(w)$ for any w which is a sampling point of camera s. Since i is convex, $i(y^-) \le \hat{I}_R(y^-)$. Therefore,

$$\hat{I}_R(y) \le I_L(x) \le \hat{I}_R(y^-).$$

Since $\hat{I}_R$ is continuous on the interval $[y^-, y]$, then by the Intermediate Value Theorem[2] there exists a $y' \in [y^-, y]$ such that $\hat{I}_R(y') = I_L(x)$. Similarly, if we had assumed that $i(x) \le i(y^+)$ we would have concluded that there exists a $y' \in [y, y^+]$ such that $\hat{I}_R(y') = I_L(x)$. In either case, this y' is between y^- and y^+, and therefore $\bar{d}(x, y) = 0$, which makes $d(x, y) = 0$. □

[1] We adopt the shorthand notation of $\bar{d}(x, y)$ for $\bar{d}(x, y, I_L, I_R)$, and $\bar{d}(y, x)$ for $\bar{d}(y, x, I_R, I_L)$.

[2] The Intermediate Value Theorem states: If a function g is continuous on the interval $[a, b]$, and if c is a number between $g(a)$ and $g(b)$, inclusive, then there exists at least one number x between a and b, inclusive, such that $g(x) = c$.

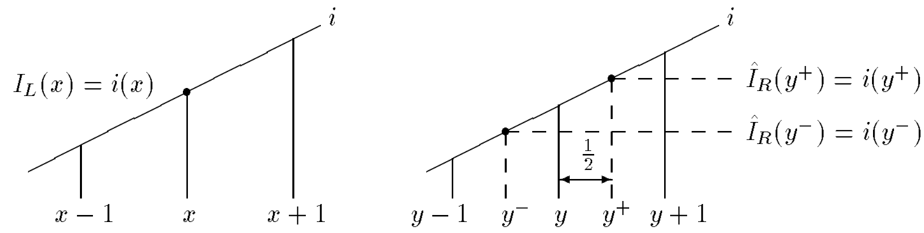


Figure A.2: The proof of Theorem 2.

Proof of Theorem 2. We will prove only the case in which i is increasing; the case in which i is decreasing can be proved similarly.

"if": If $y^- \leq x \leq y^+$, then $i(y^-) \leq i(x) \leq i(y^+)$, since i is increasing. Because i is linear, its linear interpolation is itself: $\hat{I}_R = i$. Therefore, $\hat{I}_R(y^-) \leq i(x) \leq \hat{I}_R(y^+)$. As a result of the Intermediate Value Theorem, there exists a $y' \in [y^-, y^+]$ such that $\hat{I}_R(y') = i(x)$, and therefore $\bar{d}(x,y) = 0$, making $d(x,y) = 0$.

"only if": if $x > y^+$, then $I_L(x) > \hat{I}_R(y')$ for all $y' < y^+$, since i is increasing; therefore there is no number $y' \in [y^-, y^+]$ such that $\hat{I}_R(y') = I_L(x)$, which makes $\bar{d}(x,y) > 0$. A similar argument leads us to conclude that $\bar{d}(y,x) > 0$, and therefore $d(x,y) > 0$. By symmetry, if $x < y^-$ then $d(x,y) > 0$. Therefore, if it is not the case that $y^- \leq x \leq y^+$, then $d(x,y) \neq 0$. □