REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time f
and maintaining the data needed, and completing and reviewing the collection of information. Send comments regar
information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for inform
1204, Arlington, VA 22202-4302, and to the Office of management and Budget, Paperwork Reduction Project (0704-0188)

AFRL-SR-BL-TR-98-0440

1. AGENCY USE ONLY (Leave Blank)

2. REPORT DATE
1994

3. REPORT TYI
Final

4. TITLE AND SUBTITLE
View Reconstruction from Uncalibrated Cameras for Three-Dimensional Scenes

5. FUNDING NUMBERS

6. AUTHORS
Nelson L. Chang

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of California at Berkeley

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NI
110 Duncan Avenue, Room B-115
Bolling Air Force Base, DC 20332-8080

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
See attached.

19980518 029

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Abstract

With the development of faster memory and graphics hardware, there has been increased interest in the development of virtual environments. Researchers have been investigating the representation of mathematically defined objects and more recently the subsequent reconstruction of a view of the scene. The result has led to virtual worlds consisting of mainly geometric objects.

Our interest is to extend this idea to real 3-D objects and scenes without resorting to modeling. While modeling uses little storage, the resulting images depend on the number of model parameters and require heavy computation. An alternative approach would be to use a very large number of views of the scene. This solution provides high quality images, but it also requires a large amount of memory. Hence, there is a tradeoff between storage requirements and accuracy of the reconstructed views.

In this thesis, we propose an approach which offers high quality reconstructions with relatively low storage requirements. Our approach consists of a camcorder scanning the desired stationary object along several pre-specified trajectories. Neither the camera's exact motion nor its internal parameters are assumed to be known *a priori*. For certain locations in each trajectory, depth information is recovered using an adaptive region matching algorithm. The depth and corresponding intensity information are then used to generate accurate reconstructions. Results for intermediate views are excellent and preliminary results are promising for views not originally scanned by the camcorder.

The eventual goal is to devise an environment consisting of more realistic and complicated objects. One possible system includes special hardware such as a stereoscopic display and a head tracking device. The viewer is able to gain the sense of 3-D by controlling the view of a scene by the movement of her head. We expect such a system with real-time reconstruction to become realizable in the near future.

View Reconstruction from Uncalibrated Cameras for Three-Dimensional Scenes

by Nelson L. Chang

(1994)

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

Committee:

Professor Avideh Zakhor
Research Adviser

(Date)

* * * * * *

Professor Jitendra Malik
Second Reader

(Date)

Contents

Acknowledgements ii

List of Figures iii

List of Tables v

Abstract vi

1 Introduction 1

2 Compact Representation of Scenes 4

2.1 Depth Estimation 6

2.1.1 Block Matching Algorithm (BMA) 9

2.1.2 Adaptive Block Matching Algorithm (ABMA) 12

2.1.3 Fast Adaptive Block Matching Algorithm (FABMA) 19

2.2 Normalization of Initial Estimates 22

2.3 Combination of Multiple Depth Maps 24

2.4 Cubic B-Spline Approximation 29

3 Reconstruction of Intermediate Views 33

3.1 Selection of Appropriate Reference Frame(s) 34

3.2 Generation of View Estimates 34

3.3 Combination of Reconstructed Data 37

4 Results 41

4.1 Reconstructing Horizontal View 42

4.2 Reconstructing Vertical View 55

5 Conclusion 61

References 66

Acknowledgements

First and foremost, I would like to thank my adviser, Dr. Avideh Zakhor, for giving me the opportunity to work with her on this research project. I appreciate all the support, advice, and encouragement she has given me throughout the course of this project.

Special thanks go out to Dr. Jitendra Malik for serving as second reader for this report and to Dr. Camillo J. Taylor for some enlightening discussions during the semester.

I am indebted to Professors Sanjeev Kulkarni and Bede Liu at Princeton University and Dr. Amy Reibman at AT&T Bell Labs for all the help and advice they gave me during my undergraduate years. Through working with them, I became inspired to pursue image processing/computer vision in graduate school.

My fellow officemates in 307 Cory have kept me sane during the many late nights I had to stay up. Thanks guys for lots of laughs.

Finally, I would like to thank my parents, my brother Andrew, and my sister-in-law Amy for their generous love and support throughout the years.

This work was supported in part by an Air Force Laboratory Graduate Fellowship, National Science Foundation grant number MIP–90–57466 (PYI), Office of Naval Research Young Investigator award N00014–92–J–1732, and the Joint Service Electronics Program contract number F49620–93–C–0014.

List of Figures

2.1 An example of rectangular scanning geometry 4
2.2 A diagram of the complete representation process 6
2.3 Estimating local depth between reference frame and neighboring frames . . . 7
2.4 Geometry of depth estimation problem from two fronto-parallel images under perspective projection 8
2.5 Example of BMA for horizontal motion case 10
2.6 Example of regions where BMA fails due to various artifacts 11
2.7 Two images from "Mug2" sequence 14
2.8 Example of disparity estimate from matching Frame #37 with Frame #34 . 15
2.9 Example of intensity image and its corresponding confidence regions after LVT with a 3×3 block has been performed 17
2.10 Example of a single-impulse image used with ABMA 18
2.11 Example of disparity estimate from matching Frame #37 with Frame #34 using ABMA and FABMA 21
2.12 Exploiting geometry of camera set up to normalize depth maps 22
2.13 Examples of using mean and median to throw out outliers 25
2.14 Example of a reference frame intensity-depth pair 28
2.15 Example of an image with confidence regions 31
2.16 Example of processing combined depth map with splines 31

3.1 A diagram of the complete reconstruction process 33
3.2 Example of using two reference frames to handle occlusions 34
3.3 1-D example of reference frame regarded as discrete set of points 36
3.4 1-D example of reference frame regarded as part of a deformable mesh 37
3.5 Example of interpolating data from two reference frames for pixel (i,j) 38

4.1 Experimental set up used to generate results 41
4.2 Reference Frames #35 and #65 of "Mugl" (intensity) 43
4.3 Closest image to desired view from "Mugl" sequence 44
4.4 Reference Frames #35 and #65 (depth) using BMA 45
4.5 Results from BMA: reconstructed view and error 46
4.6 Reference Frames #35 and #65 (depth) using ABMA with confidence measures 48
4.7 Reference Frames #35 and #65 (depth) using ABMA filled by splines 49
4.8 Results from ABMA: reconstructed view and error 50

4.9 Reference Frames #35 and #65 (depth) using FABMA with confidence measures 51
4.10 Reference Frames #35 and #65 (depth) using FABMA filled in by splines . . 52
4.11 Results from FABMA: reconstructed view and error 53
4.12 Graph of mean squared error for the three matching algorithms 54
4.13 Reference Frame #37 of "Mug2" (intensity) 56
4.14 Results from BMA: depth and reconstructed view 57
4.15 Results from ABMA: depth and reconstructed view 58
4.16 Results from FABMA: depth and reconstructed view 59
4.17 Reconstructed view along vertical trajectory using only vertical match information 60

5.1 Example of an interactive virtual environment 61

List of Tables

2.1 Weights used in combining together depth information of different confidence levels from both horizontal and vertical matches 25

4.1 MSE of horizontal view for three matching algorithms 55

Abstract

With the development of faster memory and graphics hardware, there has been increased interest in the development of virtual environments. Researchers have been investigating the representation of mathematically defined objects and more recently the subsequent reconstruction of a view of the scene. The result has led to virtual worlds consisting of mainly geometric objects.

Our interest is to extend this idea to real 3-D objects and scenes without resorting to modeling. While modeling uses little storage, the resulting images depend on the number of model parameters and require heavy computation. An alternative approach would be to use a very large number of views of the scene. This solution provides high quality images, but it also requires a large amount of memory. Hence, there is a tradeoff between storage requirements and accuracy of the reconstructed views.

In this thesis, we propose an approach which offers high quality reconstructions with relatively low storage requirements. Our approach consists of a camcorder scanning the desired stationary object along several pre-specified trajectories. Neither the camera's exact motion nor its internal parameters are assumed to be known *a priori*. For certain locations in each trajectory, depth information is recovered using an adaptive region matching algorithm. The depth and corresponding intensity information are then used to generate accurate reconstructions. Results for intermediate views are excellent and preliminary results are promising for views not originally scanned by the camcorder.

The eventual goal is to devise an environment consisting of more realistic and complicated objects. One possible system includes special hardware such as a stereoscopic display and a head tracking device. The viewer is able to gain the sense of 3-D by controlling the view of a scene by the movement of her head. We expect such a system with real-time reconstruction to become realizable in the near future.

Chapter 1

Introduction

In light of recent advances in technology, there has been increased interest in the development of virtual environments. Several applications immediately come to mind especially in the area of modeling and prototyping. With a virtual system, one can easily design and run simulations without having to produce a sometimes costly physical prototype. Other situations may be possible like the real-estate agent who displays houses by having interested parties "walk-through" a simulation or the surgeon who studies a 3-D simulation before attempting the real surgery. These virtual environments are also useful in transmitting hard-to-visualize information to a remote location. For instance, a designer may want to show some prospective clients on the other coast her design and the consumer may wish to purchase items at a virtual shopping mall without leaving his home.

In the past, most of the attention fell on synthetic 3-D scenes attempting to model real 3-D ones. The representation for these scenes consists of a set of geometric models for every 3-D object in the scene [33, 15]. The realism of such scenes improves with the complexity of the models and with the use of intricate models for shading, illumination and reflectance.

More recently, the focus has turned more toward developing different approaches to generate realistic scenes for virtual environments. Especially important to the development of this growing field is the problem of generating novel views of a 3-D scene, what we shall refer to as Arbitrary View Generation (AVG). The goal of AVG is to devise a compact representation for realistic 3-D scenes in order to achieve high quality reconstructions of an arbitrary view of the scene. If AVG is capable of performing in real time, then this leads to simulations of realistic 3-D scenes and thus convincing virtual environments.

There are three major approaches to AVG of real world scenes. The first approach involves forming a 3-D model of the scene, applying the appropriate transformation to the model, and then reprojecting the model to obtain the desired 2-D view. Many researchers have adopted approaches which "backproject" given 2-D information to estimate the 3-D structure of the scene by volumetric intersection. Chien and Aggarwal [9] and Adrizzone *et. al.* [1] proposed techniques for representing the 3-D scene as the volumetric intersection applied to pseudo-octrees from silhouettes of the scene. Braccini *et. al.* [5] presented work on refining a 3-D volumetric model from calibrated multiple views and mapping the texture information on to the model. Higuchi *et. al.* [16] discussed a method for building 3-D models from range data using a deformable mesh and registering multiple views. The main difficulty with creating a 3-D model of the scene, and hence the backprojection-reprojection approach to AVG, is that of registering and combining the 2-D information to generate a 3-D model.

Since the end goal of AVG is to generate a particular 2-D view, it is not clear whether a full 3-D representation is needed. As such, many researchers have investigated a more direct approach to AVG in which new views are generated by exploiting certain invariants in the geometry of the problem. Ullman and Basri [44] presented an approach whereby a 3-D object may be represented as a linear combination of 2-D images. They showed that under orthographic projection, an object with smooth boundaries undergoing rotation, translation and scaling is simply the linear combination of a set of 2-D views. An alternative approach proposed by Shashua [36] estimates projective structure of the 3-D scene from the epipolar geometry of two uncalibrated images and then reprojects the points to generate the desired view. Notice that in this case, a projective model instead of a full 3-D Euclidean model is determined. Similarly, Laveau and Faugeras [24] described a method in which a set of weakly or fully calibrated views of an object are used directly to predict the new view. The epipolar geometry of the views is again exploited. Although these approaches lead to reasonable results, the only points that can be reconstructed are the points that lie in the intersection of the views. Hence, points that do not appear in every view cannot be reconstructed directly. This poses a problem for regions that become deoccluded in the scene, typically present in more complicated 3-D scenes.

The third class of AVG algorithms attempts to deal with occluded/deoccluded regions in the scene better than the direct method while not resorting to a full 3-D representation. Generally, a set of $2\frac{1}{2}$-D surfaces is first estimated and then combined to generate the desired view. In [8], Chen and Williams provided an approach to generate intermediate views between two or more given views by establishing correspondence and interpolating correspondence information when necessary. They assumed however that relative camera transformation and depth information are both known *a priori*. Skerjanc and Liu discussed a trinocular system in [37, 26] which estimates disparities and synthesizes intermediate pictures. The disparities of edge segments are tracked through time with known camera geometry. Szeliski [38] has developed interesting techniques for registering multiple images together to achieve very high resolution scenes to create a so-called image mosaics.

In comparing the three classes, the 3-D scene reconstruction-reprojection approach results in a much more complete representation of the scene, requiring less storage than the other two techniques. However, it requires a large amount of computations and the quality of the resulting images depends on the number of model parameters. Also, a high degree of accuracy for registration is necessary. On the other hand, the direct approach is perhaps more accurate than the other two, but it cannot handle more complicated scenes where occlusion and deocclusion occurs. This problem may be overcome by having a very large number of views of the scene, though storage requirements would then become troublesome. Hence there is a tradeoff between storage requirements and accuracy of the reconstructed views. The hybrid approach seems to be a good balance offering reasonable quality reconstructions with relatively low storage requirements.

Our work falls in this third category of AVG algorithms: We wish to develop an approach for representing 3-D objects and scenes without resorting to full 3-D modeling. In our previous work [7, 48], it was shown that good reconstructed images are obtainable by using an uncalibrated camera following a translational motion. In this paper, we again consider an uncalibrated set up where a camcorder scans the 3-D scene along a linear trajectory so that the optical axis of the camcorder is oriented in the same direction every time. While the geometry is similar to that discussed in [8] and [26], our work differs in that the exact camera position or the precise camera transformation between reference frames is unknown. Known camera position simplifies the estimation problem considerably and reduces the ambiguities associated with the correspondence problem discussed in Chapter 2. We consider the unknown camera location case since in practice one cannot ensure a specific camera trajectory with a hand-held camcorder. The approach designates certain locations along the trajectory as reference locations. The representation for the scene consists of intensity and depth information at these reference locations. Notice that we have chosen not to combine these depth estimates and backproject them to form a complete 3-D model of the scene since we believe that more errors would be introduced by doing so.

The paper focuses only on a 3-D object situated in real 3-D scenes, referred to as the "outside-in" problem. The images in this case capture different views around the object with surrounding background information. The opposite "inside-out" problem consists of representing a particular location, such as a room or a cubicle, without an object of focus; this case of AVG is beyond the scope of this paper. Notice that for the outside-in case, our algorithms do not make any distinction between the object in the foreground and the rest of the scene, i.e. no explicit segmentation stage is considered. The depth estimation stage will implicitly separate regions according to depth by finding points associated with the object to be closer to the camera than points associated with the background. Most of the previously mentioned works e.g. [26, 24, 44] consider a single object alone in 3-D space without any textured background, thereby isolating the object and making the reconstruction problem much simpler.

This paper is outlined as follows. Chapter 2 discusses the proposed representation algorithm consisting of intensity and depth maps at prespecified locations. With the representation, a desired view may be generated using the reconstruction algorithm presented in Chapter 3. Experimental results for a real world scene are examined in Chapter 4. They suggest that the representation/reconstruction algorithms produce excellent reconstructed views along horizontal trajectories and promising ones along vertical trajectories. Finally, Chapter 5 consists of a conclusion on this work and directions of future research.

Chapter 2

Compact Representation of Scenes

In the problem of arbitrary view generation, we are interested in characterizing a three-dimensional (3-D) scene by a compact representation and then reconstructing an arbitrary view of the scene. The class of scenes to be considered shall be restricted to "outside-in" scenes, those containing a primary object in the foreground with other objects defining the background. One may consider extensions of the proposed algorithms to inside-out scenes but it is outside the scope of this paper.

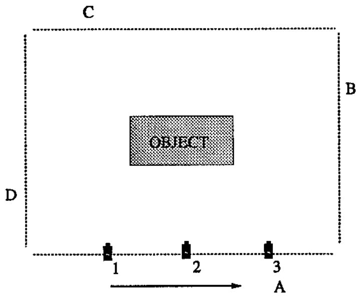


Figure 2.1: *An example of scanning geometry along four linear trajectories A, B, C, and D.*

To derive a compact representation for a given outside-in scene, a method must be devised for acquiring the necessary information. We propose the following scheme. Image sequences of an object are generated by translating a camcorder along a prespecified trajectory, e.g. a rectangular scanning pattern as shown in Figure 2.1. Translational motion is chosen to simplify the depth estimation stage as described in Section 2.1. Trajectories at different elevations or depths may also be scanned to add information about the scene. The image sequences typically consist of hundreds of frames. Note that no assumptions are made about the exact location of the camcorder or about the camcorder's internal parameters, i.e. the camcorder is uncalibrated. The motion is only approximately translational since the camcorder is moved by hand.

Given this framework, one possible representation consists of depth and intensity[1] infor-

[1] The images consist of luminance or intensity only. The chroma component of the images are discarded.

mation at specified viewpoints around the object called reference locations, e.g. locations 1, 2, and 3 for trajectory A in Figure 2.1. Assume that the reference locations are among the viewpoints originally scanned by the camcorder. Thus, only the depth information at these locations needs to be estimated to construct the representation.

Since the representation relies on depth information at the reference locations, we desire a depth estimation algorithm that results in dense information while ensuring fairly accurate estimates. Moreover, the algorithm should perform reasonably well for a general class of outside-in 3-D scenes. It would be beneficial for the algorithm to take advantage of the entire image sequence and somehow combine the useful information into a unified depth map.

There are many approaches to solving for the depth. Some approaches fall under the classification of optical flow (e.g. [18, 23]). Most optical flow algorithms rely on the so-called optical flow constraint equation [17] given by

$$\frac{\partial E}{\partial x}u + \frac{\partial E}{\partial y}v + \frac{\partial E}{\partial t} = 0 \tag{2.1}$$

where $E(x, y, t)$ is the irradiance at time t at image point (x, y) and (u, v) is the optical flow vector. Since Equation (2.1) provides only one relation for two unknowns, a second constraint is necessary to solve for the flow vectors. The results provide a dense flow field and are generally acceptable. The algorithms however work for only small motions and do not perform well across discontinuities without assuming local similarity.

A second class of approaches consist of stereo matching algorithms. With stereo algorithms, it is generally assumed that either camera positions or camera motion is known *a priori*. Typically, some additional information is furnished to aid in matching such as uniqueness and disparity constraints for random dot stereograms [28], a third view [20] or even more views [34], shading information [12], or different filtered outputs [22]. For a more complete review of stereo algorithms, see [11].

Other approaches are classified as solving the structure-from-motion (SFM) problem like [30, 39, 46, 40, 41]. For these algorithms, a set of features, e.g. edges in [40] and corners in [46], are identified and tracked. The motion of the camera and the structure of these features are then computed simultaneously. Despite the complexity of solving this nonlinear optimization problem under perspective projection, the SFM algorithms perform reasonably well given two or more arbitrary views. However, many times they are practical for only a small number of points in the scene. Moreover, many of these algorithms require point or feature correspondences in advance.

Our approach is similar to the above approaches for estimating depth at a selected reference frame. We take advantage of the information captured in an image sequence by first estimating depth between the reference frame and each neighboring frame and then combining several depth maps together to form a single accurate depth estimate. To solve for the local estimates of depth, we consider three different algorithms in Section 2.1. In each case, the correspondence problem is not assumed to have been solved. The geometry of the problem allows us to generate depth estimates directly from correspondence matches.

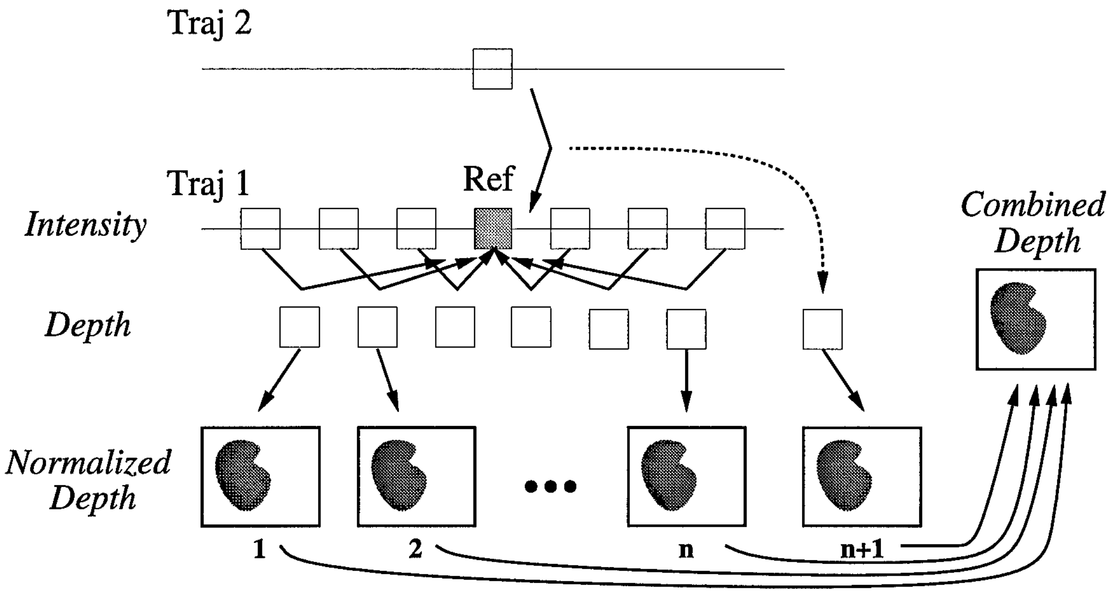


Figure 2.2: *A diagram of the complete representation process.*

The complete representation process is shown in Figure 2.2. To create an accurate depth map for a particular reference frame, local depth maps are estimated between neighboring frames and the reference frame as described in Section 2.1. The set of local depth estimates are then normalized to a common scale factor using the procedure outlined in Section 2.2. The normalized depth maps are combined in the manner presented in Section 2.3 to generate a reasonably accurate map consisting of some low confidence regions. These regions are filled in using cubic B-splines as discussed in Section 2.4. The end result is a dense depth map which, along with the corresponding intensity information, serves as the representation for the reference frame.

It is worth emphasizing that the entire representation process is really a preprocessing step. For a given 3-D scene, the representation may be determined off-line and stored in a database for later reconstruction. As such, computation time is not a crucial factor for this process. It would be desirable however to have a process which requires a reasonable amount of time. We shall see that the algorithms discussed in the following sections are fairly intensive but are still within reason.

2.1 Depth Estimation

For every reference frame, a set of local depth estimates is generated between the reference frame and each neighboring frame as shown in Figure 2.3. The depth estimation algorithm ideally should produce high quality results for each pair of frames, but because of difficulties in matching and artifacts in the routine, the reality may be far from the ideal. Lower quality depth estimates will suffice as long as the final depth map for the reference frame is of high quality. This may be ensured if each of the local estimates are fairly dense and generally accurate.

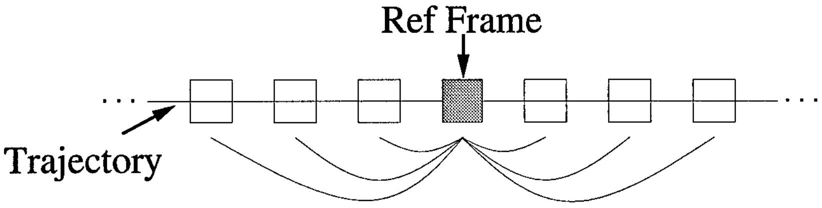


Figure 2.3: *Estimating local depth between reference frame and neighboring frames.*

Before depth estimation is possible, the correspondence between points in the two given views must be established. The general motion case is very difficult to treat since the correspondences, the motion, and the depth are all unknowns and thus the problem is nonlinear. It is possible to estimate depth iteratively in this case, but the results usually apply to only a small set of points in the scene. Instead, we shall consider a much simpler case: Suppose the two views are along the same linear trajectory, either horizontal along the x axis or vertical along the y axis, i.e. the optical axes of the camera at the two views are parallel. Then the motion is limited to one degree of freedom along the trajectory and the search for correspondences reduces to a 1-D search. Note that the problem becomes akin to the stereo problem with a fronto-parallel object and unknown baseline b. The epipolar lines associated with the two views are then parallel to the scan lines.

There is an interesting result [17] when two images are related by a horizontal motion.[2] Let $P = (X, Y, Z)$ be a point on the object in the 3-D scene shown in Figure 2.4. Suppose the world coordinate system is centered at the first image I_1 itself and the second view I_2 is $+b$ units away along the x axis from the origin. Assume the perspective projection model for the camera at the two views, that is, P corresponds to a point (u_1, v_1) in I_1 given by

$$(u_1, v_1) \;=\; \left(\frac{fX}{Z}, \frac{fY}{Z}\right), \tag{2.2}$$

where f is the focal length of the camera. Similarly for I_2,

$$(u_2, v_2) \;=\; \left(\frac{f(X+b)}{Z}, \frac{fY}{Z}\right) = \left(u_1 + \frac{fb}{Z}, v_1\right). \tag{2.3}$$

The disparity Δu shall be defined by

$$\Delta u \;\triangleq\; u_2 - u_1, \tag{2.4}$$

[2] A similar analysis may be made for the vertical case. Here, the disparity Δv will be defined as the difference between the vertical components.

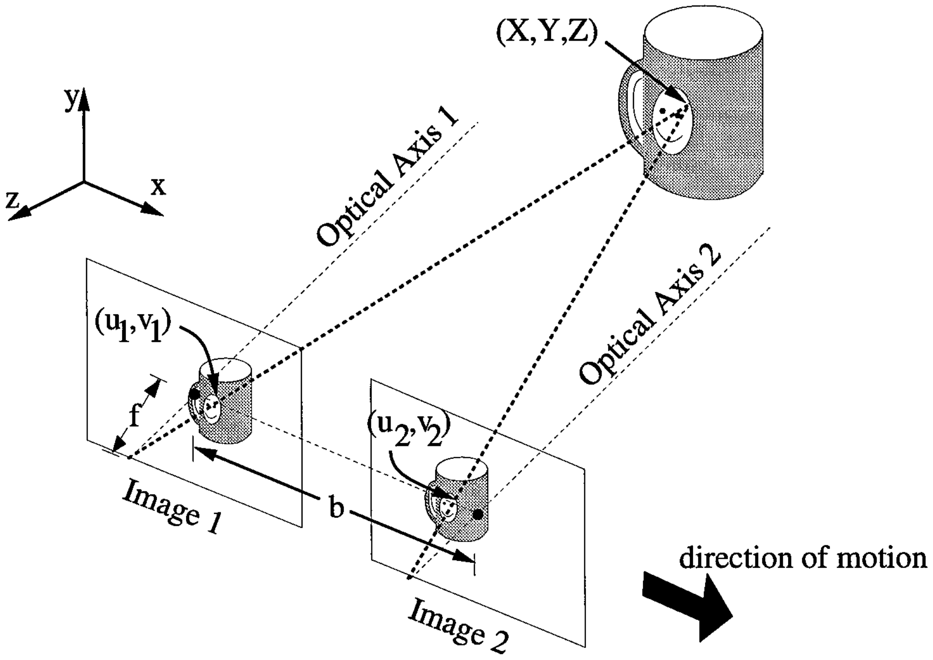


Figure 2.4: *Geometry of depth estimation problem from two fronto-parallel images under perspective projection.*

the apparent horizontal shift of the point P in the two images. Combining Equations (2.2), (2.3), and (2.4), the result becomes

$$\Delta u \;=\; u_2 - u_1 = \frac{fb}{Z}. \tag{2.5}$$

Recall that Z is the distance along the optical axis to the object or in other words, the depth from the point to the camera. We may conclude that in the fronto-parallel case, if correspondence between image points in the two images is solved, then the difference in the horizontal components is an estimate of disparity for the point. The disparity, in turn, is inversely related to the depth of the point. Notice that both f and b are unknown; for simplicity, we assume f equals 1. From this result, it may be observed that solving correspondence leads directly to an estimate of relative depth. We shall assume local perfect translation between every pair of images to reduce the depth estimation problem to a correspondence matching problem.

After examining various matching algorithms, we consider intensity-based matching approaches to be the most suitable for our purposes. Each of these approaches attempts to minimize some cost function in order to find the best match. The sum of squared differences (SSD) cost function appears to work well; Fua [12] and Hannah [14] both describe the performance of other cost functions and correlation measures for matching. The remainder of this section will focus on three correspondence matching algorithms: block matching algorithm, adaptive block matching algorithm, and fast adaptive block matching algorithm.

2.1.1 Block Matching Algorithm (BMA)

To solve the correspondence problem between two images, one may consider matching regions of intensities in the two images to generate a motion vector field [25]. The motion vector field provides the correspondence between points in the first image and those in the second. For every point (i,j) in the first image, the algorithm attempts to find the motion vector (m,n) which minimizes a cost function between points in neighborhoods around (i,j) in the first image and $(i+m, j+n)$ in the second. The cost function we shall use will be the squared difference of intensities. Let $I_1(\cdot,\cdot)$ and $I_2(\cdot,\cdot)$ be the intensities of the first and second images, respectively. Suppose B is the neighborhood centered around the given pixel and W consists of all possible motion vectors. Then the motion vector $(u(i,j), v(i,j))$ for point (i,j) is given by

$$(u(i,j), v(i,j)) = \min_{(m,n)\in W} \left\{ \iint_{(x,y)\in B} \left| I_1(x,y) - I_2(x+m, y+n) \right|^2 \, dx\, dy \right\}. \tag{2.6}$$

Solving the above equation for the best motion vector $(u(i,j), v(i,j)) = (m,n)$ is a nonlinear problem. Many attempts to address this nonlinear problem exist with varying degrees of success [25]. We shall focus on one well-known approach, the block matching algorithm (BMA) [32].

In BMA, it is assumed the region B consists of only a finite number of points; let B be a square of size $b \times b$ pixels. Moreover, the search window W of candidate vectors is also finite; W is restricted to a rectangle of $w \times l$ pixels.[3] With these in mind, Equation (2.6) reduces to a double summation known as the sum of squared differences (SSD), whereby the motion vector $(u(i,j), v(i,j))$ for point (i,j) may be obtained by

$$(u(i,j), v(i,j)) = \min_{(m,n)\in W} \left\{ \sum_{x=i-b/2}^{i+b/2} \sum_{y=j-b/2}^{j+b/2} \left| I_1(x,y) - I_2(x+m, y+n) \right|^2 \right\}. \tag{2.7}$$

The search is clearly two-dimensional. However, for strictly translational motion, the search reduces further to a one-dimensional search. The motion vector $(u(\cdot,\cdot), v(\cdot,\cdot))$ then becomes an estimate of disparity $d(\cdot,\cdot)$, and hence, an estimate of the reciprocal of depth. In the case of horizontal translation along the x axis, the equation becomes

$$d(i,j) = \min_{m\in L} \left\{ \sum_{x=i-b/2}^{i+b/2} \sum_{y=j-b/2}^{j+b/2} \left| I_1(x,y) - I_2(x+m, y) \right|^2 \right\} \tag{2.8}$$

where L is the appropriate epipolar line as drawn in Figure 2.5.

Despite the above simplifications, the search for a given disparity remains exhaustive for every pixel location along L, resulting in a computationally intensive algorithm on the order

[3] We have limited the motion vectors to be integer valued. Since the intensities themselves are specified only for pixel locations and since each depth map will later be combined, it is reasonable to consider only integral-valued vectors. The extension to subpixel motion vector estimation is straightforward by interpolation [32].

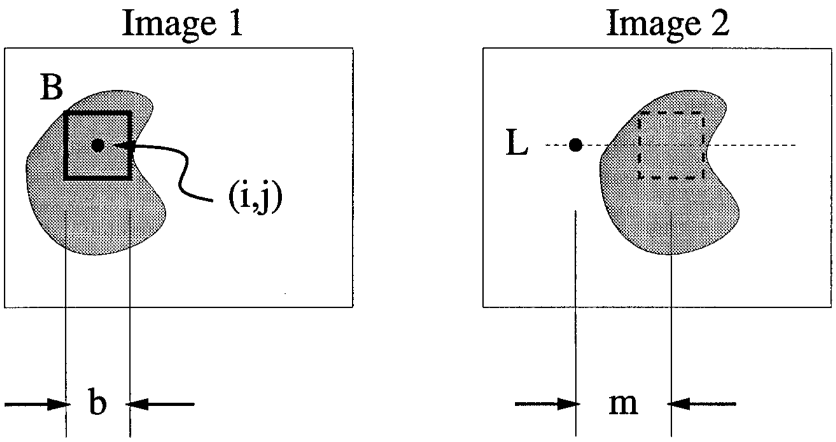


Figure 2.5: *Example of BMA for horizontal motion case.*

of $O(rclB^2)$, where r and c are the dimensions of the image, l is the length of the epipolar line, and B is the block size. While less computationally intensive matching algorithms exist, BMA produces a dense disparity map with reasonable results.

One primary assumption of BMA—and many region matching algorithms, for that matter—is that objects in the 3-D scene are rigid and move accordingly. Alternatively, the scene may be assumed to be stationary while only the camera is moving. Another assumption is that the intensities in corresponding neighborhoods of two images are roughly similar; imaging distortion of the camera is negligible.

For a pair of images that satisfy both assumptions, BMA generates adequate disparities for most points. However, there are some artifacts inherent in the algorithm and the problem itself that induce incorrect disparities for certain regions. One of these artifacts is called *aperture ambiguity*. Our definition of aperture ambiguity is the difficulty associated with matching horizontal lines in images related by a horizontal displacement. It arises because the block B used for matching is too small and does not include enough distinct features when matching.

A second artifact related to aperture ambiguity occurs in low texture regions; we refer to this condition as the *constant intensity ambiguity*. When few distinguishing features exist in the block B, it is difficult to estimate precise motion. Note that with both aperture ambiguity and constant intensity ambiguity, the SSD equation (2.7) is a shallow function over all possible disparities; the disparities are almost all equally good. The minimization depends largely on the actual intensity values, which may be noisy due to the imaging process and different lighting conditions. Despite the lack of distinct features, the matching algorithm may still lead to the correct disparity for horizontal edges and low texture regions.

In contrast, there are other artifacts of BMA which almost always produce the wrong disparity—these occur in occluded regions and near depth discontinuities. An occluded region is an area that appears in one image but not in the other. For instance, a moving object in the scene generally occludes some points and deoccludes other points from view.

In such regions, BMA blindly attempts to find the best match but fails miserably because only one image has information about the region. This artifact is referred to as *occlusion ambiguity*.

BMA also generates incorrect disparity information near depth discontinuities, an artifact known as *depth discontinuity localization ambiguity*. It is difficult to identify depth discontinuities of a scene beforehand, since the goal of BMA is to estimate depth. Intensity discontinuities are instead considered because it is not uncommon for depth discontinuities in the scene to be related to intensity discontinuities in the image. For points near object boundaries but not part of the object, the search block B is large enough to include some features of the object. In minimizing the intensity error for such a point, the matching algorithm yields a motion vector similar to the motion of the object itself. The end result is poor localization of the object boundary in the disparity domain by $b/2$ pixels, i.e. the object seems to have expanded in all dimensions. Clearly, the localization of depth discontinuities depends on the size of the block used for matching—the smaller the block, the better the localization. However, it is widely known that using blocks that are too small produces many false matches, since intensity patterns will be less distinctive [17].

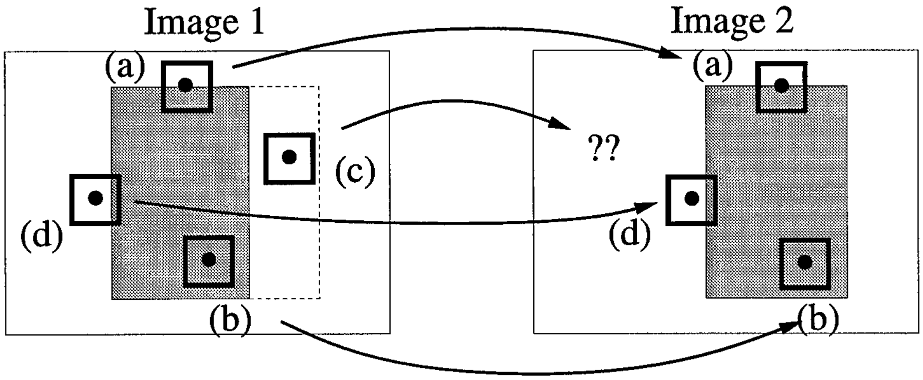


Figure 2.6: *Example of regions where BMA fails due to (a) aperture ambiguity, (b) constant intensity ambiguity, (c) occlusion, and (d) depth discontinuity localization ambiguity.*

An example of all four artifacts is shown in Figure 2.6. The two images shown are related by horizontal translational motion, i.e. the two optical axes are parallel to each other and perpendicular to the direction of motion. The object is a rectangle of constant grey while the background is entirely white. If BMA is used to match points in Image 1 with those in Image 2, the aforementioned problems will lead to incorrect disparity estimates. Mismatches at horizontal line segments identified as (a) are due to aperture ambiguity. Constant intensity ambiguity occurs in both the foreground and background as with the point indicated by (b). Little information may be obtained in occluded regions like (c). Problems with localizing the depth discontinuities is shown with (d).

One may observe that the size of block B affects the effectiveness of BMA immensely. If B is small, e.g. 5×5, the regions near depth discontinuities in the scene will possess correct disparities while most of the horizontal edges and untextured regions will have faulty disparity estimates. On the other hand, if B is large, e.g. 16×16, the disparities along edges and in constant intensity regions will be improved at the expense of poor boundary localization and larger computation time needed. A choice in between, e.g. 9×9, yields fair results. It is clear that BMA alone cannot adequately estimate disparities and that modifications to the algorithm are necessary to produce the desired goal of a dense and reasonably accurate disparity map.

2.1.2 Adaptive Block Matching Algorithm (ABMA)

If regions where the discussed artifacts occur may be located, the matching algorithm would be able to disregard the disparity information associated with them. The resulting disparity map would consist of regions of varying confidence levels, high confidence regions with reasonable estimates of disparities and lower confidence regions with potentially incorrect estimates. Furthermore, if the matching algorithm is modified so that it is capable of correcting the discussed artifacts, most disparities will become much more accurate. These confidence levels are a simple labeling of the different disparity estimates. Anandan proposed a different notion of confidence levels based on the curvature of the SSD surface; see [2] for details.

Identifying Confidence Regions

It is straightforward to identify most of these artifacts and subsequently assign confidence levels. For aperture ambiguity, detecting horizontal edges in images related by horizontal motion is the first step.[4] A gradient-based edge detector is used to generate the desired edge information [21, 25]. The convolution of the image and two edge operators $G_{\text{vert}}(\cdot,\cdot)$ and $G_{\text{horiz}}(\cdot,\cdot)$ defined by

$$G_{\text{vert}}(x,y) = -\frac{x}{2\pi\sigma^4}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.9}$$

$$G_{\text{horiz}}(x,y) = -\frac{y}{2\pi\sigma^4}e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.10}$$

the first partials of a 2-D Gaussian filter, gives horizontal and vertical gradient information, respectively. Thresholding these two convolved outputs results in the location of edge pixels. Since aperture ambiguity occurs only for line segments longer than the block size, edge pixels are discarded if they do not span an entire block length b. Furthermore, blocks which contain both horizontal and vertical edge pixels are not marked since the vertical edge resolves the ambiguity of horizontal motion. Once all suitable line segments have been found, the final step involves declaring points within $\pm b/2$ of any horizontal line segment as an aperture ambiguous point (AP).

Identifying constant intensity regions is a much simpler task. For every point, if the neighborhood around the point has an intensity variance lower than a prespecified threshold,

[4] For the remainder of this subsection, we shall assume that the images we wish to match are generated by horizontal motion. In the case of vertically related images, the described approach may be modified to locate vertical segments.

the point is considered a constant intensity ambiguous point (*CONST*). This technique is called *Local Variance Thresholding* or LVT. A low variance suggests that the block consists of low texture and nearly constant intensity. As the variance threshold decreases, the block associated with *CONST* becomes more uniform in intensity.

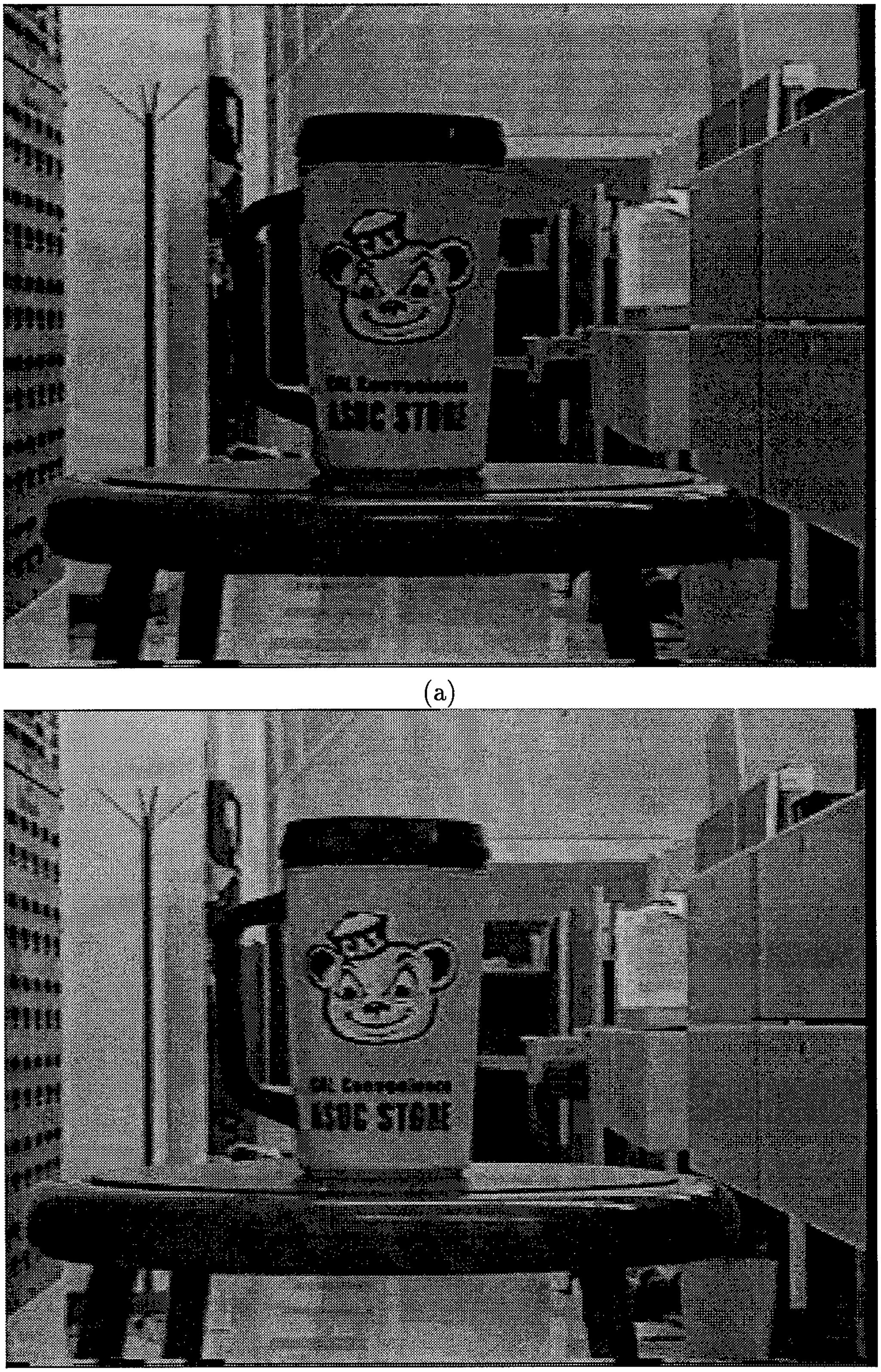
To detect occluded regions (*OCCL*) in the scene, an interesting observation can be made. When matching one image to another, BMA produces a mapping from all points in the first image to points in the second image. The regions in the second image that do not correspond to any points in the first image are precisely the occluded regions in the second image. Hence, to find the occluded regions in the first image, the matching should also occur in the reverse direction, from the second image to the first [45]. Because of possibly noisy matches, an additional filtering step may be necessary.

Another advantage of performing the match in both directions is that invalid matches due to other reasons may be found and discarded. If a point (i, j) in one image has been found to map to a point $(i+m, j+n)$ in the second, i.e. $I_1(i,j) \Longrightarrow I_2(i+m, j+n)$, then it is expected that the reverse will also be true, $I_2(i+m, j+n) \Longrightarrow I_1(i,j)$. If, however, the reverse mapping is not true, then the match is said to be inconsistent. Matching in both directions helps *validate* matches and identify inconsistent matches marked as *INCONS* to be pruned [12]. To compensate for noise in the matches, a match is said to be valid if its reverse mapping lies within a ±2 pixel window around the match.

An example of the block matching algorithm is shown in Figures 2.7 and 2.8. Figure 2.7 (a) and (b) are Frames #37 and #34, respectively from the so-called "Mug2" sequence; see Chapter 4 for more details. The disparity map of Frame #37 is estimated by matching from Frame #37 to Frame #34; the apparent object motion is to the left. A 9×9 block is used for finding the best match at every point. The disparity maps are shown in Figure 2.8. There are sixteen levels of possible horizontal disparity, and thus sixteen levels of grey scale. White signifies the largest disparity; black indicates zero disparity. In (a), the mug and stool are clearly identified. However regions near the front of the stool and the far wall consist of many incorrect disparities. The top of the mug and other horizontal edges are mismatched as are regions near the mug boundary. The second disparity map (b) shows the disparities along with four types of problem areas marked in different colors: blue, *CONST*; red, *AP*; yellow, *INCONS*; green, *OCCL*. Most of the incorrect disparities have been identified, leaving a much more accurate, albeit sparse, disparity map.

Correcting for Each Confidence Region

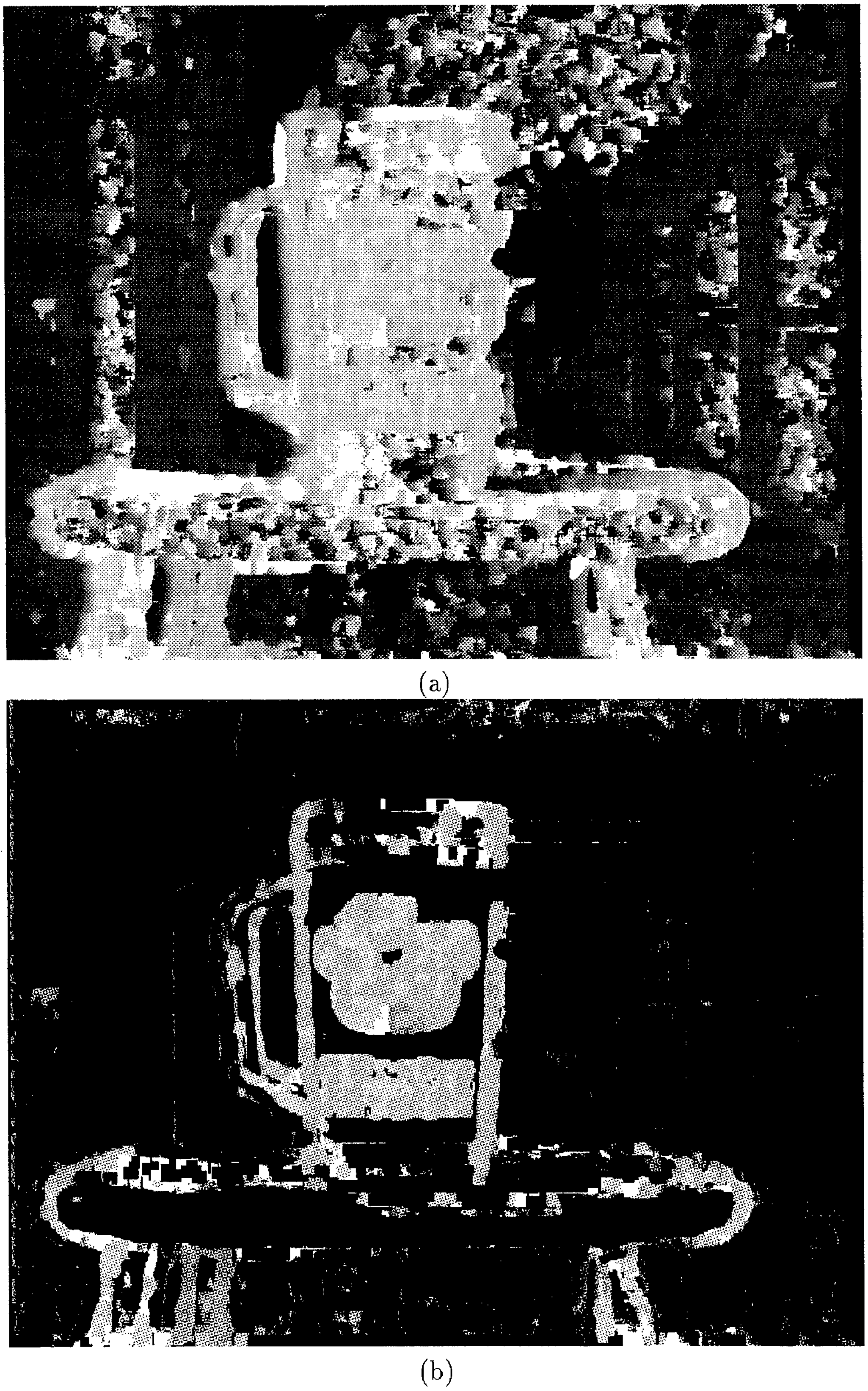
The goal is to produce a set of disparity estimates between a reference frame and each neighboring image, ultimately to generate a single accurate and dense depth map. It is possible then to use information from other disparity maps to "fill in" regions of low confidence. For instance, an *OCCL* point in one map may be unoccluded in another map. Disparity estimates from a vertically related pair of images will have vertical aperture ambiguity, but not horizontal aperture ambiguity, so it may be used to replace the lower confidence *AP* points; this fact will be considered in the combination algorithm of Section 2.3. Inconsistencies in one map may not exist in another and can replace the *INCONS* points. It is impossible to



(a)

(b)

Figure 2.7: *Two images from "Mug2" sequence: (a) Frame #37, (b) Frame #34.*



(a)

(b)

Figure 2.8: *Example of disparity estimate from matching Frame #37 with Frame #34 (a) without confidence regions, (b) with confidence regions. Legend: blue, CONST; red, AP; yellow, INCONS; green, OCCL.*

force each individual disparity map to be perfect, however we may not require each one to be so as long as the final depth map is accurate and dense.

There are two artifacts which are seemingly uncorrectable: the constant intensity ambiguity and depth discontinuity localization ambiguity. In the former, a larger search block is needed to properly match the *CONST* regions. Unfortunately, a larger block may contribute to the second problem of poor localization of depth discontinuities. It is difficult to precisely locate depth discontinuities in the scene since the object is not known *a priori*. Alternatively, a smaller block could be used from the start to help improve localization. However this will introduce more problems in *CONST* regions. Hence, a fixed block size for the entire image is incapable of reducing both artifacts.

We propose a modification to BMA which adapts to the confidence regions and provides more accurate information in *CONST* regions, near many depth discontinuities, and even along *AP* edges. The approach consists of dividing the image into *CONST* and non-*CONST* regions and finding the best matches for both regions. To identify *CONST* regions, the image undergoes low-variance thresholding with a 3×3 block. Generally, a small block size is preferred since textured regions near or along intensity discontinuities will be better localized.[5] This tends to improve the localization of depth discontinuities since many times intensity edges are related to depth ones. Once found, the non-*CONST* points are then matched using Equation (2.8).

With the non-*CONST* points matched, the next step is to find the best match for each *CONST* point (i, j).[6] Since the main ambiguity stems from using a block that is too small, we consider instead using the largest rectangular block containing the point (i, j) that consists entirely of *CONST* points. Note that the block does not have to be centered at (i, j). One way to find such a block is by growing a 3×3 block around (i, j) and then extending each side evenly until the extension encounters at least one non-*CONST* point or until some prespecified dimension maximum, i.e. block size limit, has been reached. In this way, the algorithm utilizes the shape and relative size of the *CONST* region without including too many features which may mislead the algorithm. Figure 2.9 shows an example of the image in Figure 2.6. Point P is a *CONST* point near the rectangular object but part of the background. With BMA, P would be assigned the same motion vector as that of the object. On the other hand, if LVT is performed and the block shown in the figure is used for matching, then the correct motion vector would be estimated.

[5] Notice that LVT with a small block size is similar to detecting coarse intensity edges. It is possible to perform edge detection directly instead of LVT to identify the *CONST* regions. A point is declared *CONST* if it is not an edge pixel since only textured points are captured in the edge information. This approach, however, results in fewer non-*CONST* points to be matched than with LVT, thereby increasing computation costs. In addition, many textured regions that are not edge points will be classified as *CONST* by an edge detection procedure. With LVT, these points will be marked as non-*CONST* and may be subsequently matched in this first step.

[6] One may consider using the same approach to find disparity information for *AP* regions by choosing blocks consisting of only *AP* points. However the disparities are not guaranteed to be accurate in these regions and thus may lead to lower quality reconstructed images. A much better approach would be to use vertical information to replace the incorrect *AP* points.

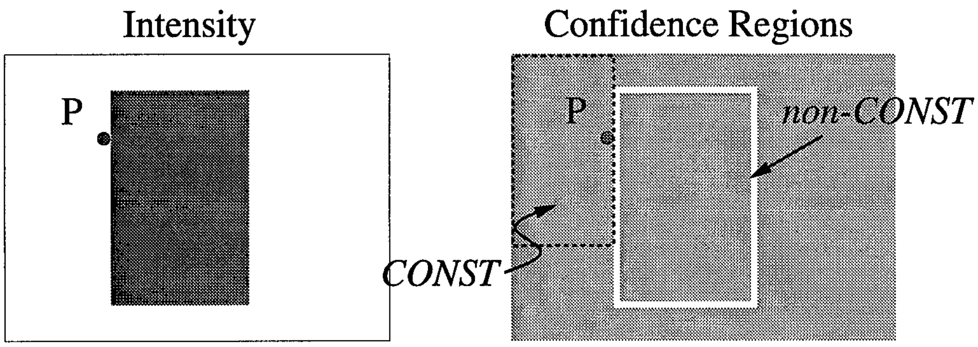


Figure 2.9: *Example of intensity image and its corresponding confidence regions after LVT with a* 3×3 *block has been performed. Notice that other regions have not been identified for clarity.*

Adaptive Block Matching Algorithm

The matching algorithm may be improved from standard BMA if it is able to locate and correct for the different artifacts inherent with BMA. The methods described above together form the so-called adaptive block matching algorithm (ABMA). The steps to match two images I_1 and I_2 using ABMA are as follows:

Algorithm (ABMA):

1. Perform LVT using 3×3 blocks on I_1 to identify *CONST* points.

2. Apply 1-D median filter to I_1 in both horizontal and vertical directions to binary image composed of *CONST* and non-*CONST* points (see below).

3. Locate *AP* points in I_1 using standard block size, e.g. 9×9.

4. Evaluate matches from I_1 to I_2 using 9×9 blocks for all non-*CONST* and non-*AP* points.

5. For every *CONST* point in I_1, find the largest rectangular block that consists strictly of *CONST* points and use this for matching. If the block size does not exceed a given threshold, assign match and mark the *CONST* point as corrected.

6. Repeat steps 1 to 5 for I_2.

7. Mark the unmapped regions from matching I_1 and I_2 as *OCCL*.

8. Validate each point by ensuring its match in one direction lies within a ±2 pixel window around its reverse match; invalid points are marked as *INCONS*.

The final result consists of fairly dense and reasonably accurate disparities. Further, there will be regions of low confidence marked as *OCCL*, *INCONS*, corrected *AP*, and corrected *CONST*in the depth map.

ABMA improves disparity estimates in *CONST* regions and localizes many depth discontinuities to within one pixel. However, as with every algorithm, there are a couple of problems with the approach. One problem pertains to the increased computation time to match two images. Since very large blocks may be used for matching, the required computation time increases severalfold. Let N_{const} be the number of *CONST* points in a given image and let N_{ap} be the number of *AP* points. Suppose the image has dimensions $r \times c$. Then, ignoring time for locating the confidence regions, the computation time is on the order of $O((rc - N_{const} - N_{ap})lB^2 + N_{const}l\bar{W})$, where l and B are defined as before, and $\bar{W}$ is the average area of all the rectangular blocks. A faster alternative is proposed in Section 2.1.3.

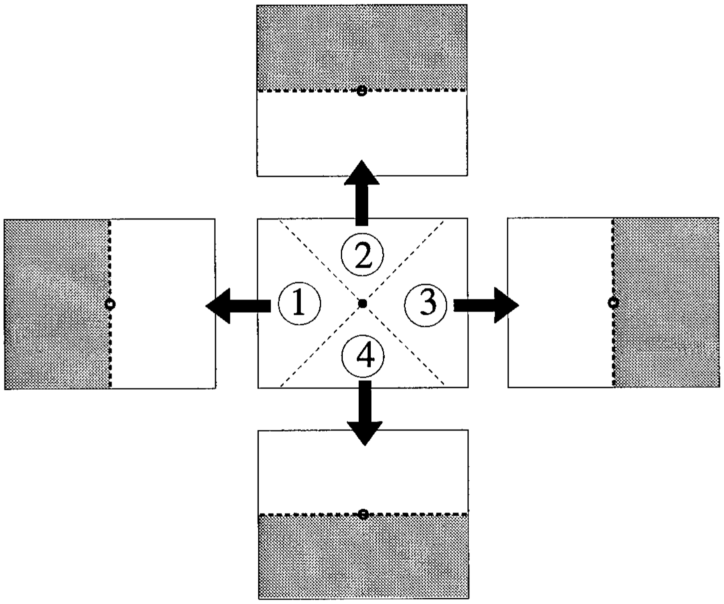


Figure 2.10: *Example of a single-impulse image used with ABMA. The image can be divided into four regions, each associated with the blocks shown. If disparities for each of the four regions are different, the resulting disparity map will exhibit a diagonal nature of the algorithm.*

A more significant problem involves a diagonal artifact for constant intensity points inherent to ABMA. Consider an image of a single point in the center drawn in Figure 2.10. Applying LVT to this image will result in identifying every point except the central point as *CONST*, since they all possess the same intensity. To match each *CONST* point, the largest block is found by growing every side initially at the same rate until a non-*CONST* point is encountered. Assume for now that the dimension maximum does not affect the size of the block. It should be obvious that the image will be separated into four regions, labeled 1 through 4 in the figure. Because of the growing nature of ABMA, all points in a given region correspond to the same block drawn in grey consisting of one half of the image. Hence, the disparity is the same for all points in the same region. If the disparities of adjoining regions are different, then the disparity map will exhibit a peculiar diagonal artifact. We believe the artifact can be avoided by using a less restrictive shape, in contrast to the rectangular blocks, for matching; we are currently investigating this problem.

To reduce the impact of the diagonal nature of ABMA, it is necessary to remove all isolated *CONST* and non-*CONST* points in the image. If we consider the binary image of *CONST* and non-*CONST* points, then the problem can be treated as the removal of salt-and-pepper noise. A common technique to resolve this problem is to apply a median filter. The binary image is filtered by a three-point 1-D median filter in both horizontal and vertical directions; the 2-D median filter was not used because it does not preserve corners [25].

An example of using ABMA is given in Figure 2.11(a). The disparity estimate shown was created from matching Frame #37 with Frame #34. Note that this corresponds to the BMA disparity map in Figure 2.8(a). The color legend is similar to that of Figure 2.8(b), i.e. blue for uncorrected *CONST* points, red for uncorrected *AP* points, yellow for *INCONS*, and green for *OCCL*. The mug and stool have been identified and their boundaries are better localized. The front of the stool corresponds to a more uniform and accurate disparity value. The far wall is found to be farther away from the camera than the mug/stool, as expected. While these regions look quite good, the diagonal artifacts and low confidence regions are very striking. However the disparity values are not unreasonable. Comparing Figure 2.8(a) with Figure 2.11(a), ABMA clearly exceeds the performance of BMA.

2.1.3 Fast Adaptive Block Matching Algorithm (FABMA)

As mentioned, one problem with ABMA is the amount of time needed to compute the matches for every point. This problem is especially severe for images with a large percentage of *CONST* points, where the corresponding block sizes are big. We propose a modification to ABMA which speeds up processing and attempts to retain all of the advantages of ABMA; we refer to the modified algorithm as the fast adaptive block matching algorithm or FABMA.

To reduce computation time, one may observe the redundancy of ABMA in assigning the same disparities for *CONST* points in the same general area (recall Figure 2.10). FABMA exploits this redundancy by assigning the same disparity to every *CONST* point in a given block. Suppose (i,j) is a *CONST* point with the corresponding block defined by $B(i,j) = [i-x1, j-y1] \times [i+x2, j+y2]$, where $x1, y1, x2, y2$ represent the distance from the point (i,j) to each one of the four block boundaries. Notice that the block dimensions are $(x1 + x2) \times (y1 + y2)$. Let d be the resulting disparity estimate for (i,j). Then every point in $B(i,j)$ is assigned the same disparity d and no matching occurs for any of these points—they are unmatched. Let the set of *CONST* points be given by S_{const}. Suppose S_{const} is the sum of $S_{matched}$ and $S_{unmatched}$ where $S_{matched}$ is the set of *CONST* points matched by FABMA and $S_{unmatched}$ is the set of unmatched *CONST* points. If N with the appropriate subscript denotes the number of elements in the given set, then clearly, $N_{const} = N_{matched} + N_{unmatched}$.

Furthermore, let $\bar{W}_{matched}$ be the average area of the rectangular blocks corresponding to the points in $S_{matched}$, i.e.

$$\bar{W}_{matched} = \frac{1}{N_{matched}} \sum_{(i,j)\in S_{matched}} B(i,j) \tag{2.11}$$

Then the computation time for FABMA may be expressed as $O((rc - N_{const} - N_{ap})lB^2 + N_{matched}l\bar{W}_{matched})$. Recall that B is the block size used to match the non-*CONST* points and l is the length of the search line. This complexity is considerably lower than that obtained for ABMA. With FABMA, we sacrifice accuracy for computation time. However, as we shall see, the depth estimates do not have to be exact for our ultimate goal of reconstruction.

With this modification in mind, the steps for FABMA are outlined as follows:

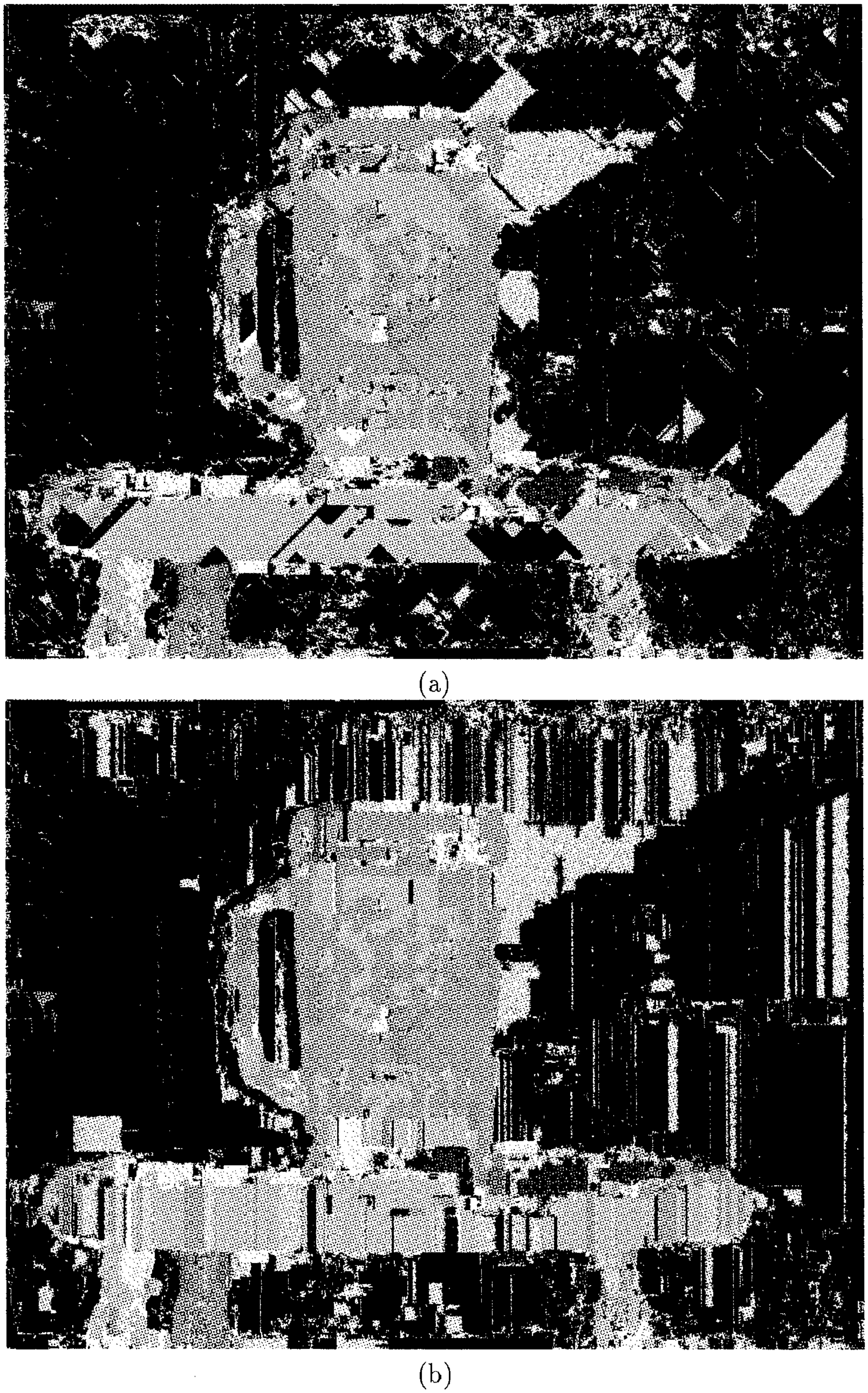
Algorithm (FABMA):

1. Perform LVT using 3 × 3 blocks on I_1 to identify *CONST* points.
2. Apply 1-D median filter to I_1 in both horizontal and vertical directions to binary image composed of *CONST* and non-*CONST* points.
3. Locate *AP* points in I_1 using standard block size, e.g. 9 × 9.
4. Evaluate matches from I_1 to I_2 using 9 × 9 blocks for all non-*CONST* and non-*AP* points.
5. For every unmarked *CONST* point in I_1, find the largest rectangular block that consists strictly of *CONST* points and use this for matching. If block size does not exceed a given threshold, assign match and mark *every* *CONST* point in the block as corrected.
6. Repeat steps 1 to 5 for I_2.
7. Mark the unmapped regions from matching I_1 and I_2 as *OCCL*.
8. Validate each point by ensuring its match in one direction lies within a ±2 pixel window around its reverse match; invalid points are marked as *INCONS*.

Notice that only step 5 differs from ABMA.

It may be easy to see that FABMA does not suffer from the diagonal artifacts produced by ABMA. The reason is that not every *CONST* point is explicitly matched. However, FABMA does introduce horizontal and vertical artifacts because every point in a given block is given the same disparity. It is not clear whether horizontal/vertical artifacts are more acceptable than diagonal ones, however the author believes that the former appears to be more pleasing to the eye.

An example will illustrate the effectiveness of FABMA. In Figure 2.11(b), FABMA was used to match Frame #37 with Frame #34 of the "Mug2" sequence. Note the color legend



(a)

(b)

Figure 2.11: *Example of disparity estimate from matching Frame #37 with Frame #34 using (a) adaptive BMA, (b) fast adaptive BMA.*

is the same as before. As with ABMA, FABMA produces a depth map that estimates the true 3-D scene quite well. The depth of the stool and mug have been recovered very well as well as the walls in the background. There are prominent vertical artifacts but they do not pose a significant problem at this stage.

We observe that with both ABMA and FABMA, the resulting depth map appears to be more accurate than that obtained by BMA. It should be clear that not every point has a correct depth estimate and that there are large areas of low confidence in the depth map. However, low confidence regions in themselves do not introduce any problems since other depth maps may be used to fill in these regions. In fact, it is preferred to have low confidence point than an incorrect one.

2.2 Normalization of Initial Estimates

After solving correspondence for every pair of frames, we obtain a set of depth maps with regions of varying confidence levels. Each of these depth maps differ in scaling factor because of the difference in disparity. Therefore each depth map should be adjusted so that they are all related by the same factor.

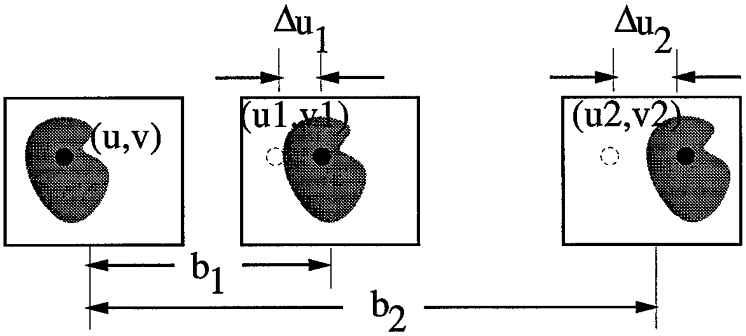


Figure 2.12: *Exploiting geometry of camera set up to normalize depth maps.*

There are a couple of possibilities for this task. The first one is a linear regression of points from different depth maps corresponding to the same physical 3-D points [7]. Another approach is to estimate the translation parameter between maps and scale by the reciprocal. As described before, a point (u_i, v_i) in one image and $(u1_i, v1_i)$ in a second horizontally translated image are related by the disparity equation

$$\Delta u_{1,i} = u1_i - u_i = \frac{f}{z_i} b_1 \tag{2.12}$$

where b_1 is the translation parameter relating the two images. If a third image is introduced, one yields a similar equation

$$\Delta u_{2,i} = u2_i - u_i = \frac{f}{z_i} b_2 \tag{2.13}$$

with b_2 the translation parameter linking the first and third images; Figure 2.12 shows this relationship. Note that the depth z_i is the same in both cases since all three image points correspond to the same physical point. Combining Equations (2.12) and (2.13) leads to the following relation

$$\frac{\Delta u_{1,i}}{\Delta u_{2,i}} = \frac{b_1}{b_2} \tag{2.14}$$

or

$$\Delta u_{1,i}\frac{b_2}{b_1} = \Delta u_{2,i} \tag{2.15}$$

Suppose now we consider k high confidence disparity points common to the two depth maps. For each point i, Equation (2.15) holds, thus leading to the matrix equation

$$\underbrace{\begin{bmatrix} \Delta u_{1,1} \\ \Delta u_{1,2} \\ \vdots \\ \Delta u_{1,k} \end{bmatrix}}_{A} \frac{b_2}{b_1} = \underbrace{\begin{bmatrix} \Delta u_{2,1} \\ \Delta u_{2,2} \\ \vdots \\ \Delta u_{2,k} \end{bmatrix}}_{y} \tag{2.16}$$

By linear least squares, we may solve Equation (2.16) for the ratio b_2/b_1 to get

$$\frac{b_2}{b_1} = (A^TA)^{-1}A^Ty \tag{2.17}$$

or, in terms of sums,

$$\frac{b_2}{b_1} = \frac{\sum_{i=1}^{k}(\Delta u_{1,i})(\Delta u_{2,i})}{\sum_{i=1}^{k}(\Delta u_{1,i})^2} \tag{2.18}$$

The scaling factor of the depth maps is arbitrary since each map is an estimate of relative depth. However, it is important that they are scaled to the same scaling factor. Without loss of generality, we may set $b_1 = 1$. Then b_2 is precisely the scaling factor α by which we need to adjust the second depth map.

An iterative process may be used to reduce the error $\|A\alpha - y\|_2$ to some desired amount. During every pass, outlier points greater than a given error percentage are disregarded when computing α. The procedure converges when the number of points does not change between iterations. This modification helps to further improve the accuracy of the scaling factor. In our experiments, we use a generous error of 30% since the vector y consists of possibly erroneous data. The algorithm typically converges in only three iterations.

The above least squares problem is a very stable one since A is a one-dimensional vector. As such, it is straightforward to compute the scale factor α by keeping a running sum of numerator and denominator. In the absence of noise, this formulation is exceptionally robust since k is very large, typically in the tens of thousands.

This normalization process is repeated to compute the translation parameter between the reference frame and its n neighboring frames to find the corresponding scale factors. One may consider a multidimensional extension in which the scaling factor α for multiple depth maps are considered simultaneously. However, this problem will be less well-conditioned than the one-dimensional counterpart and will require much more computation time. It is possible to simplify computations using a recursive least squares technique that updates the QR factorization of the matrix A as more data points are considered [13].

2.3 Combination of Multiple Depth Maps

Once all the depth maps have been normalized to a common scaling factor, they may be combined to form a single depth map for a particular reference frame. Since each local depth map may consist of areas of low confidence areas and incorrect depth data, the combination process should retain only the information which seems consistent, otherwise it should regard the information as invalid.

Let $D_i(\cdot,\cdot)$ for $i = 1, 2, \ldots n$ denote the n normalized depth maps and let $D(\cdot,\cdot)$ represent the combined result. For every point (x, y), we may regard the problem as an estimation problem, i.e. given n votes for $D(\cdot,\cdot)$, determine the most accurate value. One possibility is to implement an iterative procedure which analyzes the statistics of the given data, throws out outliers, and reduces the data set to a more consistent one. It is common to compute the mean m and standard deviation σ of the points and discard points that lie outside an "interior" range around the mean, e.g. $m \pm k\sigma$. This type of approach works well for large sets of data. However, since n is rather small in our case, the effect of outlier depths on m and σ is much greater, raising the interior range too high to include some outlier points. Also, the depth information exhibits a predominantly bimodal distribution, i.e. many of the depths may be classified as foreground or as background. Generally, the depth associated with the cluster consisting of the majority of points is reasonably correct. To take advantage of this fact, we consider using the *median* instead of the mean to define the interior range as $\text{med} \pm k\sigma$. The effect is that one cluster of the bimodal distribution of depths is discarded; the underlying majority in depths value wins. For our purposes, k is set to 1.

An example is drawn in Figure 2.13. Here, six depth maps contribute estimates for the depth at some point (i, j). It would be reasonable to choose a smaller depth for this point since four of the six depth "votes" indicate a small depth. If the mean is used to determine the center of the interior range, one of the outlier depths is included and may throw off the estimation. On the other hand, if the median is used, the center of the range corresponds to point 4 and both outliers are subsequently removed.

As discussed before, depth information from horizontal matches contain artifacts along horizontal edges due to horizontal aperture ambiguity AP. If only these depth maps are used in combination, then there will be considerable problems in AP regions. To circumvent the problems, we propose including information derived by matching a vertically related pair of images, that is, using corresponding images from two linear trajectories at different elevations (see Figure 2.2). If the second image with respect to the reference frame is a perfect vertical

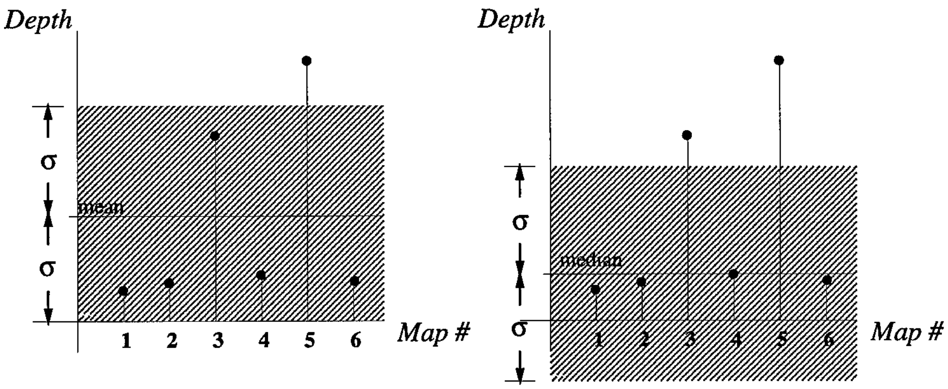


Figure 2.13: *Examples of using mean and median to throw out outliers.*

translation, then solving correspondence leads directly to an estimate of depth. Observe that the depth map will have vertical aperture ambiguity and will contain occluded regions generally not coincident with those found in the horizontal matches. Hence this information may be incorporated in the combining stage to improve the accuracy of the depth map in *AP* regions.

The algorithm may be further refined by introducing the notion of weights to the depth data. At every stage in the representation process, confidence levels are assigned based on the validity of the data. It is thus quite intuitive to weigh points in the combination stage based on these confidence levels. As shown in Table 2.1, more weight is placed on the vertical information since it is more reliable in certain regions. The reader may observe that lower confidence *AP*, *OCCL* and *INCONS* points are not included during combination whereas *CONST* points are considered since they are seemingly correctable.

Confidence Level	Horizontal	Vertical
FINE	1.0	1.0
CONST	0.3	1.0
otherwise	0.0	0.0

Table 2.1: *Weights used in combining together depth information of different confidence levels from both horizontal and vertical matches.*

The weight may also be taken into account during the statistical analysis. For depth maps D_i, $i = 1, \ldots, n$, the mean m and standard deviation σ are both easily extendible to their weighted counterparts as follows:

$$m = \frac{\sum_{i=1}^{n} w(i) D_i}{\sum_{i=1}^{n} w(i)} \tag{2.19}$$

$$\begin{aligned}
\sigma &= \left(\frac{\sum_{i=1}^{n} w(i)\{D_i - m\}^2}{\sum_{i=1}^{n} w(i)}\right)^{\frac{1}{2}} \\
&= \left(\frac{\sum_{i=1}^{n} w(i)D_i^2}{\sum_{i=1}^{n} w(i)} - 2\frac{\sum_{i=1}^{n} w(i)D_i m}{\sum_{i=1}^{n} w(i)} + \frac{\sum_{i=1}^{n} w(i)m^2}{\sum_{i=1}^{n} w(i)}\right)^{\frac{1}{2}} \\
&= \left(\frac{\sum_{i=1}^{n} w(i)D_i^2}{\sum_{i=1}^{n} w(i)} - 2m\frac{\sum_{i=1}^{n} w(i)D_i}{\sum_{i=1}^{n} w(i)} + m^2\frac{\sum_{i=1}^{n} w(i)}{\sum_{i=1}^{n} w(i)}\right)^{\frac{1}{2}} \\
\sigma &= \left(\frac{\sum_{i=1}^{n} w(i)D_i^2}{\sum_{i=1}^{n} w(i)} - m^2\right)^{\frac{1}{2}} \qquad (2.20)
\end{aligned}$$

where $w(i)$ is the weight of the depth D_i given by Table 2.1. The weighted median can be interpreted in two ways. First, let w denote half of the sum of the n weights i.e. $w = \frac{1}{2}\sum_{i=1}^{n} w(i)$. Let $s(j)$ be the partial sums of the first j weights, that is, $s(j) = \sum_{i=1}^{j} w(i)$. Suppose the list of depth data D_i for $i = 1, 2, \ldots n$ are sorted in ascending order and D_i is the first depth whose corresponding partial sum $s(i)$ is greater than or equal to w. Then the weighted median is simply

$$\text{med} = \begin{cases} D_i & \text{if } w < s(i) \\ \frac{D_i + D_{i+1}}{2} & \text{if } w = s(i) \end{cases} \qquad (2.21)$$

Alternatively, the weighted median can be thought of as the median of an augmented sequence $\hat{D}$ of depth data whereby every depth point D_i is replicated proportional to its weight, i.e.

$$\text{med} = \text{median}\left\{\hat{D}_i,\ i = 1, 2, \ldots m\right\} \qquad (2.22)$$

Notice that both views of the weighted median lead to the same result.

Incorporating the local depth estimates including the vertical estimate from a higher trajectory, the following algorithm may be used to determine the depth $D(\cdot,\cdot)$ for a given point (x, y) from depth maps $D_i(\cdot,\cdot)$ for $i = 1, 2, \ldots n$:

Algorithm (Combination):

1. Examine $D_i(x, y)$ for $i = 1, 2, \ldots n$. If more than half are infinite depth ($FMAX$), assign $D(x, y) = FMAX$.

2. If not, then compute the weighted median med, mean m and standard deviation σ of the K depths that are finite and are of high confidence. In other words, if $g(\cdot)$ is the indexing function representing the subset of the original n depth maps, then

$$\begin{aligned}
\text{med}(x, y) &= \text{median}\left\{\hat{D}_{g(i)}(x, y),\ i = 1, 2, \ldots K\right\} && (2.23) \\
m(x, y) &= \frac{\sum_{i=1}^{K} w(i) D_{g(i)}(x, y)}{\sum_{i=1}^{K} w(i)} && (2.24) \\
\sigma(x, y) &= \left(\frac{\sum_{i=1}^{K} w(i) D_{g(i)}^2(x, y)}{\sum_{i=1}^{K} w(i)} - m(x, y)^2\right)^{\frac{1}{2}} && (2.25)
\end{aligned}$$

where $w(i)$ are weights shown in Table 2.1.

3. Throw out depth outliers outside the range $\mathrm{med}(x,y) \pm \sigma(x,y)$ leaving only L depth maps indexed by $h(\cdot)$. Set $K = L$ and $g(\cdot) = h(\cdot)$.

4. Repeat Steps 2 and 3 until standard deviation $\sigma(x,y)$ is less than some threshold or until the number of considered points K does not decrease.

5. Take weighted average of remaining K points to find depth $D(x,y)$:

$$D(x,y) = \frac{\sum_{i=1}^{K} w(i) D_{g(i)}(x,y)}{\sum_{i=1}^{K} w(i)} \tag{2.26}$$

6. If $K = 2$, then mark $D(x,y)$ as an invalid depth point if the standard deviation $\sigma(x,y)$ of the two points is too high.

   If $K \leq 1$, then mark $D(x,y)$ as an invalid depth point.

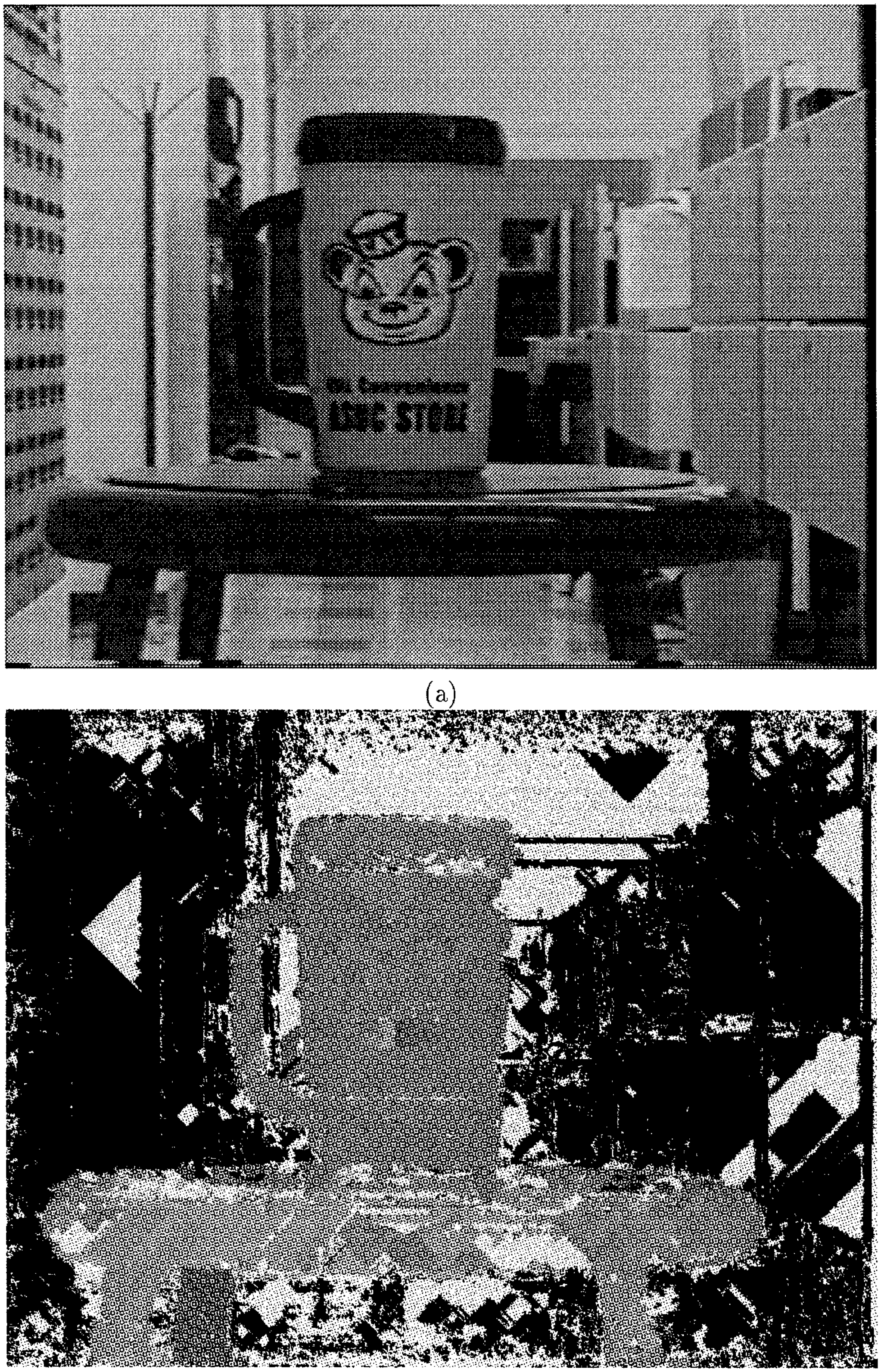
The two special cases in Step 6 ensure that potentially inaccurate depth information will not be used as the final value. For $K = 2$, it is quite possible for the two points $D_a(x,y)$ and $D_b(x,y)$ to be quite different. Clearly, if one point is in the foreground while the other is part of the background, taking their weighted average will result in a depth point somewhere in between. Furthermore, even if both points are both in the foreground, we still may wish to disregard these points because the reconstruction algorithm is sensitive to small differences in depth. To determine whether to assign a depth to point (x,y) given only two depth estimates $D_a(x,y)$ and $D_b(x,y)$, consider the following. Suppose the two contributions $D_a(x,y)$ and $D_b(x,y)$ have equal weight. Then there is a reduced expression for their mean and standard deviation given by

$$m(x,y) = \frac{D_a(x,y) + D_b(x,y)}{2} \tag{2.27}$$

$$\sigma(x,y) = \frac{D_a(x,y) - D_b(x,y)}{2}. \tag{2.28}$$

In other words, the standard deviation is simply half the difference between the data points. Using this fact, an appropriate threshold may be set to differentiate between valid and invalid data sets.

An example of a combined depth map is given in Figure 2.14 (a) and (b), where ABMA was used for solving correspondence. Both the image and depth information, respectively, are shown for a prespecified reference frame. Note that the depth map has been quantized heavily for visualization purposes; lighter colors represent shorter distance to the camera. The regions marked in yellow are low confidence points after combination. For the most part, the depth estimate looks excellent: The mug is clearly recovered as well as most of the stool. Also notice how the walls along either side gradually decay in depth. However problems occur above the mug and along the cabinets on the right. Furthermore, parts of the stool tend to drop out. Many of these artifacts can be attributed to problems with constant intensity.



(a)

(b)

Figure 2.14: *Example of a reference frame intensity-depth pair: Frame #37 of the "Mug2" sequence.*

2.4 Cubic B-Spline Approximation

The depth map after the combination stage is fairly accurate in many regions. There are however a considerable number of low confidence regions. While no depth information is preferred over incorrect data, we would like some reasonable estimates for depth in the areas of low confidence. Generally, an interpolation scheme is used to "fill in" these regions and to make the map much denser while not sacrificing too much accuracy. For this purpose, cubic B-splines are employed to carry out this task.

Cubic B-splines are a class of piecewise continuous polynomials which serve to fit curves or surfaces. They exhibit several desirable properties including locality and piecewise continuity [4, 19]. For real world scenes, it is not unreasonable to assume local continuity of the surfaces. Furthermore, if the surface is treated as a tensor product, i.e. the product of 1-D functions, then the data may be processed first along one direction and then along the other which helps to simplify computations. Hence, we need only discuss techniques for the one-dimensional case; the two-dimensional case is a straightforward extension.

Suppose we would like to approximate a given number of vertices V_i by cubic B-splines. Let $\bar{u}$ be a global parametrization of the curve and let $\bar{u}_i$ be an array of knots over which the spline is defined. Then each curve segment $Q_i(\bar{u})$ may be represented as a linear combination of vertices by certain basis functions, i.e.

$$Q_i(\bar{u}) \;=\; \sum_{r=-3}^{0} V_{i+r} B_{i+r}(\bar{u}). \tag{2.29}$$

These basis functions are generated using a recurrence relation [10] given by

$$B_{i,r}(\bar{u}) \;=\; \frac{\bar{u}-\bar{u}_i}{\bar{u}_{i+r-1}-\bar{u}_i} B_{i,r-1}(\bar{u}) + \frac{\bar{u}_{i+r}-\bar{u}}{\bar{u}_{i+r}-\bar{u}_{i+1}} B_{i+1,r-1}(\bar{u}), \tag{2.30}$$

the basis function of order k at level r with base step

$$B_{i,1}(\bar{u}) \;=\; \begin{cases} 1 & \bar{u}_i \leq \bar{u} < \bar{u}_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.31}$$

Note that these basis functions have finite support; B-spline curve segments depend only on a local set of vertices.

Equation (2.30) applies to the general case where the knot array $\bar{u}_i$ consists of arbitrarily spaced points, i.e. nonuniform knot spacing, Note that in general, the basis functions $B_{i,r}$ differ from one parameter range to the next. If the knots happen to be spaced equally, then the basis functions $B_{i+r} = b_r$ turn out to be the same for every segment:

$$b_{-3} \;=\; \frac{1}{6}(1-3u+3u^2-u^3) \tag{2.32}$$

$$b_{-2} \;=\; \frac{1}{6}(4-6u^2+3u^3) \tag{2.33}$$

$$b_{-1} \;=\; \frac{1}{6}(1+3u+3u^2-3u^3) \tag{2.34}$$

$$b_{-0} \;=\; \frac{1}{6}u^3 \tag{2.35}$$

where u is the local parameterization ranging from 0 to 1.

There are a few ways of implementing splines in our application. Each depth estimate could be processed by B-splines before combining to yield a denser map. Combining each of these maps may produce a more accurate depth estimate, however overall the approach is very computationally intensive and requires a large amount of storage. At the other extreme, one could attempt to locate the regions in the reference depth maps which correspond to holes in the final reconstructed image and use the splines accordingly in these regions. However, it is very difficult to locate these regions and we generally would like the reconstruction algorithm to be as fast as possible. We propose to insert the spline processing stage after combining the depth maps and before reconstructing. Here, the intensity and depth[7] information are used with splines to produce denser maps. Both the intensity and depths are processed independently. In each case, the data to be processed are treated as the vertices V_i as well as the knots corresponding to the grid point (u, v). Notice there is a knot line in the u direction for every column v and a knot line in the v direction for every row u. Applying Equation (2.30) to every knot line fills in the low confidence depth regions and produces the desired 2-D surface. Using a denser map helps to improve the reconstruction and increase the quality of the final images.

Following this technique, one obtains vertices for every point in the grid array. A much denser map may be produced by subdividing the grid array into many more points, thereby producing additional estimates for these new points. Note for a given subdividing factor, we need only compute these so-called extended maps once. In this way the reconstruction process can be much faster. However, as the subdividing factor increases, the amount of information used to represent the scene also increases significantly, thus slowing down the reconstruction algorithm considerably. Also, by experimentation, it turns out that increasing the subdividing factor over one generates only marginally better results. For this reason, we shall fix the subdividing factor to one and hence need only to process the sparse depth maps, not the intensity maps.

To fill in a combined depth map, every row is processed first, followed by every column. The vertices V_i come from the high confidence inverse depth points and the knots $\bar{u}_i$ are the location of the points in the grid array. In the presence of low confidence regions, the spacing of the knots, i.e. the distance between pairs of confident vertices, may be nonuniform.

An example is shown in Figure 2.15. The shaded areas are the regions of low confidence. If we attempt to assign knots and vertices as described above, i.e. at the intersections of each of the dotted lines, then we encounter problems in specifying surface patches—this is clearly one of the main problems with the rectangular topology. The knot lines of v are in fact functions of the u value. One alternative is the following: for every row, create a knot line as before, where nonuniformity occurs if the line intersects a low confidence region. Next we find the vertices corresponding to the "missing" knots by generating the splines passing through the vertices and determining the value on the curve corresponding to the knots. Then with respect to the other direction, we perform the same type of 1-D processing.

[7] Technically, we use the *disparity* instead of depth. The reason is that the depth maps consist of many spikes where disparity equals zero and considering the disparity results in better behaved surfaces.

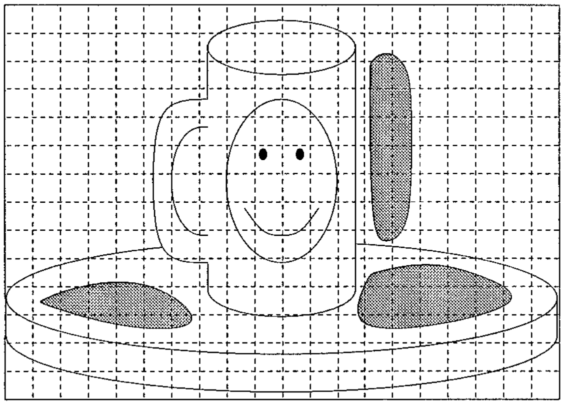


Figure 2.15: *Example of an image with confidence regions.*

As an example, suppose we are given a 4 × 7 array as shown in Figure 2.16 where again the shaded regions are low confidence regions. Assume that there are data points (vertices) corresponding to each knot. Then, we examine each row and compute the spline along each row. For instance, the second row corresponds to a spline with vertices $\{V_{01}, V_{31}, V_{41}, V_{61}\}$ and knots $\{C_{01}, C_{31}, C_{41}, C_{61}\}$ plus additional knots and zero vertices on either end. We use this curve to determine the vertices corresponding to the knots at C_{11}, C_{21}, and C_{51}. In this way, we have the appropriate vertices to interpolate uniformly in the other direction.

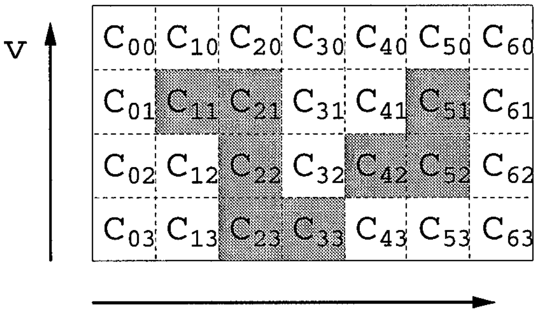

v
C00 C10 C20 C30 C40 C50 C60
C01 C11 C21 C31 C41 C51 C61
C02 C12 C22 C32 C42 C52 C62
C03 C13 C23 C33 C43 C53 C63
u

Figure 2.16: *Example of processing combined depth map with splines.*

For every point in the depth map, there is a preferred direction of processing, either along the row or the column. It is possible that approximating every row first, finding the missing vertices, and then processing every column will result in strange artifacts in one direction. An additional step is introduced to overcome this problem. For every low confidence point, we consider the nearest vertices along the horizontal direction and vertical direction. If the variance of the vertices in the horizontal direction is smaller than that in the vertical direction, the point is processed along the horizontal direction first; otherwise, the point will be approximated in the vertical direction. The purpose is to ensure that regions of similar depths are processed first and that depth discontinuities are not interpolated over incorrectly.

Once the depth map for each reference frame has undergone spline approximation, we are left with $2\frac{1}{2}$-D surface estimates at different locations around the scene. It is possible to further process these depth maps to generate a full 3-D model of the scene by volumetric intersection. However, we believe that registering the maps together is a nontrivial task and that accuracy would be degraded.

The final step in the representation process is to estimate the relative camera motion between reference frames. Estimating camera motion under perspective projection is a nonlinear problem; possible least squares solutions include [45, 39, 30]. For our purposes, we have used the motion estimation scheme proposed by Szeliski [39]. Once the relative motion between all reference frames is known, a geometric relationship may be constructed among the different reference frames. This enables us to better select which reference frames to use in the reconstruction stage.

In the end, the representation of the object consists of the intensity-depth pair at each reference location along with the relative motion among reference frames. Once these data have been derived, they may be stored in a database for later reconstruction. The representation may be compacted even more by compressing both the intensity and depth maps using traditional compression schemes as described in [21, 32].

Chapter 3

Reconstruction of Intermediate Views

Once we have generated the representation for a particular 3-D object, we may choose to reconstruct the view of the object at some specified viewpoint. Assume that the center of one reference frame coincides with the origin of the coordinate system and that the desired viewpoint is known with respect to this origin. The reconstruction algorithm consists of the following: First the appropriate reference frame(s) are chosen. Initial estimates of the desired view are constructed by applying motion parameters to each reference frame. Finally, the estimates are combined into a single image, interpolating when necessary. The entire reconstruction process is diagrammed in Figure 3.1.

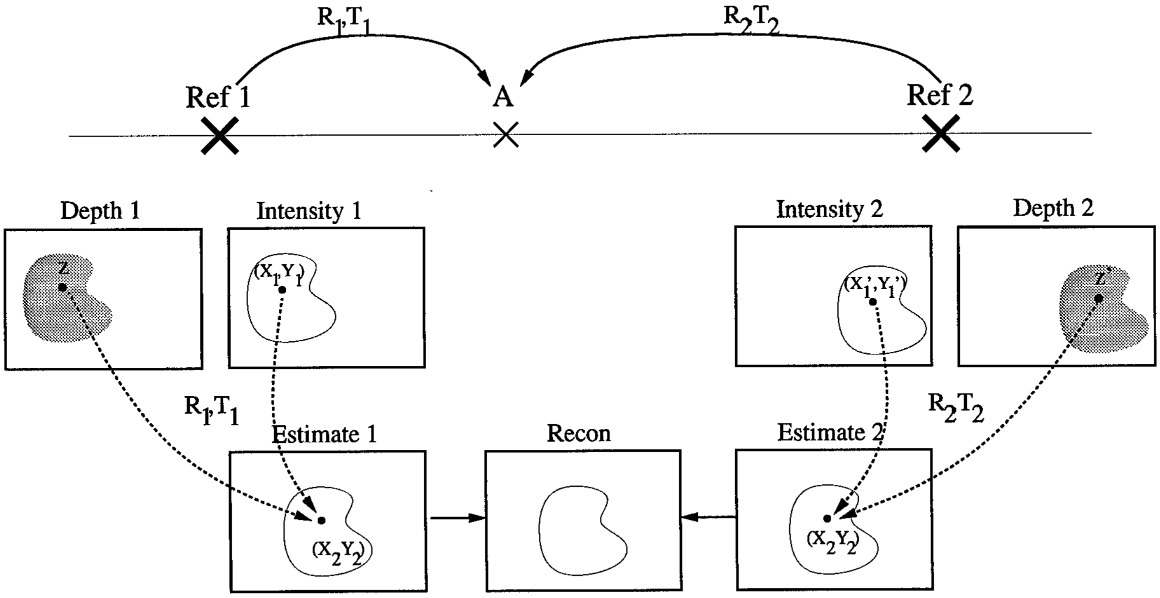


Figure 3.1: *A diagram of the complete reconstruction process.*

3.1 Selection of Appropriate Reference Frame(s)

Given the relative position and orientation of the desired view, it should be a straightforward task to determine which reference frames to use. One way of deciding is to include those frames with the smallest motion in norm relative to the view. It is also possible to consider all reference frames but weigh each according to its reliability for the desired view.

Another consideration is the number of reference frames. If the specified view is very close to one of the reference frames, then we may choose to use only that single frame. However, in most cases, at least two reference frames are needed to properly reconstruct the desired view. Aside from reducing the effect of noisy data, using multiple reference frames is important in recovering occluded regions. If the view lies on a linear trajectory between two reference locations, then two reference frames may be sufficient. If more views are introduced, the reconstructed image should be improved. However, this slows down the process considerably. Our current implementation consists of using two reference frames only.

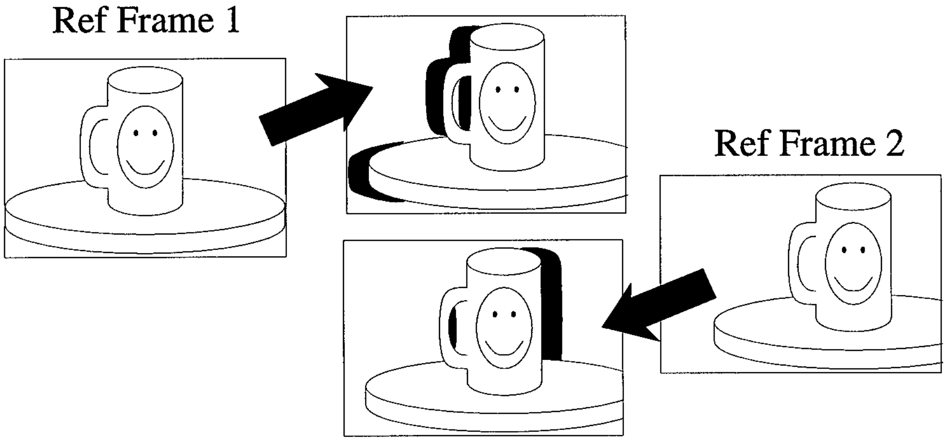


Figure 3.2: *Example of using two reference frames to handle occlusions.*

Figure 3.2 shows an example where two reference frames lie along the same horizontal trajectory, similar to the set up in Figure 3.1. If only one reference frame, say Reference Frame 1, is used to generate the desired view, then there are regions of points where the algorithm lacks information; these are precisely the locations in the scene which were previously occluded and become deoccluded, drawn in black in Figure 3.2. The view from a second reference frame also contains occluded regions. However, the regions do not generally intersect. Therefore, using both reference frames and combining their view estimates results in an improved reconstruction.

3.2 Generation of View Estimates

For every reference frame, we generate an estimate of the desired view by applying the appropriate rigid transformation to the intensities according to the corresponding depth.

Transformation of Reference Frame

The notion of applying motion parameters to a frame has been addressed in conventional computer vision and robotics literature [17, 47, 31]. Let $p = (X, Y, Z)$ be a point in the scene and suppose its projection onto the image plane is $q = (u_1, v_1) = (fX/Z, fY/Z)$. Suppose the frame of reference undergoes a rigid transformation (R, T) given by

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \qquad T = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}, \tag{3.1}$$

where both rotation R and translation T are in terms of the world coordinates. Then the point $p' = (X', Y', Z')$ in terms of the new frame of reference is given by

$$p' = Rp + T \tag{3.2}$$

or, if we consider homogeneous coordinates, the affine equation becomes the matrix equation

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} \tag{3.3}$$

The projection of p' onto the image plane is simply $q' = (u_2, v_2) = (fX'/Z', fY'/Z')$ or, after some algebra,

$$u_2 = \frac{(r_1u_1 + r_2v_1 + r_3)Z + \Delta x}{(r_7u_1 + r_8v_1 + r_9)Z + \Delta z} \tag{3.4}$$

$$v_2 = \frac{(r_4u_1 + r_5v_1 + r_6)Z + \Delta y}{(r_7u_1 + r_8v_1 + r_9)Z + \Delta z} \tag{3.5}$$

where the focal length f is assumed to be 1.

Reference Frame: Discrete Points vs. Deformable Mesh

It is clear that if (R, T) is the relative motion from a given reference frame to the desired view, Equations (3.4) and (3.5) determine the new coordinates of the points in the reference frame. In this framework, the points (x, y) in the reference frame are regarded as discrete independent points since neither the image nor the depth map is a continuous surface. The view estimate generated by this method does not exploit the connectedness of the scene and it may exhibit inconsistencies in the ordering of foreground and background points. Figure 3.3 provides some insight to this problem. Consider a simple one-dimensional sequence, say a portion of one row in the reference frame. Both the depth and the intensities are shown; for simplicity, the intensity values increase from one to seven. Points b and c are considered part of the foreground since their depth is very small. The remaining points form the background of this signal. Suppose the desired view is translated to the right. Then the points are mapped according to Equations (3.4) and (3.5) to the configuration shown on the right. Since points d through f have a large depth, the transformed points move very little if at all. In contrast, points b and c are displaced the most since they have the smallest depth. The transformed sequence indicates that some erroneous background point (point f) appears within the object.

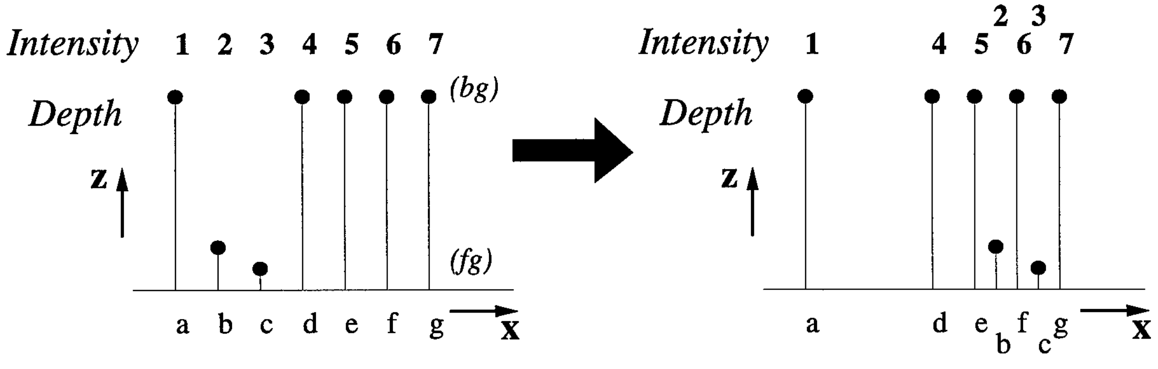


Figure 3.3: *1-D example of reference frame regarded as discrete set of points. Motion is to the right.*

A better approach is to consider the points of the reference frame arrays as vertices of a deformable wire mesh. Neighboring points in the reference frame are viewed as connected to one another. A view estimate is generated by applying the appropriate transformation to the collection of points and examining not only the new coordinates of every point, but also the ordering in the mesh. In the event that points with large depth, i.e. part of the background, appear in between points with small depth, i.e. part of the foreground, the ordering may be used to discard such points. In this manner, the ordering of points may be better preserved and inconsistencies in the transformed data are not as prevalent. If we apply this mesh approach to the previous example, the result is shown in Figure 3.4. Here, the ordering of the points contributes to improve the view estimates. By examining the sequence of the transformed points, it is straightforward to determine occluded point f in the background and subsequently discard it. Since the reference frames are 2-D arrays, the deformation must occur in both directions. A simplification may be made where one direction is considered first, followed by the other one.

The problem of occluding surfaces has been examined in traditional computer graphics [33, 15] and is referred to as the hidden surface removal problem. With a representation of the surfaces of the scene in memory, it is possible to determine which surface precedes the rest according to different tests, e.g. using the surface with smaller depth in z-buffering technique [15]. In our case, we have not segmented the scene into explicit surfaces, so the hidden surface removal problem is not as easy to solve. Using a connected mesh approach helps connect surfaces in the scene and resolves some of the occlusion ambiguities.

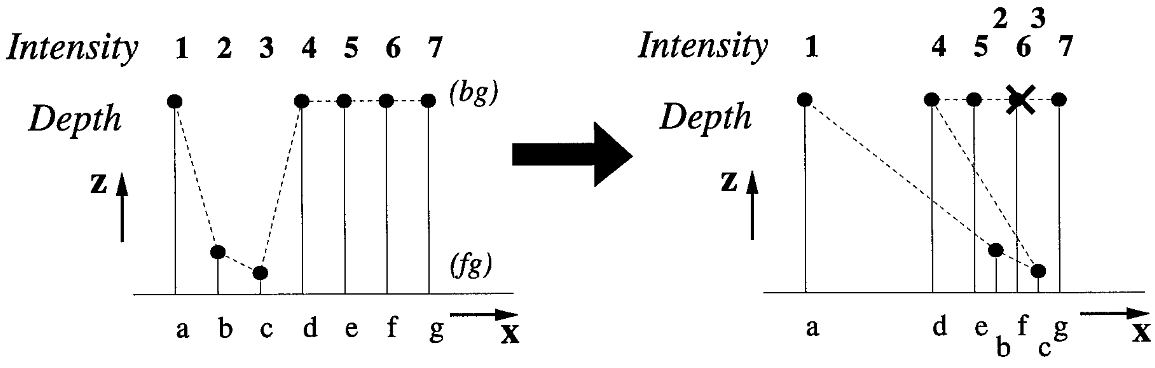


Figure 3.4: *1-D example of reference frame regarded as part of a deformable mesh. Motion is to the right.*

Refining the Deformed Mesh

To make the view estimates more robust, interpolation between consecutive points according to the mesh may be included. In the previous example it is clear that there is no information to replace the original location of points *b* and *c*. This "hole" may be filled by linearly interpolating between successive points in the array, e.g. points are added in between points *a* and *d* and also between *b* and *c*. Such an interpolation scheme serves to give local estimates of intensity information. These interpolated points are marked as low confidence since there is no information about the deoccluded region and hence there is no guarantee the interpolated values are correct.

3.3 Combination of Reconstructed Data

Once we compute the estimates of the desired view with respect to each of the chosen reference frames, we must decide how to combine this data to generate the appropriate reconstruction. Furthermore, it is quite possible after the previous step that the estimates do not coincide with the sampling grid, so we must also ensure that the data are interpolated to the grid points.

Interpolation by Distances

One possibility is to use the following technique for interpolation [7, 48]. Suppose there are N reference frames for reconstruction and suppose that there exists at least one data point near the given pixel (i,j). Then, the intensity value of the pixel on the sampling grid is given by

$$I(i,j) = \sum_{n=1}^{N} w_n \left(\frac{\sum_{k_n \in A_{ij}} d_{k_n} I_{k_n}}{\sum_{k_n \in A_{ij}} d_{k_n}} \right) \tag{3.6}$$

where A_{ij} is a $2L \times 2L$ region centered around pixel (i, j), L is the spacing on the sampling grid, d_{k_n} is the distance of the $k_n th$ point in frame n with intensity I_{k_n} from the (i, j) pixel, and w_n is the weight for the data from reference frame n. Generally, these weights depend on the location with respect to the reference frames.

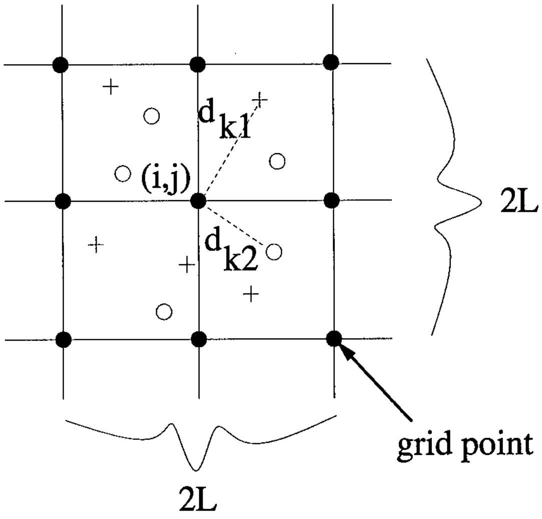


Figure 3.5: *Example of interpolating data from two reference frames for pixel (i,j).*

We consider an example of data from two (N=2) reference frames in Figure 3.5. The desired viewpoint is assumed to lie along a linear trajectory between a left and right reference frame. The translation parameter of the intermediate view is Δx, while that of the right frame is d. Notice both parameters are with respect to the left frame. If we use the weighting function $w_0 = (1-\Delta x/d)$ for the left frame, $w_1 = \Delta x/d$ for the right frame, then the previous equation reduces to

$$I(i,j) \;=\; (1-\frac{\Delta x}{d})\left(\frac{\sum_{l\in A_{ij}} d_l I_l}{\sum_{l\in A_{ij}} d_l}\right) + \frac{\Delta x}{d}\left(\frac{\sum_{r\in A_{ij}} d_r I_r}{\sum_{r\in A_{ij}} d_r}\right) \qquad (3.7)$$

the interpolation formula for bilinear reconstruction.

It was shown in [7] that the interpolation scheme works reasonably well for reference frames regarded as discrete point arrays. This approach relies on the accuracy of the depth estimates to perform reliably since the distance to the nearest grid point is implicitly determined by the depth values. Moreover, points in a given $2L \times 2L$ region may be inconsistent in both depth and intensity with one another, and no provisions are made to analyze them.

Interpolation with Outlier Removal

Based on the above discussion, we may propose a second improved algorithm. Suppose there are again N reference frames for reconstruction. To find the intensity for pixel (i, j), we examine points in a smaller $L \times L$ region $A_{i,j}$ centered around pixel (i, j). Outliers in the depth domain are thrown out until the intensities of the points are consistent using an approach similar to the approach described in Section 2.3. Once these points have been reduced, the intensity $I(i,j)$ is simply the weighted average of the remaining points.

More specifically, let us suppose there are n_k data points $I_k(i,j)$ and $D_k(i,j)$, intensity and depth respectively, from reference frame k in a $L \times L$ region centered around pixel (i,j). All points in this region from reference frame i which were generated by the interpolation step after mesh deformation are removed as long as there exist some valid points from other reference frames. To find $I(i,j)$, we follow the following steps:

Algorithm (Reconstruction):

1. Examine $D_k(i,j)$ for $k = 1, 2, \ldots N$.
2. If more than half are infinite depth ($FMAX$), then
   (a) Compute mean $m(i,j)$ and variance $\sigma^2(i,j)$ of $I_{g(k)}(i,j)$, where $g(k)$ is the appropriate indexing function.
   (b) Throw out intensity outliers outside the range of $m(i,j) \pm \sigma(i,j)$.
   (c) Repeat Steps 2a through 2b until σ is less than some threshold or until the number of points does not decrease.
   (d) Take the average of the intensities of the remaining points.
3. Otherwise
   (a) Compute mean $mi(i,j)$ and variance $\sigma i^2(i,j)$ of $I_{g(k)}(i,j)$, where $g(k)$ is the appropriate indexing function.
   (b) Compute mean $md(i,j)$ and variance $\sigma d^2(i,j)$ of $D_{g(k)}(i,j)$.
   (c) Throw out depth outliers outside the range of $md(i,j) \pm \sigma d(i,j)$.
   (d) Repeat Steps 3a and 3c until σi is less than some threshold or until the number of points does not decrease.
   (e) If σi is still larger than some threshold, then remove outliers in the intensity domain; follow approach taken in Steps 2a and 2b.
   (f) Take the average of the intensities of the remaining points.
4. If σi is too large, mark the intensity $I(i,j)$ as an invalid point.

The goal of the algorithm is to determine which data are consistent. Depth outliers are removed until the intensity values of the points in a given region are similar.

One refinement is to place more weight on points with smaller depths when interpolating. When an object moves in a frame, we would like the pixels of the foreground to have more weight than those from the background occupying the same region. One possible solution is to multiply the closer points by a large factor and to weigh less the points further away.

The result is that pixels that are displaced more will dominate over those that tend to be stationary. The regions near depth discontinuities of the reconstructed image are improved with this technique. We note that such a refinement places a strong dependence on the accuracy of the depths. This approach is akin to the z-buffering technique discussed before [15].

It is possible that there exists no points in a given $L \times L$ region, creating "holes." This condition arises because of uncovered regions in the scene, i.e. deoccluded regions, and because of sparse depth information. Generally, introducing more reference frames or using spline-processed depth maps with a larger subdividing factor helps to reduce the size of these holes. For the remaining holes, some sort of interpolation is necessary to fill-in the holes. An approach is to grow the area A_{ij} out to a $mL \times mL$ region, where m is the smallest value for which a point falls within the area A_{ij}, i.e. the region is no longer a hole. Once we find such an area, we then use either interpolation scheme to find the intensity value at the grid point (i, j).

A comparison of the two interpolation schemes shows that the interpolation-with-outlier-removal algorithm performs much better than the interpolation-by-distances algorithm. It is however considerably slower since statistical analysis is performed for every point. For the eventual goal of real-time reconstruction, one may consider simplifying the outlier removal process to speed up the overall algorithm. Nevertheless, we believe the second interpolation scheme produces much more reliable results; we shall consider only this algorithm for the next chapter.

Chapter 4

Results

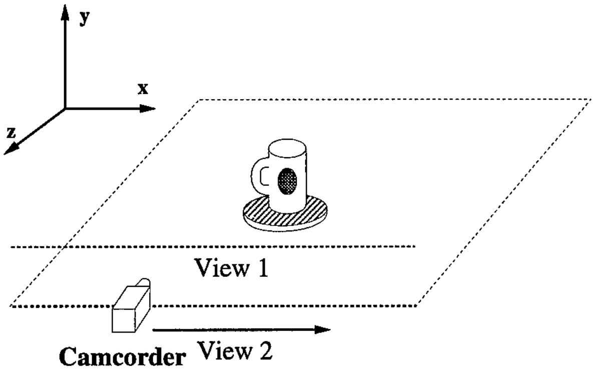


Figure 4.1: *Experimental set up used to generate results.*

We shall now examine some results using the techniques described above. The outside-in scene employed is one hallway in our office consisting of several objects at different depths. The object of interest is a mug placed atop a stool. A CCD camcorder is moved on a rolling tripod without rotation along two trajectories about two feet in length in front of the mug three feet away at two different elevations to generate a 100-frame sequence per trajectory, similar to the set up drawn in Figure 4.1. The sequences, named "Mug1" and "Mug2," respectively, are subsampled to produce shorter-length sequences and then processed by the representation algorithms. Each frame is 640 × 480 pixels large and consists of intensity only. When creating the sequences, we attempted to make the motion strictly translational along the x axis. However the camcorder was moved by hand without a track, we cannot ensure this to be true. Moreover, no special lighting was used to film the scene; specularities of the stool and the lid of the mug are very apparent in the images.

For the results, two frames, both on the same trajectory (Section 4.1) or one on either trajectory (Section 4.2), are chosen as reference frames. To generate these depth maps, ten neighboring frames from the same trajectory and one vertically-related frame from the other trajectory are matched to yield local depth estimates. The effect of using fewer neighboring frames is also described. These depth estimates are normalized and then combined as described in Chapter 2.

The low confidence regions are then filled in using a cubic B-spline approximation technique to produce a dense depth map. The subdivision factor is one in both directions. A larger subdivision factor would result in a much finer depth map but did not seem to improve the reconstructed image overall. Negative intensity and depth values are possible after approximation due to the sharp variation in disparities and are disregarded during reconstruction.

4.1 Reconstructing Horizontal View

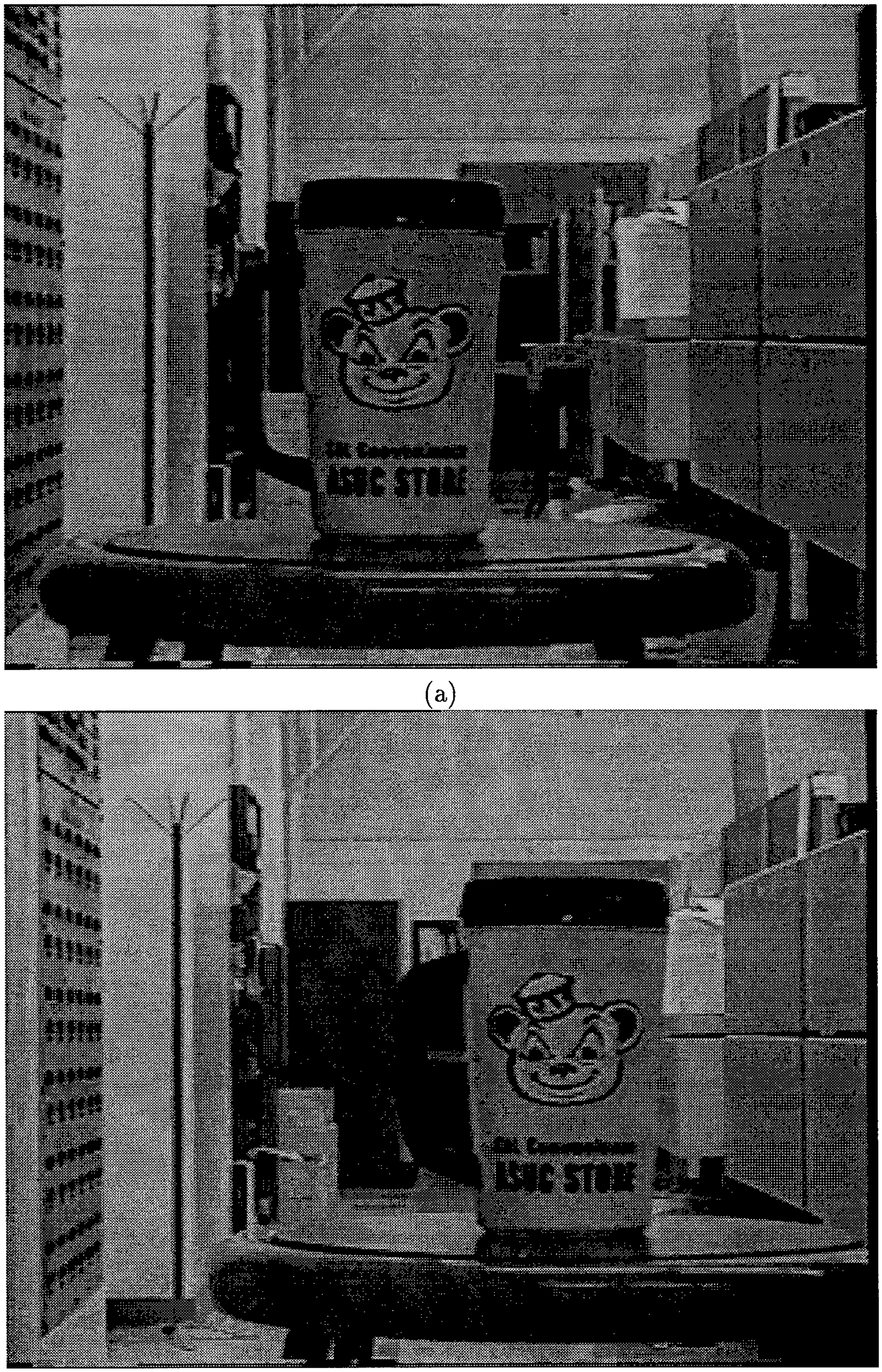
For the first set of results, the three matching algorithms described in Chapter 2 are used to reconstruct the view roughly halfway between two reference frames along the same horizontal trajectory. Frames #35 and #65 from the first trajectory "Mugl" sequence are selected as the reference frames; they are shown in Figure 4.2 (a) and (b). This desired view is perhaps the one most prone to errors due to the large occluded regions. Note that there is roughly a maximum of 120 pixel disparity between the two reference frames. The closest image to the desired view is Frame #49 of "Mugl," shown in Figure 4.3. Since the exact camera motion is not known, the image is only an approximation and serves only to aid in comparison.

Block Matching Algorithm

The reference depth maps for #35 and #65 using BMA with 9×9 blocks in the representation algorithm are shown in Figure 4.4 (a) and (b). The general shape of the mug and stool have been recovered in both depth maps, however it is evident that the matching algorithm fails in many regions. The front of the stool, the drawers to the right, and the wall in the background have spurious depths due to low texture (*CONST*) in those regions. Problems on the top of the stool occur due to the reflections from the light. Most of the horizontal edges have incorrect depth estimates because of the aperture ambiguity (*AP*).

The bilaterally reconstructed view is shown in Figure 4.5 (a). Despite the problems with the depth maps, the image quality of the reconstruction is surprisingly good. The main reason is that the reconstruction algorithm discards inconsistent depth estimates and their corresponding intensities after mesh deformation. These regions become empty and are then replaced by more accurate intensities to fill up the image. Notice that occluded regions have been recovered for the most part, although there are some problems to the right of the mug and near the mug handle. There are also some artifacts along the various horizontal lines in the image e.g. the reflections in the front of the stool, the drawer to the right of the mug, and the door in the background.

The error between the two images is shown in Figure 4.8 (b). The output has been offset so that gray level 128 corresponds to zero error; brighter levels indicate positive error and darker levels reflect negative error. It is clear that most of the errors occur along the intensity discontinuities and especially near the occlusion regions, as mentioned above. The



(a)

(b)

Figure 4.2: *Reference Frames (a) #35 and (b) #65 of "Mug1" (intensity).*

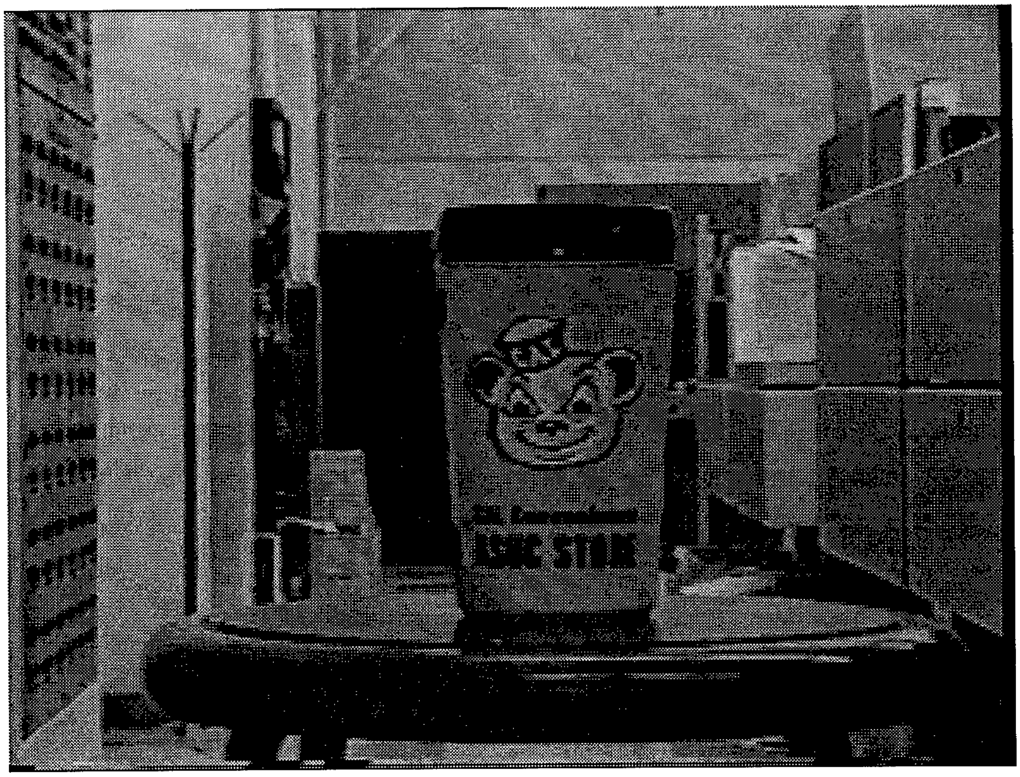
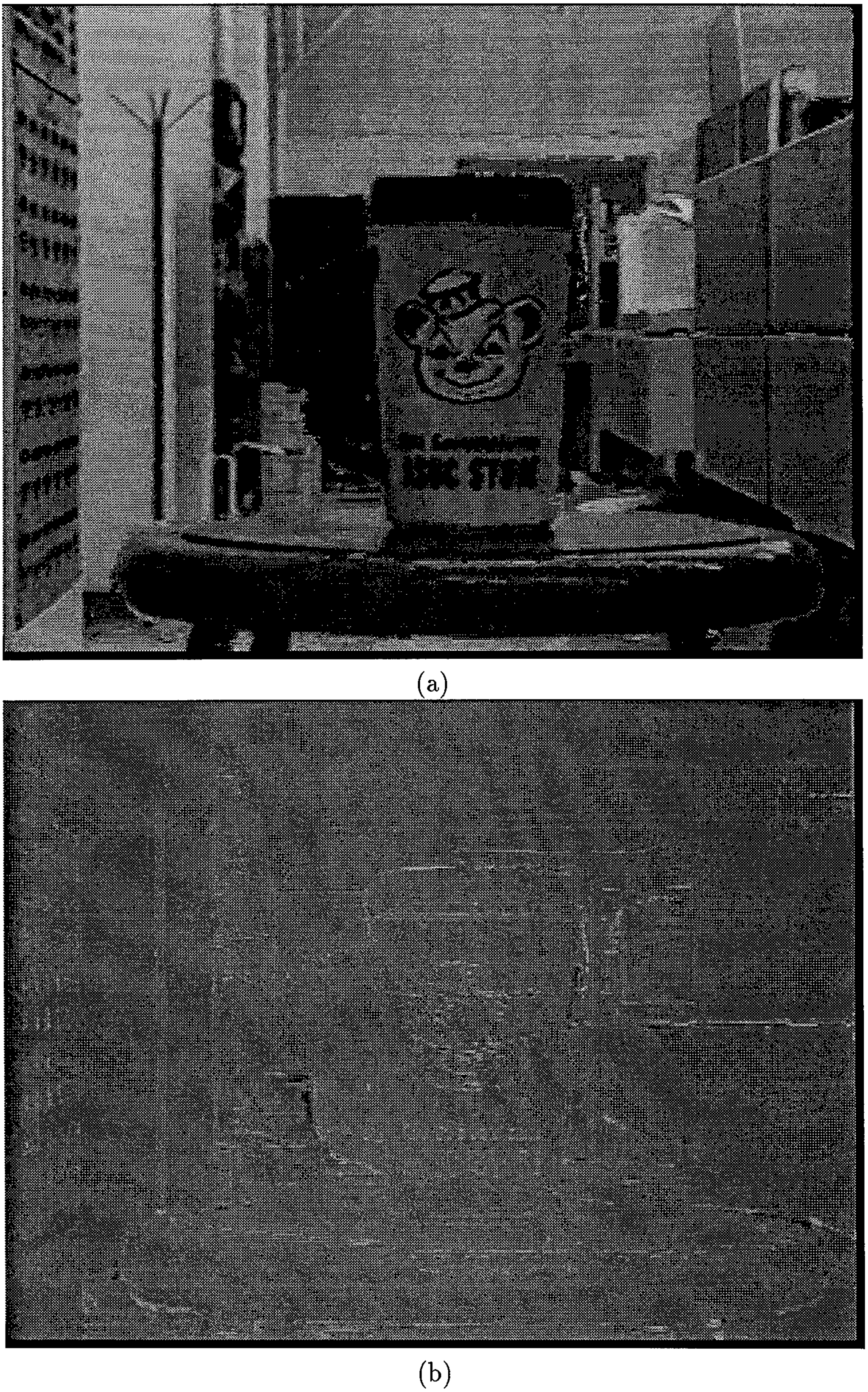


Figure 4.3: *Closest image to desired view from "Mug1" sequence: Frame #49.*



(a)

(b)

Figure 4.5: *Results from BMA: (a) Reconstructed view along horizontal trajectory; (b) Error obtained by subtracting reconstructed view and Frame #49.*

mean squared error (MSE) of the reconstructed view with the closest image from "Mugl" is 42.30, implying that on average the error is 6.5 levels of gray per point.

Adaptive Block Matching Algorithm

To improve the previous results, we consider using a different matching algorithm, ABMA. Figure 4.6 (a) and (b) show the depth maps obtained by using ABMA. The regions of low confidence depths are marked in yellow. They may be filled in using the spline approximation described in Section 2.4; Figure 4.7 (a) and (b) have the results after spline processing. It is clear that the result is much a better estimate of depth than the one obtained by BMA: the mug and stool are estimated much better and do not contain as many spurious depths. There is a gradual change in depth as expected for a hallway scene. Artifacts are still prevalent in the top left portion of the stool; this is primarily due to the specularities of the surface. Also, there are problems in recovering the handle of the mug accurately mainly because intensity-based matching schemes perform poorly for regions in the background that can be seen through regions in the foreground.

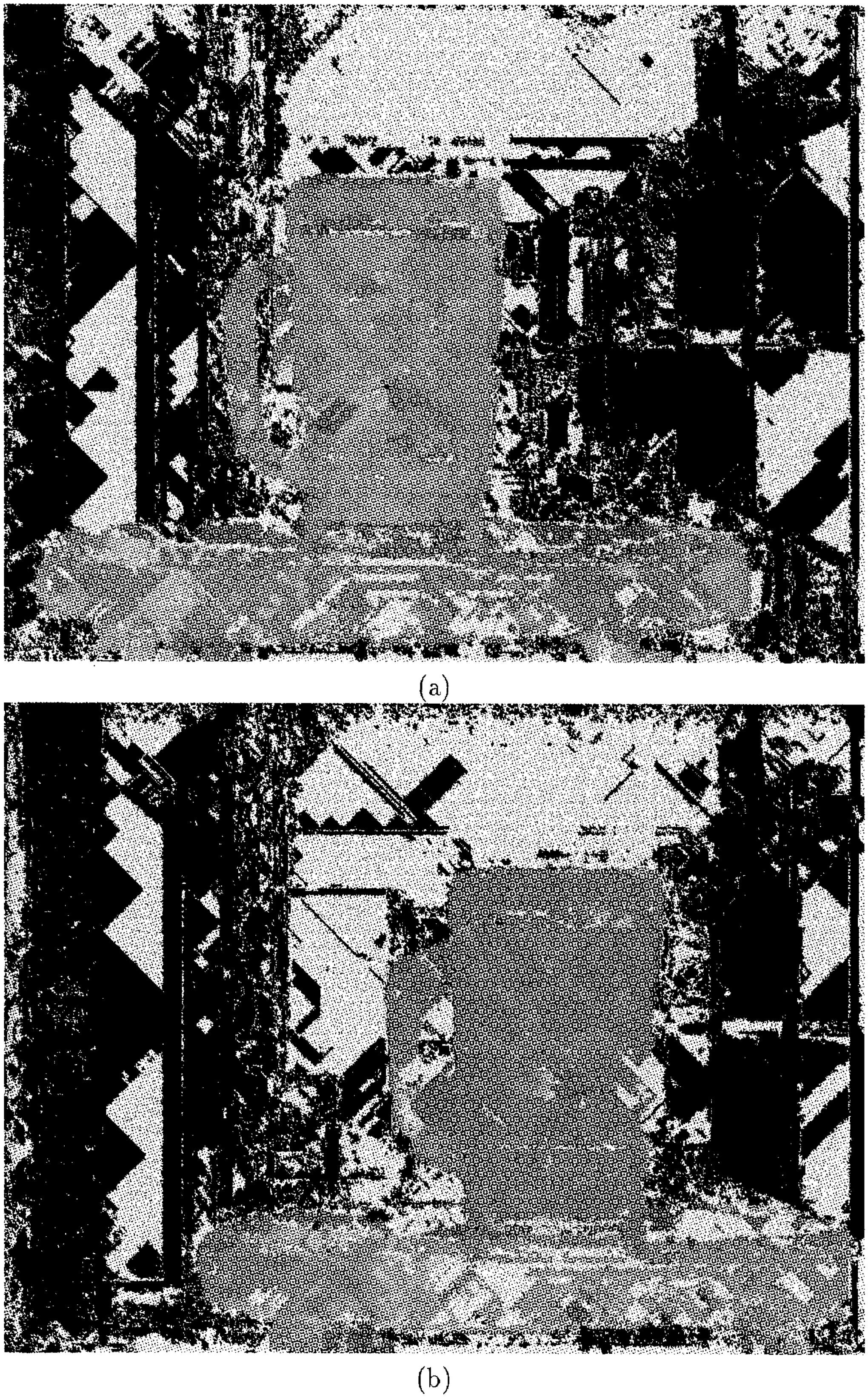
Figure 4.8 (a) shows the reconstructed view. Again, the image quality is good for the most part. The horizontal edges, e.g. top of the door, top of the mug, specularities in front of the stool, and the drawers, have been improved. ABMA takes care of problems in occluded regions: There are fewer errors to the right of the mug and near the mug handle.

The error between the two images is found in Figure 4.8 (b). As with BMA, most of the errors occur around intensity discontinuities and near occluded regions. It is interesting to note that the reflections from the stool top and front were reconstructed quite well with ABMA and result in close to zero error. The MSE for the view compared with Frame #49 is 57.2. This is roughly an error of 7.6 levels per point, one pixel/point higher than the error associated with BMA.

Fast Adaptive Block Matching Algorithm

The results may be further improved and generated faster using the FABMA. In Figure 4.9 (a) and (b), the reference depth maps for Frames #35 and #65, respectively, are shown. The regions of low confidence are marked in yellow in the figures. Using spline approximation to attempt to improve the depth estimates leads to the depth maps shown in Figure 4.10 (a) and (b). The depths have been estimated quite well for the mug and the stool. The main problem in both maps occurs inside the handle of the mug. Since the matching algorithm cannot adequately estimate the depth for this region, the region is marked as low confidence. This implies a stronger dependence on the spline approximation stage, which incorrectly tends to fill in the entire region.

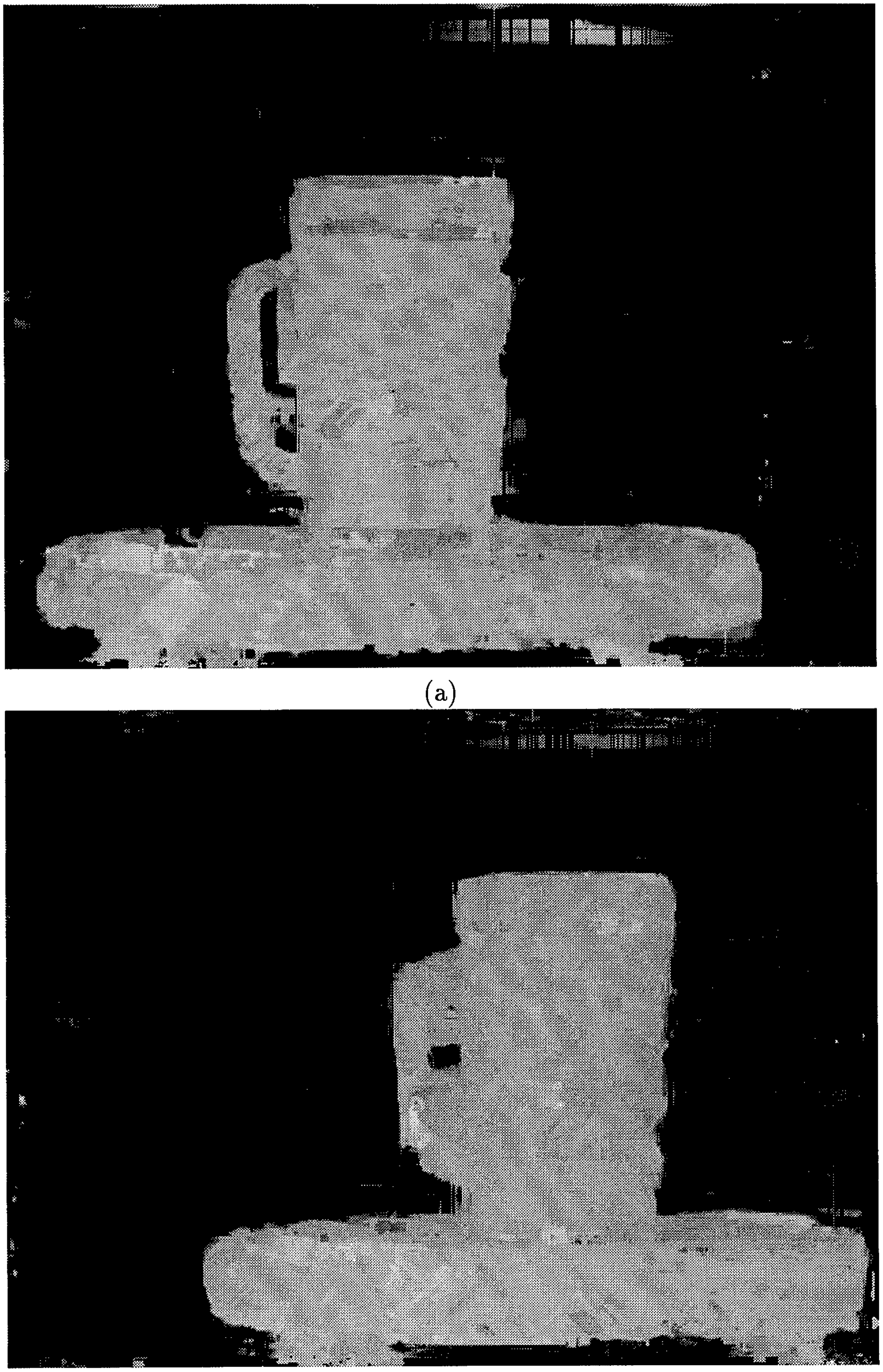
FABMA generates the view shown in Figure 4.11 (a). The reconstructed image appears to be very similar to that of Frame #49 as desired. However, as with the previous two matching approaches, most of the errors occur near occluded regions. These errors are evident in the difference image in Figure 4.11 (b). The largest errors occur in the bottom half of the handle



(a)

(b)

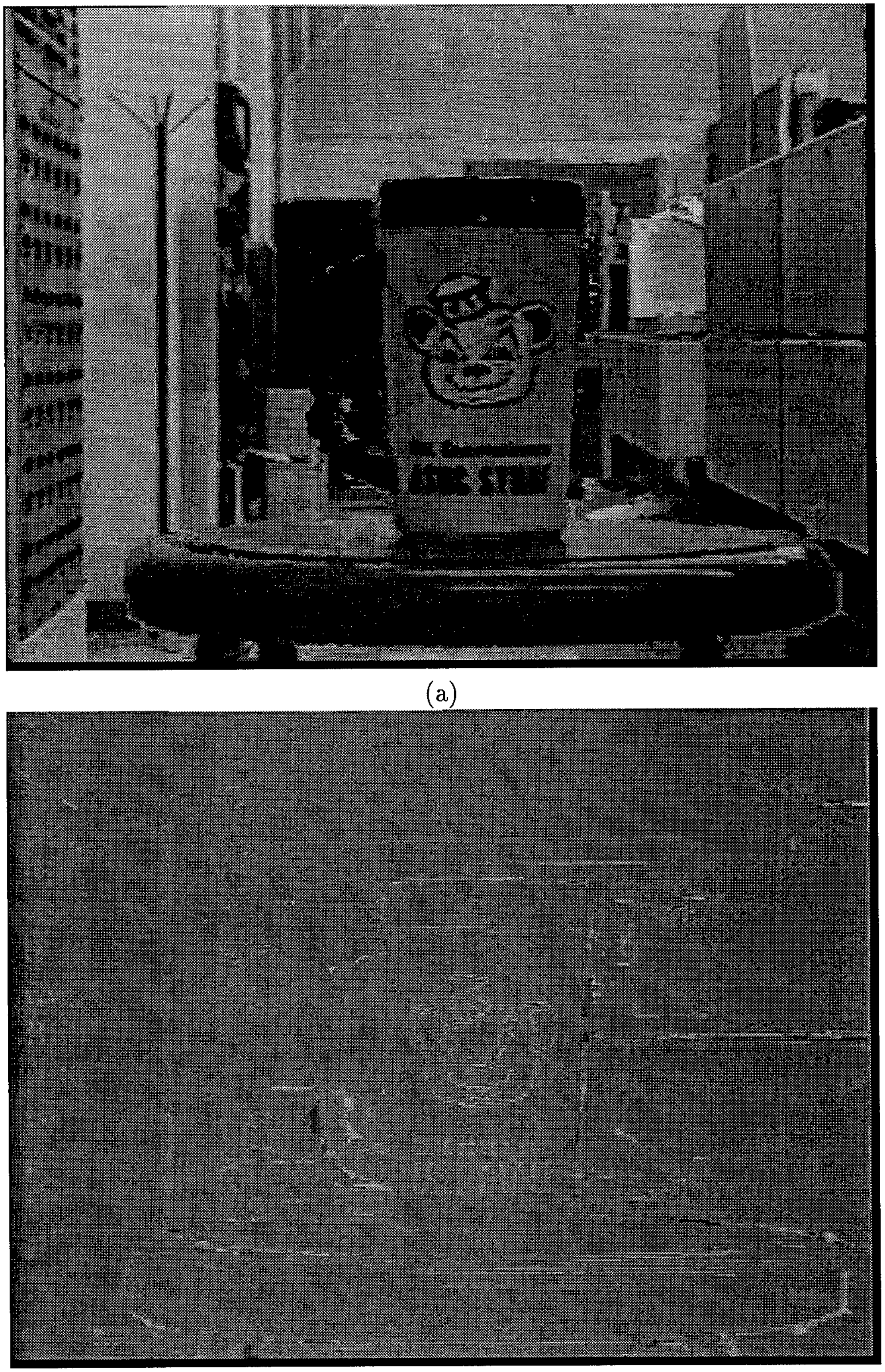
Figure 4.6: *Reference Frames (a) #35 and (b) #65 (depth) using ABMA with confidence measures.*



(a)

(b)

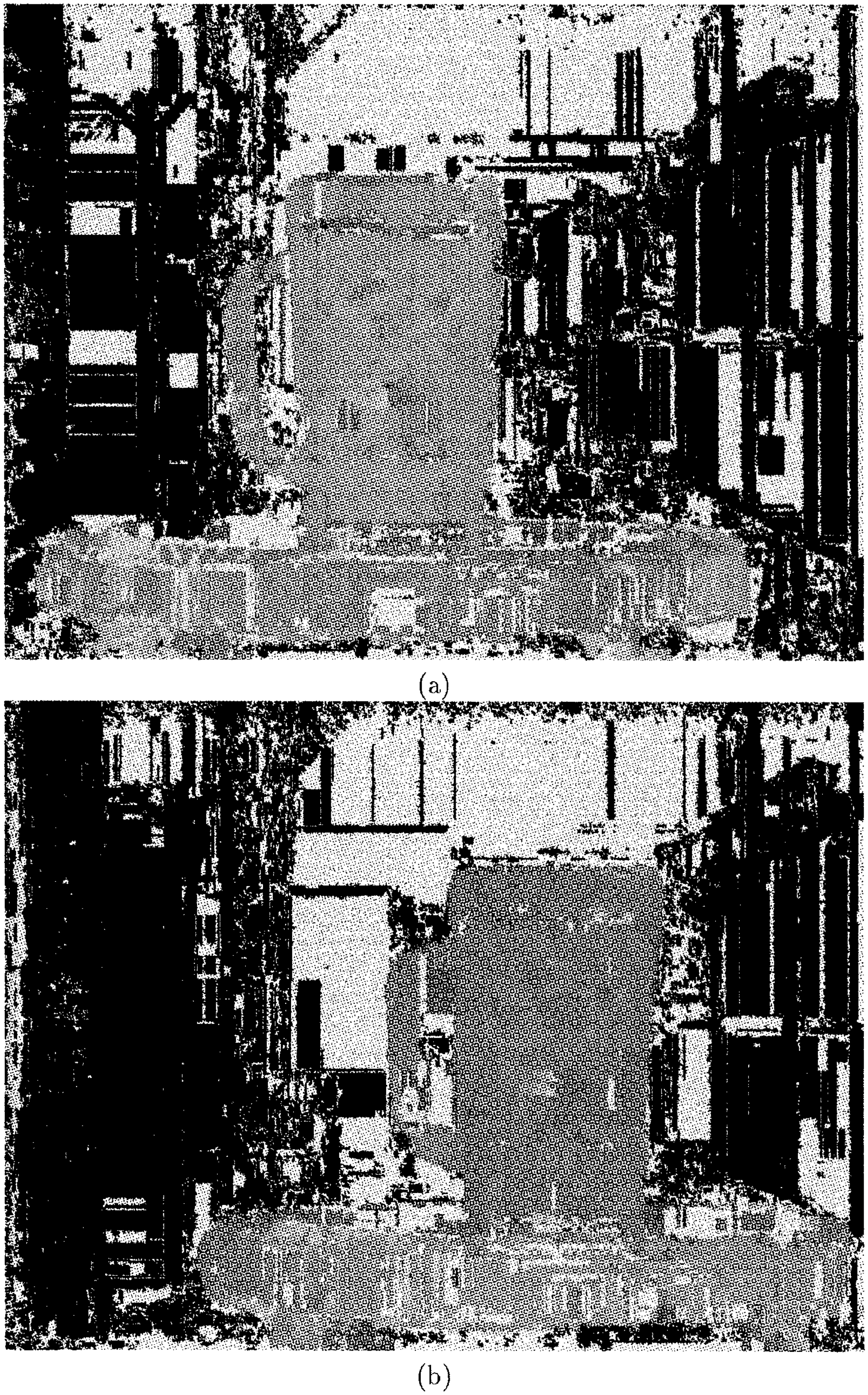
Figure 4.7: *Reference Frames (a) #35 and (b) #65 (depth) using ABMA, filled in by splines.*



(a)

(b)

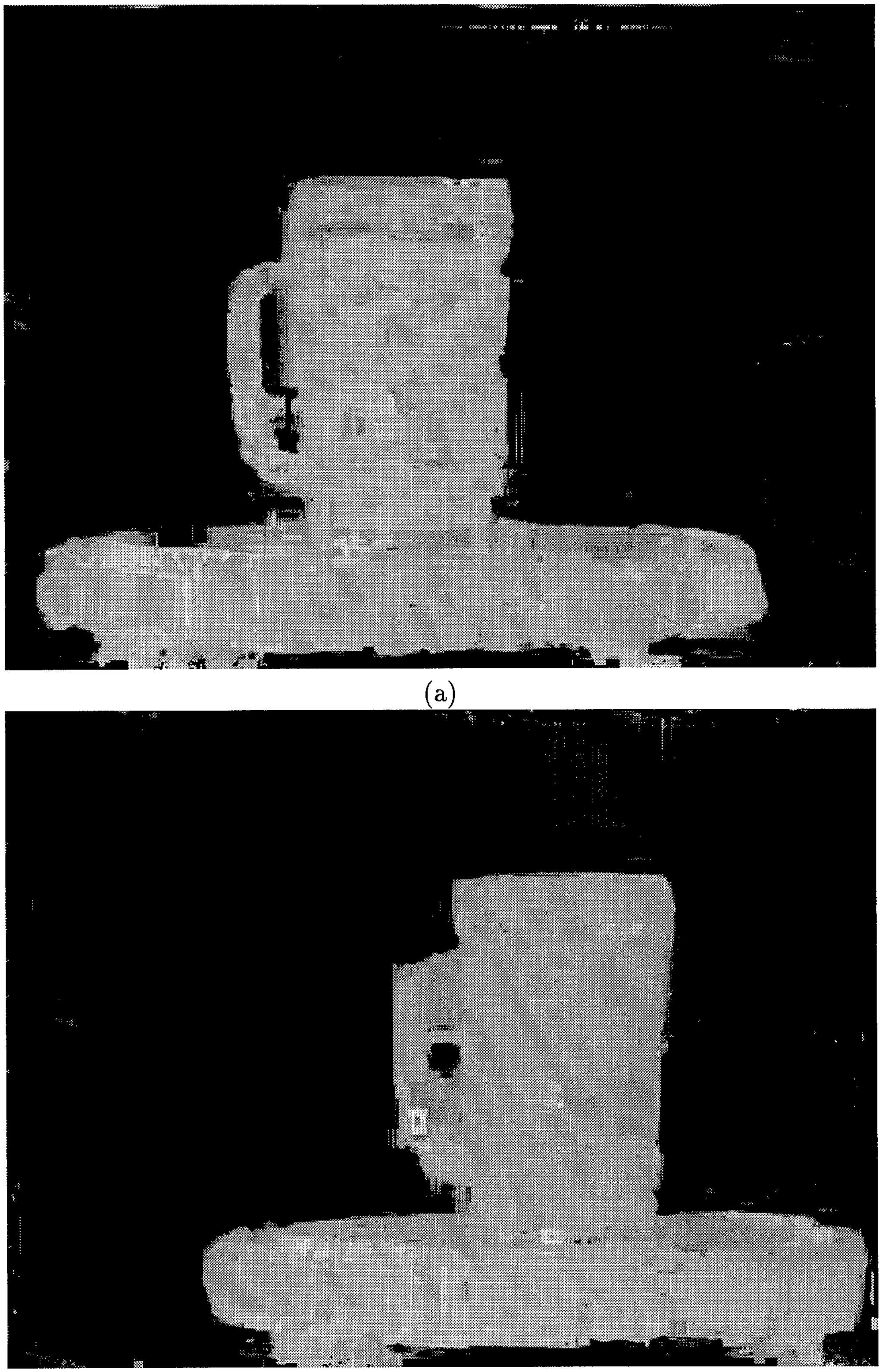
Figure 4.8: *Results from ABMA: (a) Reconstructed view along horizontal trajectory; (b) Error obtained by subtracting reconstructed view and Frame #49.*



(a)

(b)

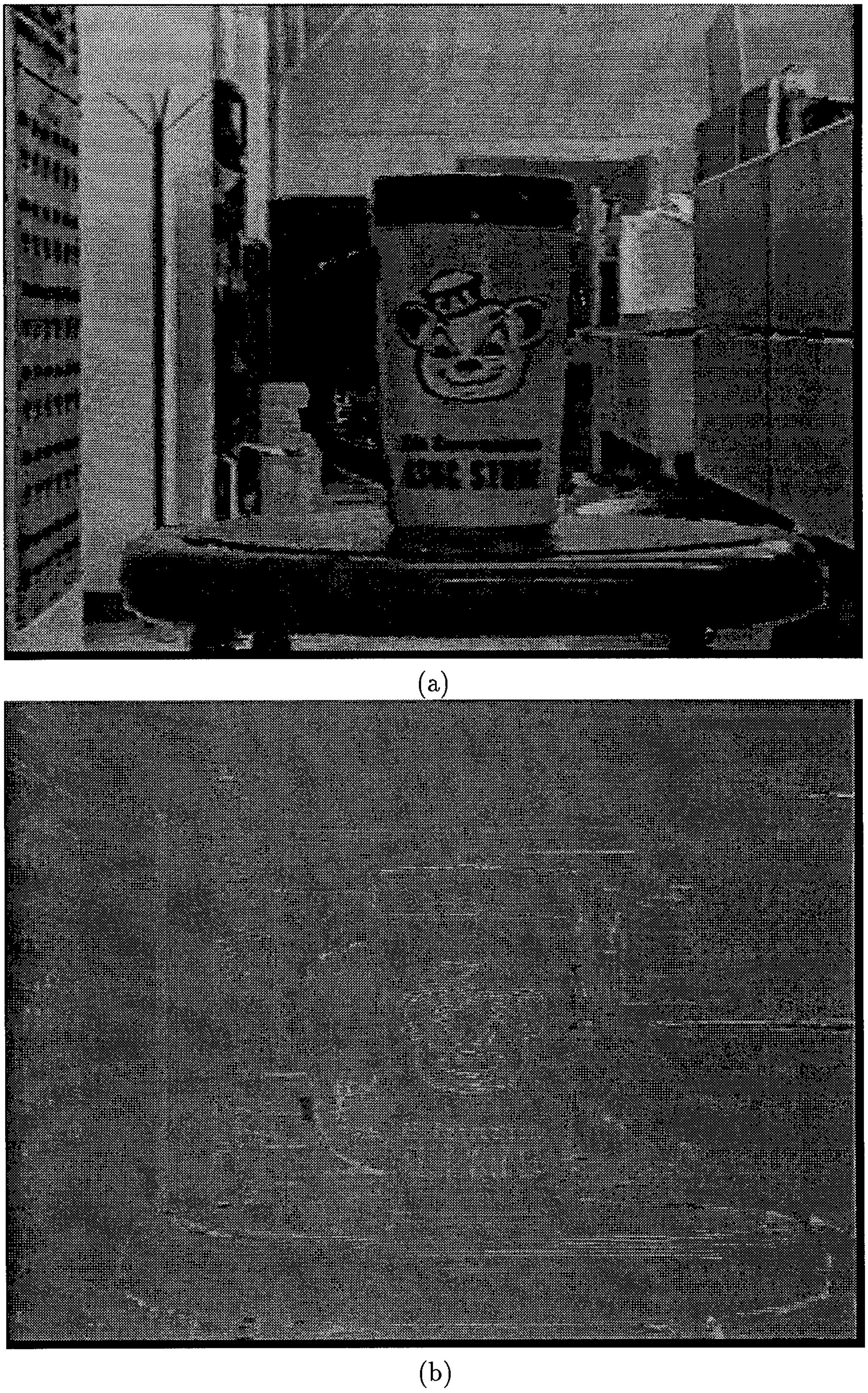
Figure 4.9: *Reference Frames (a) #35 and (b) #65 (depth) using FABMA with confidence measures.*



(a)

(b)

Figure 4.10: *Reference Frames (a) #35 and (b) #65 (depth) using FABMA, filled in by splines.*



(a)

(b)

Figure 4.11: *Results from FABMA: (a) Reconstructed view along horizontal trajectory; (b) Error obtained by subtracting reconstructed view and Frame #49.*

and some points to the right of the mug. Also, artifacts occur near horizontal edges in the image due to problems with depth due to *AP*. The MSE for FABMA is 50.4 or about 7.1 pixel error per point, slightly lower than ABMA but still higher than BMA.

Analysis of Results

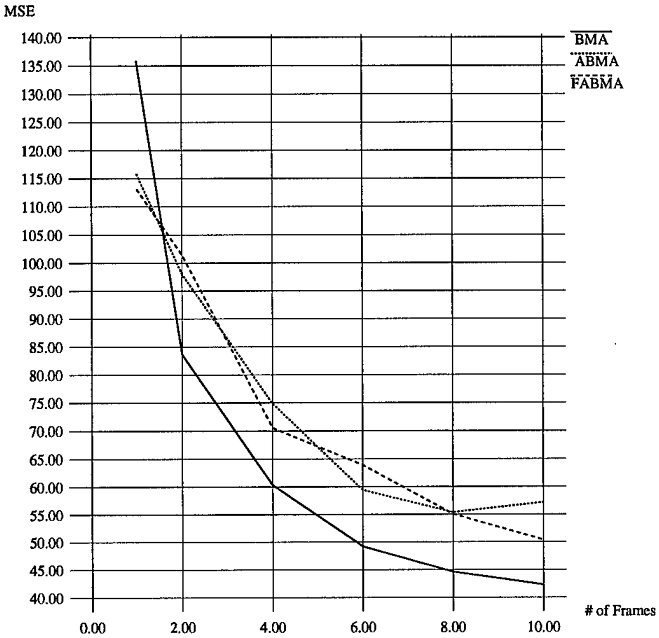


Figure 4.12: *Graph of mean squared error for the three matching algorithms as a function of number of neighboring frames used.*

For the previous results, ten neighboring frames, along with one frame translated vertically, were used to estimate the depth for a given reference frame. An interesting experiment is to determine the trend of the error of the reconstructed image as a function of neighboring frames. We perform such an experiment for one, two, four, six, eight, and ten neighbors on the same horizontal trajectory. As seen in Figure 4.12, the mean squared error of BMA tends to about ten pixels/point lower than either ABMA and FABMA. This is a rather unexpected result especially since ABMA and FABMA produce more accurate depth maps. One explanation is that the regions where BMA yields incorrect depths, namely *CONST* and *AP*, do not affect the overall quality of the reconstructed image. Accurate depth values are not necessary for *CONST* and *AP* points since the view transformation will confine these points to remain in the same general neighborhood, thereby obscuring the error. Also, there are a large number of low confidence regions in both ABMA and FABMA which does not provide the spline processing stage enough vertices with which to interpolate.

From the graph and results shown in Table 4.1, we can also conclude that as more neighboring frames are included, the better is the resulting reconstructed view. While there is clearly some sort of limit, one must consider the tradeoff between the quality of the desired view and the amount of processing required for the increased number of neighboring frames.[1]

# of Frames	BMA	ABMA	FABMA
1	135.84	115.80	113.11
2	83.63	98.03	101.48
4	60.37	74.86	70.54
6	49.23	59.46	63.94
8	44.62	55.40	55.08
10	42.30	57.20	50.38

Table 4.1: *MSE of horizontal view for three matching algorithms as a function of number of neighboring frames.*

4.2 Reconstructing Vertical View

In this section, we consider generating a view not originally scanned by the camcorder. Two frames from different elevations, namely Frame #35 of "Mug1" and Frame #37 of "Mug2" in Figure 4.13, are chosen as reference frames. The desired view is roughly the midpoint on the vertical trajectory relating the two views. Unlike in the horizontal case, there is no corresponding frame for this view. Consequently, we cannot provide any quantitative measure of the performance of these algorithms.

Block Matching Algorithm

Using BMA, the depth map corresponding to Frame #37 is shown in Figure 4.14 (a). While the mug and stool are identifiable in the depth map, there are still many incorrect estimates of depth throughout the map. The reconstructed view in Figure 4.14 (b) looks quite degraded. The bottom of the stool appears to be completely disintegrated. Even part of the top of the stool seems to have disappeared completely. Almost all of the errors occur near horizontal edges. The reason is that the depth associated with horizontal edges are inconsistent due to aperture ambiguity (AP). It is interesting to note that this problem is not manifested in the horizontal case; despite horizontal edges having inconsistent depth, the reconstructed edges still look good since points on horizontal lines remain on the line. However, in the vertical case, the error associated with horizontal edges is the greatest since the reconstructed edges are shifted vertically.

[1] The reader is warned not to weigh the MSE results too heavily. There is no ground truth with which the reconstructed images can be compared; the view we have chosen is the best approximation to the desired view. Also, good MSE results do not necessarily reflect good visual results and vice versa.

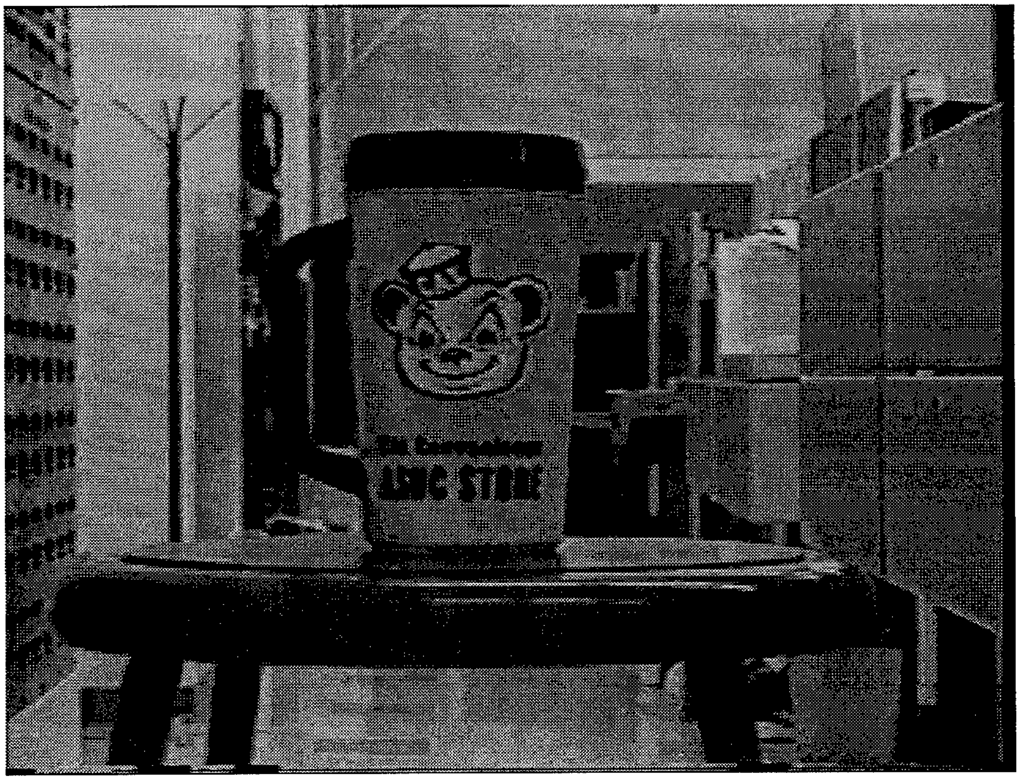


Figure 4.13: *Reference Frame #37 of "Mug2" (intensity).*

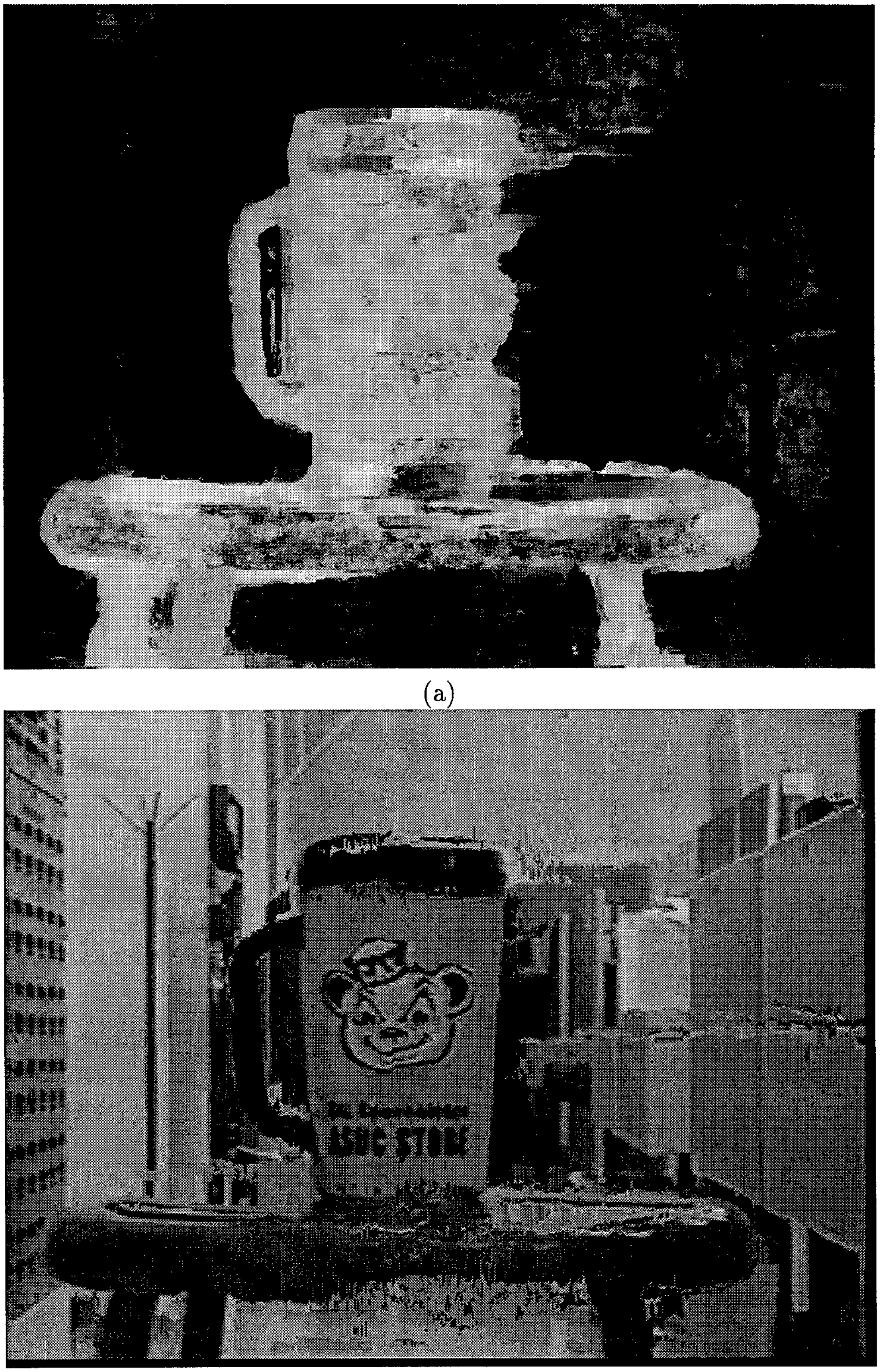
Adaptive Block Matching Algorithm

To improve the results near horizontal AP, we need to introduce more information from the vertical match; matches between vertically translated images result in AP along *vertical* edges and not along horizontal ones. If AP regions are identified during combination, then more weight can be placed on the vertical match information.

Figure 4.15 (a) is the depth map from using ABMA during matching. The estimate is significantly better than that generated by BMA. The mug and stool are again recovered very clearly and there are fewer regions of spurious depths. Even the shape of the handle has been recovered adequately. The reconstructed view is given in Figure 4.15 (b). While the image appears to be an improvement over the one produced by BMA, the overall quality is still not that high. The top of the mug is jagged as are parts of the stool. As described above, the primary reason for this artifact is that the depth along horizontal edges is inconsistent so that after transformation, the reconstructed edges are no longer aligned.

Fast Adaptive Block Matching Algorithm

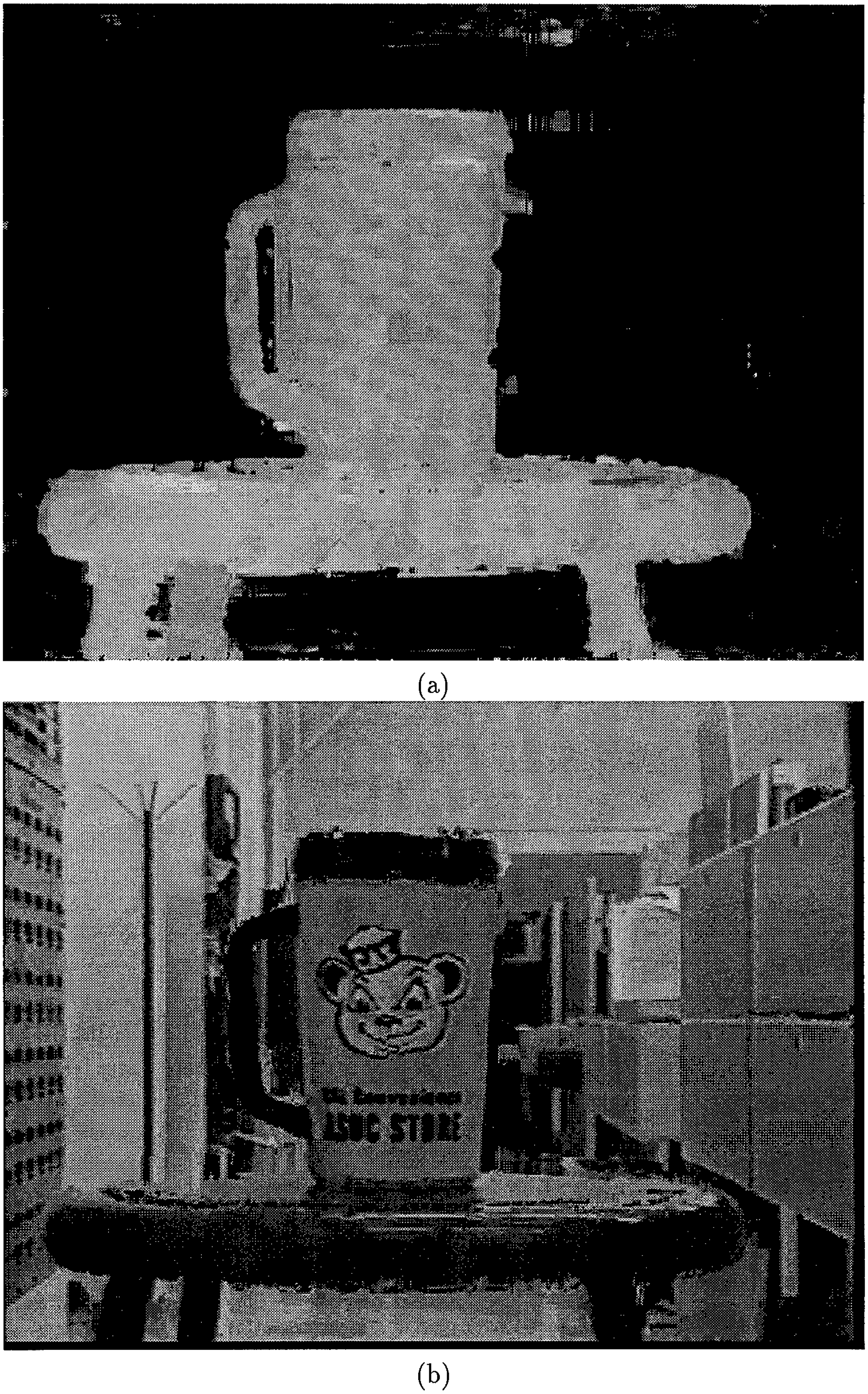
The depth corresponding to the reference Frame #37 is shown in Figure 4.16 (a). For the most part, the depth estimates are reasonable. The reconstructed view generated using this reference frame and Frame #35 is found in Figure 4.16 (b). Similar to the results from



(a)

(b)

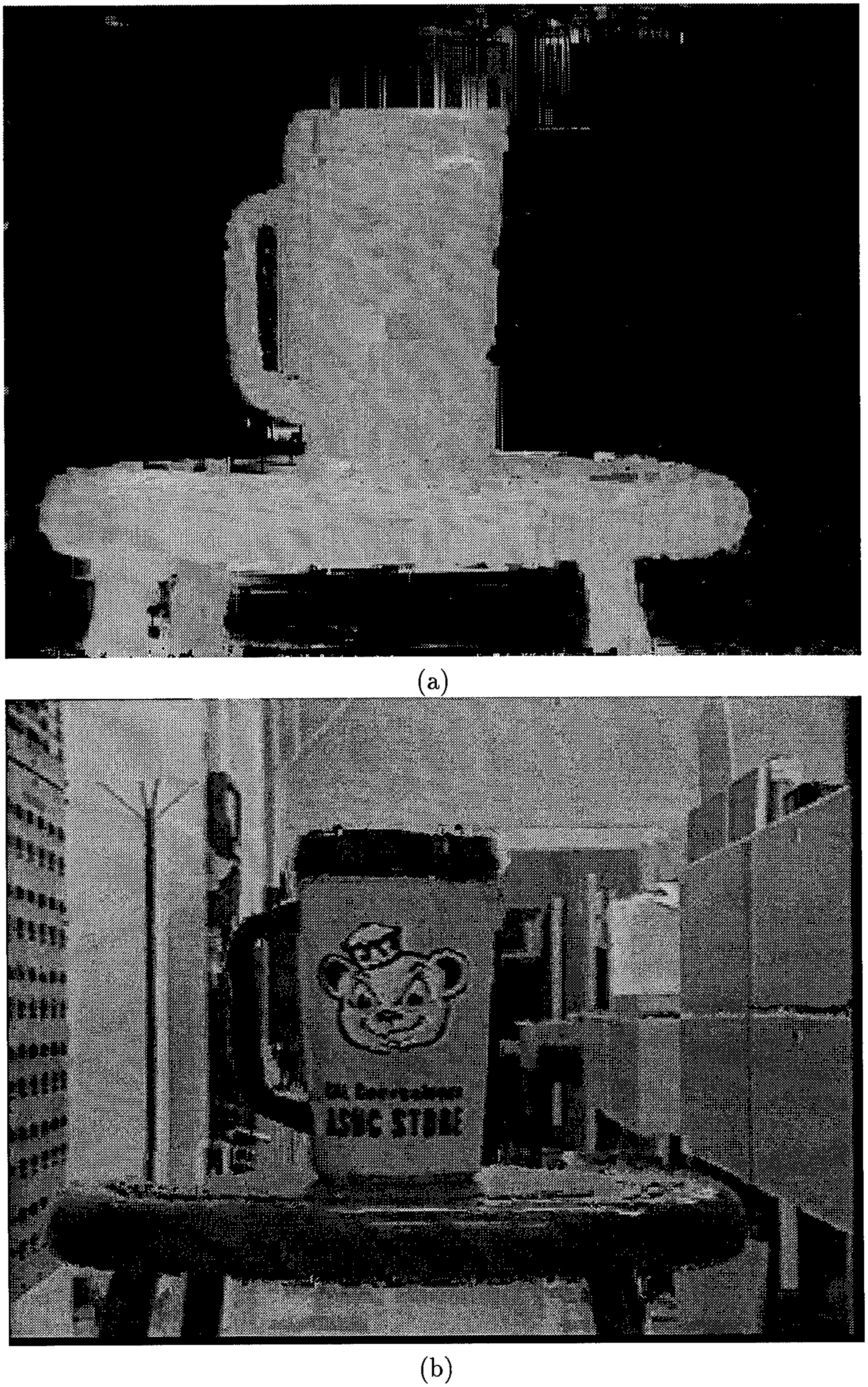
Figure 4.14: *Results from BMA: (a) Reference Frame #37 (depth); (b) Reconstructed view along vertical trajectory.*



(a)

(b)

Figure 4.15: *Results from ABMA: (a) Reference Frame #37 (depth); (b) Reconstructed view along vertical trajectory.*



(a)

(b)

Figure 4.16: *Results from FABMA: (a) Reference Frame #37 (depth); (b) Reconstructed view along vertical trajectory.*

ABMA, many of the horizontal edges appear jagged. The stool is marginally improved, however there are still errors near the reflections.

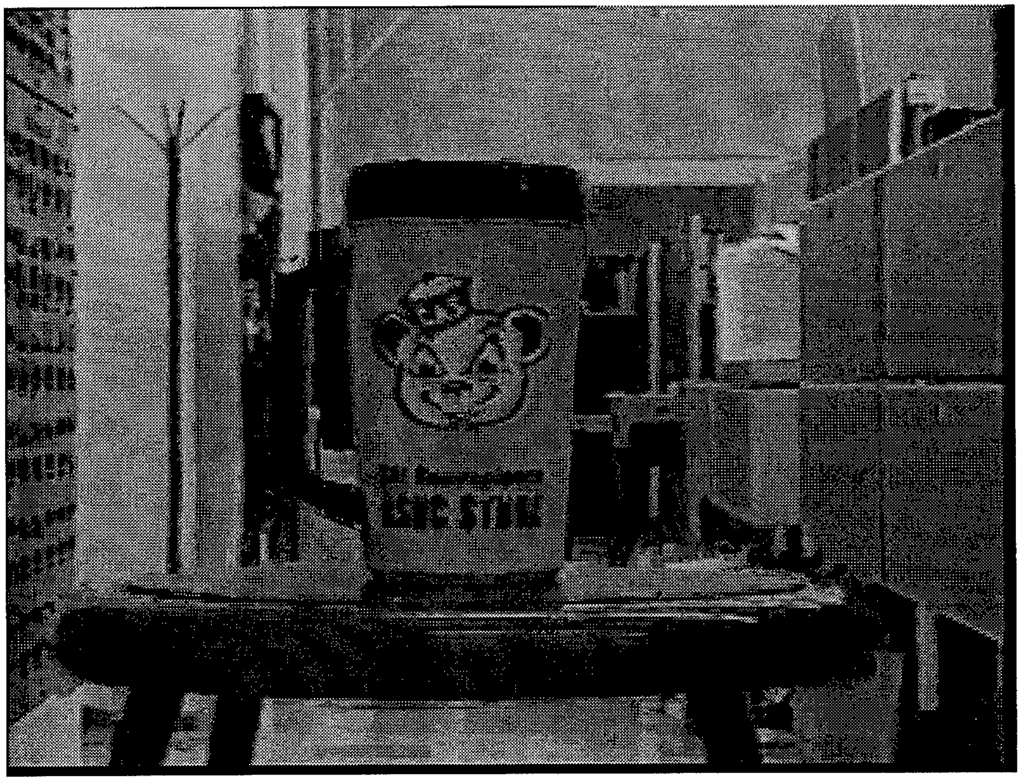


Figure 4.17: *Reconstructed view along vertical trajectory using only vertical match information.*

Analysis of Results

It is clear that the desired view could be improved significantly if the depths of the reference frames were more consistent along the horizontal edges. Since frames from the same horizontal trajectory all exhibit problems with horizontal *AP* points, one should rely more on information from a vertical match. In our current system, only one vertical match is used. If more matches from different trajectories are included, we believe the depth map would be much more consistent and hence the reconstructed image would be improved; we are currently investigating this issue.

An interesting observation is that if only the vertical match information is used and depth estimates from neighboring horizontal frames are discarded, the resulting reconstructed image looks quite good. As shown in Figure 4.17, many of the previous artifacts around horizontal edges are not as apparent. This is because there is no horizontal aperture ambiguity with the vertical matches and the depth information is much more consistent in these regions.

Chapter 5

Conclusion

We have proposed an approach for representing and reconstructing stationary 3-D objects. The reconstructions in the previous chapter seem to indicate that this approach is very worthwhile. For views along a horizontal trajectory, the algorithm produces decent reconstructed images where most of the error is concentrated near the occlusion boundaries. The reconstruction algorithm performs well especially when given very noisy depth information as in the case of BMA. The views along a vertical trajectory were also promising. It is clear that while ABMA and FABMA give slightly lower quality horizontal reconstructions, they lead to a much better estimate of depth than BMA.

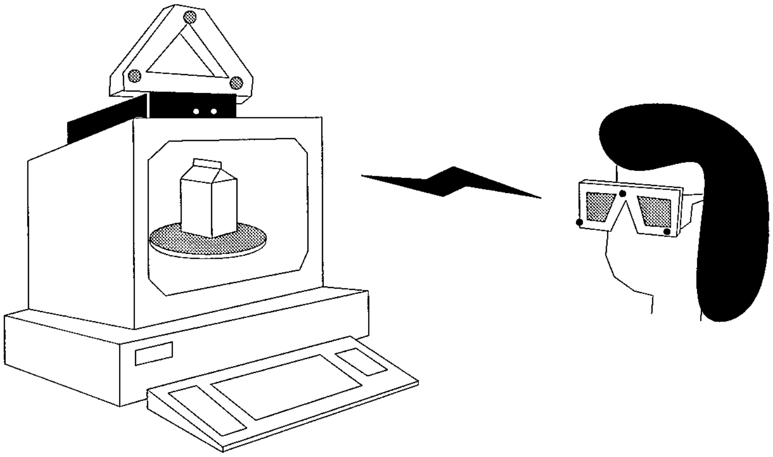


Figure 5.1: *Example of an interactive virtual environment.*

The main bottleneck in the accuracy of the views is the initial depth estimation stage. Replacing the current matching algorithm by a more sophisticated one may improve the accuracy of the reference frame depth maps. The three matching algorithms presented all consider matching at only one scale. While for small motions a single scale should be sufficient [43, 14], a more general multiresolution/multiscale may improve the results by resolving many of the ambiguities in our correspondences. Hierarchical matching methods like [3, 35, 6] and

Kalman-filtering based methods such as [29] might lead to better results. Also, we have assumed a translational camera motion model which allowed us to solve correspondence and estimate depth simultaneously. A more general camera motion model might lead to better results. The camera motion and depth could be iteratively estimated using a linear motion estimation algorithm like [27, 42] or a nonlinear structure-from-motion technique like [39, 46] to improve correspondence results. If such improvements are made, we believe the framework presented may be used to produce a good method for generating arbitrary views.

Future work in this area includes examining the optimum number of reference frames to fully capture an object. A more complete analysis must be performed in order to determine what the scope of a single reference frame is, or conversely, what the optimum set and locations of reference frames to compactly represent a given object is. In addition, a real-time implementation of the reconstruction algorithm would expedite the development of a virtual environment. Using a stereoscopic display and head tracking device, we will be able to simulate such a system by reconstructing an arbitrary view of an object in real time as the user moves his/her head; an example is drawn in Figure 5.1. Another problem is to extend this work for inside-out scenes, i.e. representing a location such as a room rather than an object. In this case, it is unclear what is the proper representation and what are the necessary views to capture it sufficiently. The area of arbitrary view generation and its application to virtual environments seems very fertile and this research serves as a good starting point.

Bibliography

[1] E. Adrizzone, M. A. Palazzo, and F. Sorbello, "3-D scene reconstruction from multiple 2-D views," in *Proceedings of the 5th Int'l Conf. on Image Analysis and Processing*, pp. 394–398, Positano, Italy, 20–22 Sept. 1989.

[2] P. Anandan, "Computing dense displacement fields with confidence measures in scenes containing occlusion," in *Proceedings of the SPIE: Intelligent Robots and Computer Vision*, vol. 521, Cambridge, MA, 5–8 Nov. 1984.

[3] P. Anandan, J. R. Bergen, K. J. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," in *Motion Analysis and Image Sequence Processing* (M. I. Sezan and R. L. Lagendijk, ed.), ch. 1, Kluwer Academic Publishers, 1993.

[4] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Los Altos, CA: Morgan Kaufmann, 1987.

[5] C. Braccini, A. Grattarola, and S. Zappatore, "Volumetric and pictorial reconstruction of 3D objects from correspondences in moving 2D views," in *Recent Issues in Pattern Analysis and Recognition* (V. Cantoni, R. Creutzburg, S. Levialdi, G. Wolf, ed.), Springer-Verlag, 1989.

[6] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Comm.*, vol. COM–31, no. 4, pp. 532–540, Apr. 1983.

[7] N. L. Chang and A. Zakhor, "Intermediate view reconstruction for three-dimensional scenes," in *Proceedings of the Int'l Conf. on Digital Signal Processing*, Nicosia, Cyprus, 14–16 July 1993.

[8] S. E. Chen and L. Williams, "View interpolation for image synthesis," in *Proceedings of SIGGRAPH*, New York, 1–6 Aug. 1993.

[9] C. H. Chien and J. K. Aggarwal, "Identification of 3D objects from multiple silhouettes using quadtrees/octrees," *Computer Vision, Graphics and Image Processing*, vol. 36, no. 2–3, pp. 256–273, Nov.-Dec. 1986.

[10] C. de Boor, "On calculating with B-splines," *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50–62, July 1972.

[11] U. R. Dhond and J. K. Aggarwal, "Structure from stereo—a review," *IEEE Trans. Sys. Man Cyber.*, vol. 19, no. 6, pp. 1489–1509, 1989.

[12] P. Fua, "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine Vision and Applications*, vol. 6, no. 1, pp. 35–49, Winter 1993.

[13] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore: Johns Hopkins University Press, 1989.

[14] M. J. Hannah, *Computer Matching of Areas in Stereo Images.* PhD thesis, Stanford University, July 1974.

[15] D. Hearn and M. P. Baker, *Computer Graphics.* Englewood Cliffs, NJ: Prentice-Hall, 1986.

[16] K. Higuchi, M. Hebert, and K. Ikeuchi, "Building 3-D models from unregistered range images," Tech. Rep. CMU-CS-93-214, Carnegie Mellon University, Nov. 1993.

[17] B. K. P. Horn, *Robot Vision.* Cambridge, MA: MIT Press, 1991.

[18] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, nos. 1-3, pp. 185–203, Aug. 1981.

[19] H. S. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Trans. Acous. Speech Sig. Proc.*, vol. ASSP–26, no. 6, pp. 508–517, Dec. 1978.

[20] M. Ito and A. Ishii, "Three-view stereo analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI–8, no. 4, pp. 524–532, July 1986.

[21] A. K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs: Prentice Hall, 1989.

[22] D. G. Jones and J. Malik, "A computational framework for determining stereo correspondence from a set of linear spatial filters," in *Proceedings of ECCV*, Santa Marherita Ligure, Italy, 18–23 May 1992.

[23] J. J. Koenderink and A. J. van Doorn, "Facts on optic flow," *Biological Cybernetics*, vol. 56, no. 4, pp. 247–254, 1987.

[24] S. Laveau and O. Faugeras, "3-D scene representation as a collection of images and fundamental matrices," Tech. Rep. 2205, INRIA, Feb. 1994.

[25] J. S. Lim, *Two-Dimensional Signal and Image Processing.* Englewood Cliffs: Prentice Hall, 1990.

[26] J. Liu and R Skerjanc, "Construction of intermediate pictures for a multiview 3D system," in *Proceedings of the SPIE: Stereoscopic Displays and Applications III*, vol. 1669, pp. 10–19, San Jose, CA, 12–13 Feb. 1992.

[27] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[28] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 194, no. 4262, pp. 283–287, Oct. 1976.

[29] L. Matthies, R. Szeliski, and T. Kanade, "Incremental estimation of dense depth maps from image sequences," in *Proceedings of IEEE Comp. Vis. and Patt. Recog.*, pp. 366–374, Ann Arbor, MI, 5–9 June 1988.

[30] R. Mohr, F. Veillon, and L. Quan, "Relative 3D reconstruction using multiple uncalibrated images," in *Proceedings of IEEE Comp. Vis. and Patt. Recog.*, pp. 543–548, New York, 15–18 Jan. 1993.

[31] R. M. Murray, Z. Li, S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation.* Boca Raton: CRC Press, 1994.

[32] A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression.* New York: Plenum Press, 1988.

[33] W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics.* New York: McGraw-Hill, 1979.

[34] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 4, pp. 353–363, Apr. 1993.

[35] T.-C. Pong and B. G. Kaiser, "A hierarchical approach to the correspondence problem," *IEEE Trans. Sys. Man Cyber.*, vol. 19, no. 2, pp. 271–276, Mar. 1989.

[36] A. Shashua, "Projective structure from two uncalibrated images: Structure from motion and recognition," Tech. Rep. 1363, MIT AI Laboratory, Sept. 1992.

[37] R. Skerjanc and J. Liu, "A three camera approach for calculating disparity and synthesizing intermediate pictures," *Signal Processing: Image Communication*, vol. 4, pp. 55–64, 1991.

[38] R. Szeliski, "Image mosaicing for tele-reality applications," Tech. Rep. CRL 94/2, Digital Equipment Corporation: Cambridge Research Lab, May 1994.

[39] R. Szeliski and S. B. Kang, "Recovering 3D shape and motion from image streams using non-linear least squares," Tech. Rep. CRL 93/3, Digital Equipment Corporation: Cambridge Research Lab, March 1993.

[40] C. J. Taylor and D. J. Kriegman, "Structure and motion from line segments in multiple images," Tech. Rep. 9402b, Yale University, Center for Systems Science, Jan. 1994.

[41] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[42] R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI–6, no. 1, pp. 13–26, Jan. 1984.

[43] D. Tzovaras, M. G. Strintzis, and H. Sahinoglou, "Evaluation of multiresolution block matching techniques for motion and disparity estimation," *Signal Processing: Image Comm.*, vol. 6, no. 1, pp. 59–67, Mar. 1994.

[44] S. Ullman and R. Basri, "Recognition by linear combinations of models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 10, pp. 992–1006, Oct. 1991.

[45] J. Weng, N. Ahuja, and T. S. Huang, "Matching two perspective views," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 8, pp. 806–825, Aug. 1992.

[46] J. Weng, N. Ahuja, and T. S. Huang, "Optimal motion and structure estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 864–884, Sept. 1993.

[47] A. Zakhor and F. Lari, "Edge-based 3-D camera motion estimation with application to video coding," in *Motion Analysis and Image Sequence Processing* (M. I. Sezan and R. L. Lagendijk, ed.), ch. 4, Kluwer Academic Publishers, 1993.

[48] A. Zakhor and F. Lari, "3-D camera motion estimation with applications to video compression and 3-D scene reconstruction," in *Proceedings of the SPIE: Image and Video Processing*, vol. 1903, San Jose, CA, 3–4 Feb. 1993.

To appear at the International Conference on Acoustics, Speech and Signal Processing (ICASSP), Detroit, Michigan, 8-12 May 1995.

ARBITRARY VIEW GENERATION FOR THREE-DIMENSIONAL SCENES FROM UNCALIBRATED VIDEO CAMERAS

Nelson L. Chang and Avideh Zakhor

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720 USA
e-mail: nlachang@eecs.Berkeley.EDU, avz@eecs.Berkeley.EDU

ABSTRACT

This paper focuses on the representation and arbitrary view generation of three dimensional (3-D) scenes. In contrast to existing methods that construct a full 3-D model or those that exploit geometric invariants, our representation consists of dense depth maps at several preselected viewpoints from an image sequence. Furthermore, instead of using multiple calibrated stationary cameras or range data, we derive our depth maps from image sequences captured by an uncalibrated camera. We propose an adaptive matching algorithm which assigns various confidence levels to different regions. Nonuniform bicubic spline interpolation is then used to fill in low confidence regions in the depth maps. Once the depth maps are computed at preselected viewpoints, the intensity and depth at these locations are used to reconstruct arbitrary views of the 3-D scene. Experimental results are presented to verify our approach.

1. INTRODUCTION

In light of recent advances in technology, virtual environments have become an important tool in engineering, design, manufacturing and many other areas. Especially important to the development of this growing field is the problem of Arbitrary View Generation (AVG) in which an intermediate view of a three dimensional (3-D) scene is interpolated from its neighboring views. Existing work in this area can be classified into three classes: in the first class, a full 3-D model of the scene is constructed by volumetric intersection and then reprojected in order to generate the desired view [1]. The main difficulty with this approach is that of registering and combining the 2-D information to generate a full 3-D model. In the second class, views are generated by exploiting certain invariants in the geometry of the problem [2]. This approach however does not correctly reconstruct points that become deoccluded.

The third class of AVG algorithms attempts to deal with occluded/deoccluded regions in the scene better than the second class while not resorting to a full 3-D representation of the first class. Generally, a set of $2\frac{1}{2}$-D surfaces is first estimated and then combined to generate the desired view. For example, Chen and Williams [3] measure range and camera transformation to establish pixel correspondence and then apply morphing to interpolate intermediate views. Similarly, Skerjanc and Liu [4] compute depth with known camera positions in order to synthesize intermediate pictures.

Our approach to AVG falls into this third category [5]. However, unlike existing techniques, we use a sequence of images captured by a hand held, uncalibrated camcorder. Uncalibrated cameras with unknown position are used to avoid the difficult and time-consuming step of calibration and therefore increase the flexibility of the image acquisition process. Our motivation for using a sequence of video images rather than a few still images is to improve the robustness of the depth estimation step. Wide availability of video cameras in today's research and commercial environment justifies their use in place of still cameras in many applications.

Our proposed approach consists of scanning a camcorder across several trajectories of the scene in order to generate image sequences to be used in constructing the depth maps. The idea is to estimate depth only at several prespecified locations, called "reference frames," by using their neighboring captured frames. Once the depth has been computed at reference frames, the neighboring intensity frames are discarded, and only the depth and intensity at reference frames are kept as a compact representation of the scene. This representation is then used to reconstruct arbitrary views located on or off the scanning trajectories.

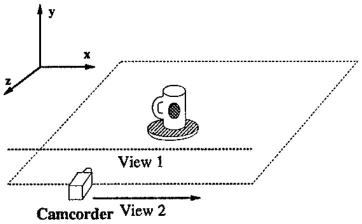


Figure 1: Experimental set up used to generate results.

In this paper, we consider a simple imaging geometry in which a camcorder is translated across the object on a line at multiple elevations, shown in Figure 1. The motivation for not choosing rotation, or a combination of rotation and translation motion, is the sensitivity of depth reconstruction to these classes of motion, especially when the motion parameters are unknown. In addition, it is well known that depth reconstruction can be more accurate when the camera translates across an object, rather than when it translates toward or away from it.

The outline of the paper is as follows. In Section 2, we discuss an adaptive approach to dense depth estimation. Section 3 describes the reconstruction algorithm. Results are presented in Section 4. The paper concludes with a discussion in Section 5.

This work was supported by an Air Force Laboratory Graduate Fellowship, PYI-NSF grant MIP–9057466, ONR young investigator award N00014-92-J-1732, and Sun Microsystems.

2. COMPACT REPRESENTATION

Our overall approach in deriving the depth information at reference locations is to establish correspondence between the reference frame and each of its neighboring frames. Theoretically speaking, it is sufficient to establish correspondence with only one of the neighbors. In practice, however, it is advantageous to do so with a large number of neighboring frames in order to improve the accuracy of the resulting depth map. Note that once these neighboring frames are used in computing the depth at the reference frames, they are discarded in the reconstruction process; therefore, their use only affects the quality of the representation and not its compactness.

After correspondence between the reference frame and each of its neighbors has been achieved, the resulting depth maps at the reference frames are normalized and combined in order to form a depth map for the reference frame. In the remainder of this section, each step will be discussed in detail.

2.1. Depth Estimation

In the first step of the representation process, local dense depth maps are generated by matching the reference frame and each neighboring frame. Existing stereo matching techniques [6] cannot be used because they assume correspondence or known camera positions. Similarly, structure-from-motion algorithms [7] estimate the structure of only a small set of feature points in the scene.

We shall assume local perfect translation between every pair of images to reduce the depth estimation problem to a 1-D correspondence matching problem [8]. In this case, the epipolar lines of the two images are parallel with the scan lines of the image. For every point (i, j), the depth may be estimated as the inverse of disparity $d(i, j)$ given by

$$d(i,j) = \min_{m \in L} \left\{ \sum_{x=i-b/2}^{i+b/2} \sum_{y=j-b/2}^{j+b/2} |I_1(x,y) - I_2(x+m,y)|^2 \right\} \quad (1)$$

where L is the appropriate epipolar line.

There are some artifacts inherent both in the algorithm and the problem itself that induce incorrect disparities for certain regions. If the relative motion between two images is translational along the x axis, then an artifact known as aperture ambiguity occurs for horizontal lines. It arises because the block B used for matching is too small and does not include enough distinct features when matching. A second artifact occurs in regions of constant intensity where disparities are incorrectly matched because the block size is again too small. Other artifacts occur in occluded regions and near depth discontinuities; see [5] for more details.

It is straightforward to identify most of these artifacts and subsequently assign confidence levels to different regions in the scene. These confidence levels are important for locating the regions to ignore when combining multiple depth maps together. To detect aperture ambiguity, a gradient-based edge detector is used to locate the horizontal edges [5]. Points in the image near these edge pixels are marked as possibly spurious. To identify constant intensity regions, a small window is used to find regions where the intensity variance is lower than a prespecified threshold. A low variance suggests that the block consists of low texture and nearly constant intensity. Occluded regions consist of the unmapped points from matching two images in both directions. Performing the match in both directions also helps to validate the matches [5]. In the end, the scene will consist of low confidence regions marked according to the different artifacts: constant intensity, aperture ambiguity, occlusion, and inconsistencies in matching.

Since many real world scenes consist largely of low textured regions, the matching algorithm will produce a high percentage of low confidence regions due to constant intensity. To avoid too sparse a depth map, we attempt to improve estimates in these regions. We propose an adaptive matching approach whereby a small block size is used to match regions near boundaries and a larger block size is used to match constant intensity regions [5]. This overcomes the well-known tradeoff between good boundary localization with a small window and improved matching in low textured regions with a large window. The final result consists of fairly dense and reasonably accurate disparities.

2.2. Normalization of Initial Estimates

The depth maps from the previous stage need to be normalized so that they are all related by the same scaling factor. For this task, we propose to estimate the translation parameter between maps and scale by the reciprocal. The relationship between disparities $\Delta u_{m,i}$ and relative motion b_m may be derived [5] to get the linear least squares solution

$$\frac{b_m}{b_1} = \frac{\sum_{i=1}^{k} (\Delta u_{1,i})(\Delta u_{m,i})}{\sum_{i=1}^{k} (\Delta u_{1,i})^2} \quad (2)$$

where b_1 is assumed to be one. Then b_m is precisely the scaling factor α_m by which we need to adjust the m-th depth map. An iterative process is used to reduce the error $\|A\alpha_m - y\|_2$ to some desired amount where outlier points greater than a given error percentage are disregarded when computing α_m.

2.3. Combination of Multiple Depth Maps

Once all the depth maps have been normalized to a common scaling factor, they are combined to form a single depth map for a particular reference frame. For every point, an iterative procedure is used to analyze the statistics of the given data, throw out outliers, and reduce the data set to a more consistent one. Points outside the range median $\pm k\sigma$ are discarded. The remaining points are combined in a weighted average based on confidence levels [5]. Depth information from matching a vertically-related pair of images is also included in combination to overcome spurious estimates due to horizontal aperture ambiguity.

2.4. Cubic B-Spline Approximation

The depth map after the combination stage is fairly accurate in many regions. There are however a considerable number of low confidence regions. To fill in these regions and to make the map much denser while not sacrificing too much accuracy, nonuniform cubic B-splines are used. Every depth point in low confidence regions is interpolated by its neighboring high confidence depth vertices along the same row or column, depending on the variance of these vertices. The depth surface is treated as a tensor product, i.e. the product of 1-D functions, so the data may be processed first along one direction and then along the other which helps to simplify computations.

Once the depth map for each reference frame has undergone spline approximation, we are left with $2\frac{1}{2}$-D surface estimates at different locations around the scene. The final step in the representation process is to estimate the relative camera motion between reference frames using an approach like [7]. Once the relative motion between all reference frames is known, a geometric relationship may be constructed among the different reference frames. This enables us to select the reference frames needed to use in the reconstruction stage.

In the end, the representation of the object consists of the intensity-depth pair at each reference location along with the relative motion among reference frames. Once these data have been derived, they may be stored in a database for later reconstruction.

3. RECONSTRUCTION OF VIEWS

Once we have generated the representation for a particular 3-D object, we may choose to reconstruct the view of the object at some specified viewpoint. Assume that the center of one reference frame coincides with the origin of the coordinate system and that the desired viewpoint is known with respect to this origin. The reconstruction algorithm consists of the following: First the appropriate reference frame(s) are chosen. Then initial estimates of the desired view are constructed by applying motion parameters to each reference frame. Finally, the estimates are combined into a single image, interpolating when necessary.

3.1. Selection of Appropriate Reference Frame(s)

Given the relative position and orientation of the desired view, it should be a straightforward task to determine which reference frames to use. One way of deciding is to include those frames with the smallest motion in norm relative to the view.

Another consideration is the number of reference frames. If the specified view is very close to one of the reference frames, then we may choose to use only that single frame. However, at least two reference frames are needed to properly reconstruct the desired view to reduce noise and to recover occluded regions in the scene.

3.2. Generation of View Estimates

The notion of applying motion parameters to a frame has been addressed in conventional computer vision literature [8]. Let (u_1, v_1) be the projection of a point in the scene onto the image plane. Suppose the frame of reference undergoes a rigid transformation (R, T) given by $R = [r_{i,j}]$ and $T = (\Delta x, \Delta y, \Delta z)'$ where both rotation R and translation T are in terms of the world coordinates. Then the new image coordinates are given by

$$u_2 = \frac{(r_{1,1}u_1 + r_{1,2}v_1 + r_{1,3})Z + \Delta x}{(r_{3,1}u_1 + r_{3,2}v_1 + r_{3,3})Z + \Delta z} \qquad (3)$$

$$v_2 = \frac{(r_{2,1}u_1 + r_{2,2}v_1 + r_{2,3})Z + \Delta y}{(r_{3,1}u_1 + r_{3,2}v_1 + r_{3,3})Z + \Delta z} \qquad (4)$$

where the focal length f is assumed to be 1.

The points of the reference frame arrays are considered not as discrete independent points, but rather as vertices of a deformable wire mesh [5] to overcome possible inconsistencies after transformation. Neighboring points in the reference frame are viewed as connected to one another. A view estimate is generated by applying equations (3) and (4) to the collection of points and examining not only the new coordinates of every point, but also the ordering in the mesh. In this manner, the ordering of points may be better preserved and inconsistencies of spurious background points appearing among foreground points in the transformed data are not as prevalent. Regions behind moving objects may become uncovered after view transformation. In this case, interpolation between consecutive points according to the mesh may be included.

3.3. Combination of Reconstructed Data

For each point, a small region around the point is considered. Outliers in the depth domain are thrown out until the variance in the intensity of the points in the region is approximately uniform. The motivation is that the points are expected to possess similar depth and intensity in the same neighborhood. This step further rules out discrepancies among the data.

During reconstruction, "holes" may be created when no points fall within a region. This condition arises because of uncovered regions in the scene, i.e. deoccluded regions, and because of sparse depth information. Generally, introducing more reference frames helps to reduce the size of these holes. For the remaining holes, the region around each point is grown until a sufficient number of points exists within the region [5].

4. RESULTS

We shall now examine some results using the techniques described above. The object of interest is a mug placed atop a stool. A CCD camcorder is moved by hand to follow trajectories at two different elevations to generate an image sequence for each trajectory, similar to the set up drawn in Figure 1. Each frame is 640×480 pixels large and consists of intensity only. We attempted to make the motion roughly translational along the x axis to demonstrate that neither a calibrated set up nor a track is needed. Moreover, no special lighting was used to film the scene; specularities of the stool and the lid of the mug are very apparent in the images.

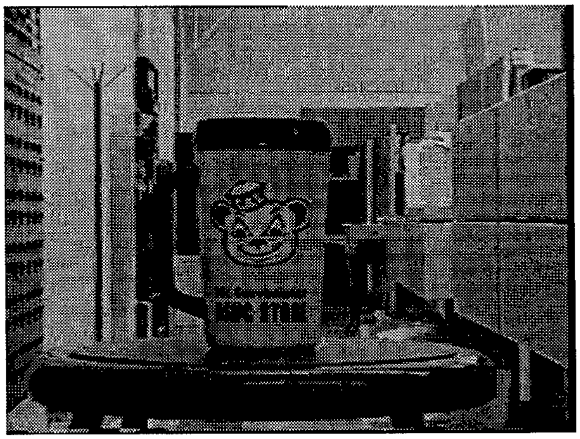


Figure 2: Example of reference frame (intensity).

For the first set of results, the desired view is roughly halfway between two reference frames along the same horizontal trajectory; one reference frame is shown in Figure 2. This desired view is perhaps the one most prone to errors due to the large occluded regions. Note that there is roughly a maximum of 120 pixel disparity between the two reference frames.

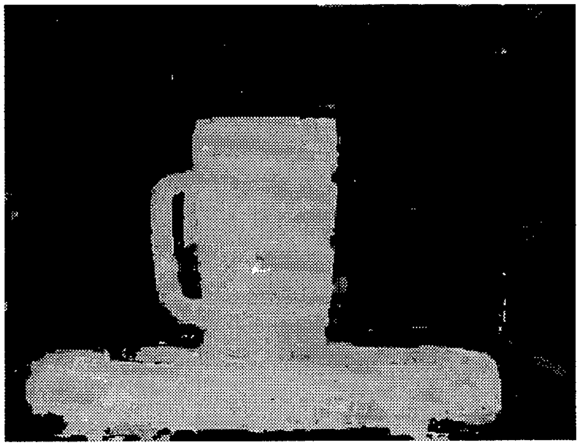


Figure 3: Example of reference frame (depth) filled in by splines.

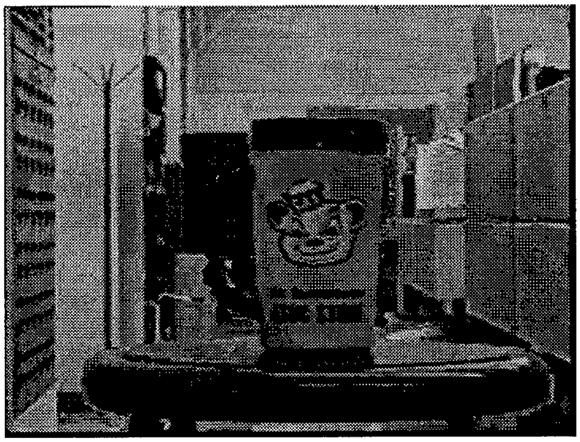


Figure 4: Reconstructed view along horizontal trajectory.

Figure 3 shows the corresponding depth map obtained by using the proposed matching algorithm. The mug and stool are estimated well and do not contain many spurious depths. There is a gradual change in depth as expected for a hallway scene. Artifacts are prevalent in the top left portion of the stool; this is primarily due to the specularities of the surface. Also, there are problems in recovering the handle of the mug accurately mainly because intensity-based matching schemes perform poorly for background regions that can be seen through foreground regions.

The reconstructed view is shown in Figure 4. The image quality is good for the most part. The horizontal edges, e.g. top of the door, top of the mug, specularities in front of the stool, and the drawers, have been reconstructed quite well. The proposed algorithms take care of problems in occluded regions: There are only a few errors to the right of the mug and near the mug handle.

To generate a view not originally scanned by the camcorder, two frames from different elevations are chosen as reference frames. The desired view is roughly the midpoint on the vertical trajectory relating the two views.

The reconstructed view in Figure 5 is a reasonable estimate of the desired view. The most noticeable artifact occurs around the upper left portion of the stool caused by specularities that result in spurious depths. This problem may be overcome by using a larger number of frames to form the combined depth map; we are currently investigating this issue.

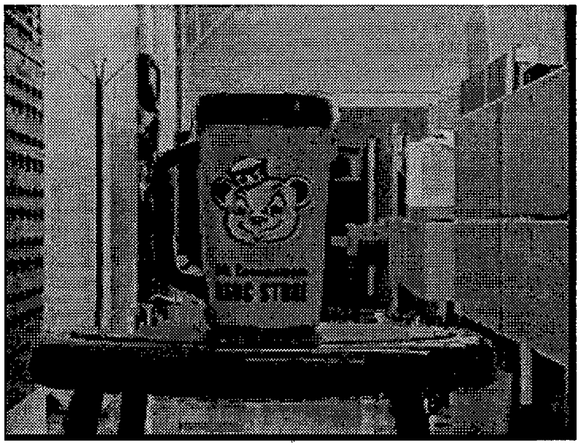


Figure 5: Reconstructed view along vertical trajectory.

5. DISCUSSION

We have proposed an approach for representing and reconstructing stationary 3-D objects. The results in the previous section seem to indicate that this approach is very worthwhile. Future work in this area includes considering more general imaging geometry and examining the optimum positions of reference frames required for a given scene. The area of arbitrary view generation and its application to virtual environments seems very fertile and this research serves as a good starting point.

6. REFERENCES

[1] C. H. Chien and J. K. Aggarwal, "Identification of 3D objects from multiple silhouettes using quadtrees/octrees," *Computer Vision, Graphics and Image Processing*, vol. 36, no. 2–3, pp. 256–273, Nov.-Dec. 1986.

[2] A. Shashua, "Projective structure from two uncalibrated images: Structure from motion and recognition," Tech. Rep. 1363, MIT AI Laboratory, Sept. 1992.

[3] S. E. Chen and L. Williams, "View interpolation for image synthesis," in *Proceedings of SIGGRAPH*, New York, 1–6 Aug. 1993.

[4] R. Skerjanc and J. Liu, "A three camera approach for calculating disparity and synthesizing intermediate pictures," *Signal Processing: Image Communication*, vol. 4, pp. 55–64, 1991.

[5] N. L. Chang, "View reconstruction from uncalibrated cameras for three-dimensional scenes," Master's thesis, University of California at Berkeley, 1994.

[6] U. R. Dhond and J. K. Aggarwal, "Structure from stereo—a review," *IEEE Trans. Sys. Man Cyber.*, vol. 19, no. 6, pp. 1489–1509, 1989.

[7] R. Szeliski and S. B. Kang, "Recovering 3D shape and motion from image streams using non-linear least squares," Tech. Rep. CRL 93/3, Digital Equipment Corporation: Cambridge Research Lab, March 1993.

[8] B. K. P. Horn, *Robot Vision.* Cambridge, MA: MIT Press, 1991.