Constructing a Multivalued Representation for View Synthesis

Nelson L. Chang
Imaging Technology Department
Hewlett-Packard Laboratories
1501 Page Mill Road, MS 4U-6, Palo Alto, CA 94304 USA
email: nlachang@hpl.hp.com

Avideh Zakhor
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720 USA
email: avz@eecs.Berkeley.EDU

Abstract

A fundamental problem in computer vision and graphics is that of arbitrary view synthesis for static 3-D scenes, whereby a user-specified viewpoint of the given scene may be created directly from a representation. We propose a novel compact representation for this purpose called the multivalued representation (MVR). Starting with an image sequence captured by a moving camera undergoing either unknown planar translation or orbital motion, a MVR is derived for each preselected reference frame, and may then be used to synthesize arbitrary views of the scene. The representation itself is comprised of multiple depth and intensity levels in which the k-th level consists of points occluded by exactly k surfaces. To build a MVR with respect to a particular reference frame, dense depth maps are first computed for all the neighboring frames of the reference frame. The depth maps are then combined together into a single map, where points are organized by occlusions rather than by coherent affine motions. This grouping facilitates an automatic process to determine the number of levels and helps to reduce the artifacts caused by occlusions in the scene. An iterative multiframe algorithm is presented for dense depth estimation that both handles low-contrast regions and produces piecewise smooth depth maps. Reconstructed views as well as arbitrary flyarounds of real scenes are presented to demonstrate the effectiveness of the approach.

Keywords

Multivalued representation, Arbitrary view synthesis, Dense depth estimation, Layered depth images (LDI), Multiframe stereo algorithm, Low-contrast region processing, Segmentation and tracking, Virtual flythroughs and flyarounds

Published as Nelson L. Chang and Avideh Zakhor, "Constructing a Multivalued Representation for View Synthesis," *International Journal of Computer Vision*, volume 45, number 2, pages 157–190, November 2001.

1 Introduction

Compact scene representation from image data spans a broad range of applications in image processing and computer vision such as video conferencing, 3-D scene modeling, virtual walkthroughs, and interactive flythroughs over the Internet. These applications dictate the size and form of the appropriate representation, and often, the representation used for one task may not be adequate for a different one.

In this paper, we are interested in the problem of view synthesis in which a user-specified viewpoint of the scene may be created directly from a representation. It is assumed that the scene is stationary and is scanned by a moving camcorder or camera. For simplicity, the imaging device will undergo either an unknown translation restricted to the x-y plane or an orbital motion about the y-axis in the scene, both of which are conducive to depth estimation [Chang and Zakhor, 1998]. Given the image data, the problem is to derive a compact representation which allows us to reconstruct the original images and also synthesize new viewpoints of the scene.

At first glance, video coding schemes might seem to be a logical solution, especially since they are designed to excel at image reconstruction. Traditional video coding schemes such as MPEG and wavelet-based techniques however fail at new view synthesis since they simply attempt to decorrelate the data. At the other extreme, 3-D modeling techniques like [Koch, 1993; Shum *et al.*, 1995] form a coherent model from input depth data. These types of approaches typically handle scenes consisting of only a single object and impose polyhedral constaints on the object.

More recently, layered representation (LR) schemes such as [Wang and Adelson, 1994; Darrell and Pentland, 1995; Sawhney and Ayer, 1996; Weiss and Adelson, 1996] have been introduced. In this case, the image data are segmented into regions exhibiting similar 2-D affine motions and then grouped into layers defined with respect to a single reference frame. LR overcomes problems of occlusion and redundancy by integrating information over the entire image set. LR can generate views with different objects removed or synthesize new views of only scenes adequately modeled by 2-D affine motions.

In contrast, image-based rendering (IBR) techniques offer a solution to the problem of new view synthesis. In this case, a subset of frames from the input data is selected to be reference views and serves to represent a given 3-D scene. View interpolation schemes is one class of IBR where a virtual view of the scene is generated by warping and interpolating the reference views. These schemes typically augment the representation with dense pixel correspondences [Chen and Williams, 1993; Laveau and Faugeras, 1994; McMillan and Bishop, 1995; Seitz and Dyer, 1996] or dense depth

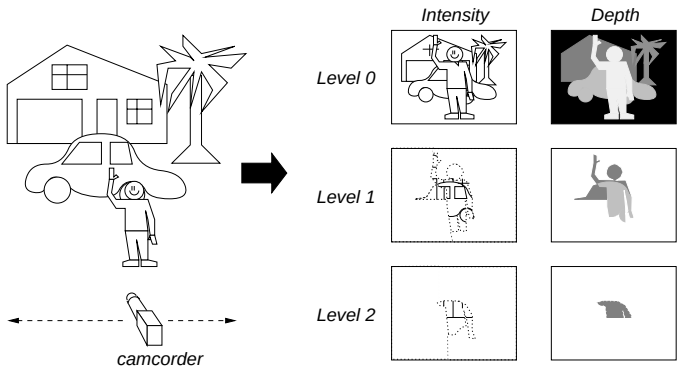
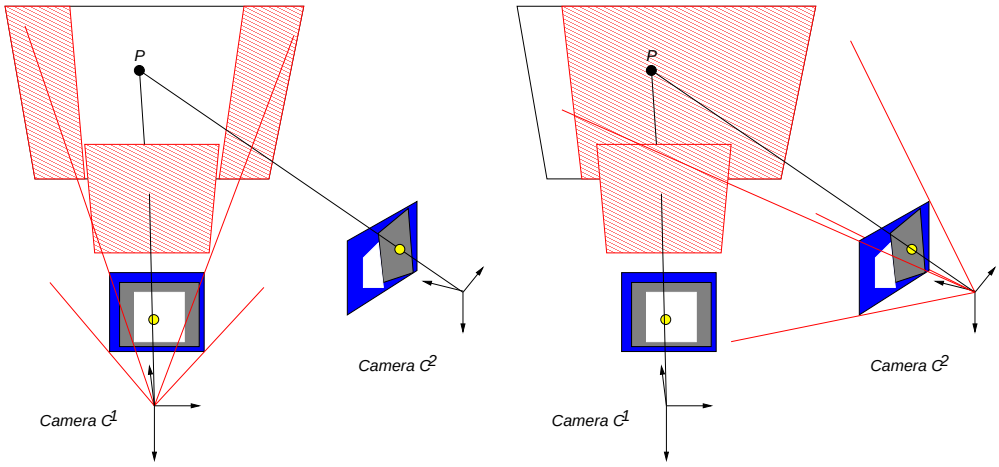


Figure 1: *Relationship between levels in MVR and occlusions in scene. Level 0 consists of the unoccluded surfaces visible to the camera. Level 1 consists of the points in the scene obscured by one surface* (e.g. *the points on the car located behind the person, the points on the house hidden by the car). Similarly, Level 2 consists of the points occluded by two surfaces (the points on the house that are occluded by both the car and the person).*

data [Chang and Zakhor, 1997b; Kanade *et al.*, 1997; Kang and Szeliski, 1997; Shum *et al.*, 1998] to provide the necessary geometrical information with which to generate realistic looking views. They perform quite well for synthesizing new views and are capable of rendering relatively complex objects. However, their representations suffer from redundancy—many points in the scene are represented multiple times in the representation. For view interpolation between two reference views, only views that lie spatially in between the reference views may be adequately generated.

A second class of IBR techniques consists of so-called light fields [Levoy and Hanrahan, 1996; Gortler *et al.*, 1996]. In this case, a very dense set of images is captured at many locations around the scene, and the entire set of images is considered to be the representation. New views are simply created by rebinning the light rays collected from different pixels in the image set. The light fields synthesize realistic views since illumination and specularities are embedded in the data set. However, the representation is overly redundant and requires a huge amount of storage. Also, light fields can only accurately synthesize views that lie outside the convex hull of the 3-D scene.

In this paper, we propose a multivalued representation (MVR) that combines the strengths of LR and IBR techniques. The representation is comprised of a dense array of intensity and depth points, possibly multivalued, with respect to a reference frame. As seen in Figure 1, the representation consists of scene points organized into different levels based on occlusion properties rather than the number of objects [Chang and Zakhor, 1997a; Chang and Zakhor, 1999; Chang, 1999]. The levels indicate the number of surfaces occluding the given point with respect to the



(a) (b)

Figure 2: *Example of a 3-D scene represented by depth maps with respect to (a) Camera* $\mathcal{C}^1$ *and (b) Camera* $\mathcal{C}^2$.

reference view—points in Level 0 are completely visible, those in Level 1 are occluded by one surface in Level 0, and so forth. Since MVR is depth-based, it can generate new views like IBR. However, MVR defines the information with respect to a single reference frame, allowing it to overcome problems of redundancy and incompleteness in a manner similar to LR.

To motivate the construction of the MVR, consider the scene in Figure 2 consisting of two planar surfaces viewed from two different camera locations. The depth map at $\mathcal{C}^1$ captures the foreground plane and parts of the background plane. Notice that the point P on the background plane is not visible from this perspective. However, it is possible to estimate depth from a different viewpoint such as $\mathcal{C}^2$ where the foreground plane does not occlude the background one. With this information, one can estimate the depth to point P with respect to $\mathcal{C}^2$. If the relative camera transformation is known, then this information may be warped into camera $\mathcal{C}^1$'s coordinate system and thus the two depth maps can be compacted and redefined with respect to only one camera's coordinate system.

Independent but parallel to our work [Chang and Zakhor, 1997a], Shade *et al.* have developed a framework similar to MVR known as layered depth images (LDI) [1998]. LDI is a representation that features possibly multiple depth values at every pixel location in the reference image. The paper alludes to methods for constructing such a representation from multiple real-world images or depth maps. It primarily focuses on issues relating to the efficient representation of and fast rendering with LDIs. It suggests that one can synthesize novel views of the given data at approximately interactive

rates. In contrast, our paper emphasizes the construction of a multivalued representation in an automatic approach. Morever, it describes a complete end-to-end system for producing and using such a representation. It also demonstrates integrating multiple MVRs together to completely capture 3-D environments.

Baker *et al.* have presented a layered stereo technique for constructing LDI from real-world images [1998]. The input images are decomposed into overlapping layers of texture and depth. However, there are some important differences between this approach and ours. With their approach, the objects in the scene are modeled as approximately planar layers, which may lead to a large number of disjoint layers. Also, the segmentation and initial layer assignment are both performed manually. In contrast, ours is an automatic approach which determines the number of levels, organizes the image data by occlusion properties, and imposes no planar constraints on the levels. The resulting MVR levels also lead to fewer gaps in the synthesized views and to more convincing synthesized views.

The remainder of the paper is outlined as follows. Section 2 presents the camera models and notation used throughout this paper. It also describes least squares techniques for estimating the camera motion from feature correspondences. Section 3 proposes a two-stage dense depth estimation scheme featuring a multiframe segmentation and plane fitting technique to estimate depth in low-contrast regions, and a multibaseline stereo approach in an iterative dynamic programming framework to derive piecewise smooth depth. Section 4 describes the steps needed to form the MVR. Section 5 includes summary and conclusions. Results of the proposed algorithms applied to real-world sequences are provided throughout the paper.

2 Camera Motion Estimation

Suppose a 3-D scene is captured by M identical cameras denoted by $\mathcal{C}^s$, $s = 1, \ldots M$.[1] Let $I^s(p)$ represent the image corresponding to camera $\mathcal{C}^s$ at pixel $p = (u, v, 1)^T$ defined in homogeneous coordinates, and let $I^s(\cdot)$ refer to the entire image. Denote the 3×3 matrix A to be the internal parameters of the cameras, *i.e.*

$$A = \begin{bmatrix} fs_u & 0 & x_0 \\ 0 & fs_v & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

[1] An alternate view is that the scene is captured by a moving camera at M discrete locations.

where f is the focal length, (s_u, s_v) is the scale factor in both directions, and (x_0, y_0) is the projection of the camera's z-axis in pixel coordinates [Faugeras, 1994]. It is assumed that the cameras do not suffer from nonlinear lens distortion and that they have already been calibrated [Tsai, 1987; Faugeras, 1994; Debevec, 1996].

Suppose two cameras $\mathcal{C}^s$ and $\mathcal{C}^t$ are related by a 3-D affine motion of rotation R^{st} and translation T^{st}. Then, a point $P^s = (X^s, Y^s, Z^s)^T$ in the scene defined with respect to $\mathcal{C}^s$ may be redefined as P^t in terms of camera $\mathcal{C}^t$'s coordinate system by the relation $P^t = R^{st}P^s + T^{st}$. The point P^t projects to the image point $p^t = (u^t, v^t, 1)^T$ given by

$$\begin{aligned} Z^t p^t &= AP^t \\ &= A\left(R^{st}P^s + T^{st}\right) \\ Z^t p^t &= Z^s \underbrace{[AR^{st}A^{-1}]}_{R} p^s + \underbrace{[AT^{st}]}_{T} \end{aligned} \tag{2}$$

where Z^s is the depth of the point P^s relative to $\mathcal{C}^s$, and Z^t is the distance of P^t relative to $\mathcal{C}^t$.[2] One may expand Equation (2) to remove the scale factor Z^t and to solve for the depth Z^s from the image points p^s and p^t. If $R = [r_k]$ and $T = (\Delta x, \Delta y, \Delta z)^T$, Equation (2) becomes[3]

$$u^t = \frac{(r_1u^s + r_2v^s + r_3)Z^s + \Delta x}{(r_7u^s + r_8v^s + r_9)Z^s + \Delta z} = \frac{r_1u^s + r_2v^s + r_3 + \Delta x\ \zeta^s}{r_7u^s + r_8v^s + r_9 + \Delta z\ \zeta^s} \tag{3}$$

$$v^t = \frac{(r_4u^s + r_5v^s + r_6)Z^s + \Delta y}{(r_7u^s + r_8v^s + r_9)Z^s + \Delta z} = \frac{r_4u^s + r_5v^s + r_6 + \Delta y\ \zeta^s}{r_7u^s + r_8v^s + r_9 + \Delta z\ \zeta^s} \tag{4}$$

with $\zeta^s = \frac{1}{Z^s}$ defined as inverse depth, a quantity which will be used more extensively in subsequent sections.

Instead of full 6-D motion estimation [Longuet-Higgins, 1981; Tsai and Huang, 1984; Tomasi and Kanade, 1992; Maybank, 1993; Hartley, 1997], a more tractable problem is to consider a class of constrained motions which are parametrized by only one or two variables. The task in this case is to estimate the appropriate motion parameters corresponding to each of the given frames with respect to a chosen reference frame. Since there are fewer degrees of freedom, the constrained motion estimates should be more reliable and consistent over multiple frames as compared to the general motion case.

[2]The matrix $AR^{st}A^{-1}$ in Equation (2) has been labeled R out of convenience. Even though it reflects the original 3-D rotation with calibration effects, R is obviously not a true rotation matrix since it is not orthogonal.

[3]The indices s and t are assumed to be a part of r_k, Δx, Δy, and Δz. They have been omitted in the equations for clarity. The indices will be used later in the text when the parameters' explicit dependence on s and t needs to be emphasized.

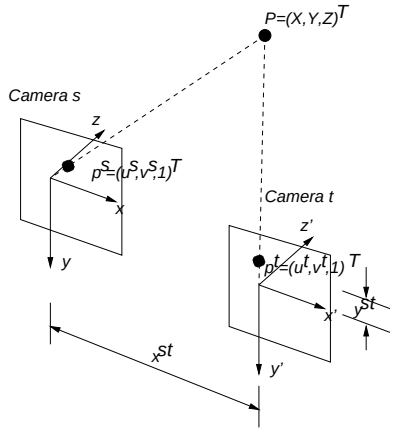


Figure 3: *Example of camera undergoing unknown translation* $(x^{st}, y^{st}, 0)$ *in the* x*-*y *plane.*

The following sections present algorithms to handle two commonly-used constrained motions: translation in the x-y plane and orbital motion around an object. The algorithms assume that feature correspondences have been obtained. For this purpose, a robust matching algorithm proposed by Zhang *et al.* [1995] is used to identify high contrast features in every frame and then to establish correspondence among the features through correlation and relaxation.

2.1 Planar Translation

In the first case, it is assumed that the camera undergoes a planar translation of the form $(R^{st}, T^{st}) = (\mathcal{I}, [x^{st}, y^{st}, 0]^T)$ between all pairs of frames s and t, where $\mathcal{I}$ is the 3×3 identity matrix; Figure 3 shows this configuration. As before, $R \triangleq AR^{st}A^{-1} = \mathcal{I}$ and $T \triangleq AT^{st} = (\Delta x^{st}, \Delta y^{st}, 0)^T$. From Equations (3) and (4), one can define disparity to be $\Delta u_k^{st} \triangleq u_k^t - u_k^s = \frac{\Delta x^{st}}{Z_k^s}$ and $\Delta v_k^{st} \triangleq v_k^t - v_k^s = \frac{\Delta y^{st}}{Z_k^s}$ for multiple feature points indexed by subscript k, $k = 1 \ldots N$. These expressions depend on Z_k^s which is fixed for a scene point with respect to a particular reference frame s over the frames that can view the point. Without loss of generality, let $s = 1$. After some algebra, the translation parameters Δx^{1t} and Δy^{1t} between frame 1 and t can be estimated through linear least squares by relating these equations to those between frames 1 and 2:

$$\Delta x^{1t} = \frac{\sum_k (\Delta u_k^{12})(\Delta u_k^{1t})}{\sum_k (\Delta u_k^{12})(\Delta u_k^{12})} \tag{5}$$

$$\Delta y^{1t} = \frac{\sum_k (\Delta u_k^{12})(\Delta v_k^{1t})}{\sum_k (\Delta u_k^{12})(\Delta u_k^{12})} \tag{6}$$

where Δx^{12} is defined to be 1, and $t = 1 \ldots M$ [Chang, 1999]. Note that these equations do not require knowledge of the calibration parameters *a priori*.

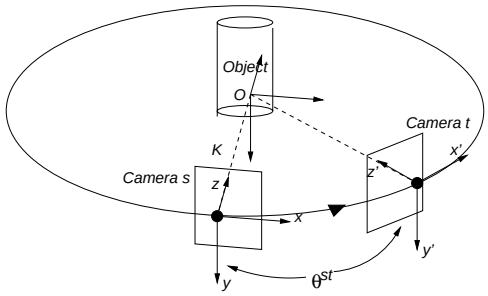


Figure 4: *Example of camera undergoing orbital motion of angle θ^{st} around object of interest.*

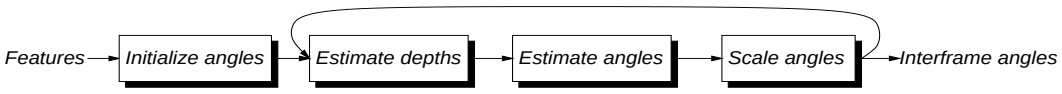


Figure 5: *Block diagram of the iterative algorithm to estimate interframe camera angles in the case of orbital motion.*

2.2 Orbital Motion

We now consider orbital motion around a given object. As drawn in Figure 4, let point O serve as the center of a circular arc of radius K about the y-axis. In this case, $R^{st} = e^{\widehat{\omega}\ \theta^{st}}$ and $T^{st} = (\mathcal{I} - e^{\widehat{\omega}\ \theta^{st}})(0,0,K)^T$, where $\omega = (0,1,0)^T$ and θ^{st} is the rotation angle to be estimated [Murray *et al.*, 1994]. Define $U^s \triangleq \frac{u^s - x_0}{f s_u}$ and $U^t \triangleq \frac{u^t - x_0}{f s_u}$, where f, s_u, and x_0 come from the calibration matrix A defined in Equation (1). Then after some rearranging, Equation (3) becomes

$$U^t = \frac{(U^s \cos\theta^{st} + \sin\theta^{st})z^s - \sin\theta^{st}}{(-U^s \sin\theta^{st} + \cos\theta^{st})z^s + 1 - \cos\theta^{st}} \tag{7}$$

where $z^s = \frac{Z^s}{K}$. Because radius K can be factored into the depth, the exact length of K becomes the global scale factor and hence is not necessary *a priori*. Since there is very little vertical displacement with this type of motion, the estimation algorithm is highly sensitive to changes in the vertical direction and thus the vertical projection component is ignored.

We employ an iterative algorithm to solve for both the depth of the points in the feature set and the interframe angles θ^{st}, $s = 1 \ldots M$ and $t = s + 1$, of the entire sequence. The steps of the algorithm are shown in the block diagram in Figure 5 and outlined as follows:

1. Initialize the angles θ^{st} to be uniformly distributed.

2. Estimate depth z_k^s corresponding to the k-th feature point using least squares. Define

$$a_k^{st} = U_k^s \cos\theta^{st} + \sin\theta^{st} - U_k^t(-U_k^s \sin\theta^{st} + \cos\theta^{st}) \tag{8}$$

$$b_k^{st} = \sin\theta^{st} + U_k^t(1 - \cos\theta^{st})) \tag{9}$$

Then, for all frames s and t for which the same k-th scene point is visible, Equation (7) results in:

$$\begin{bmatrix} a_k^{s1} \\ a_k^{s2} \\ \vdots \\ a_k^{sM} \end{bmatrix} z_k^s = \begin{bmatrix} b_k^{s1} \\ b_k^{s2} \\ \vdots \\ b_k^{sM} \end{bmatrix} \tag{10}$$

The depth z_k^s may be solved using least squares for all frames $s = 1, 2, \ldots M$ and all feature points $k = 1, 2, \ldots N$.

3. Estimate angles θ_{st} by minimization. Rearranging Equation (7) again, let

$$\alpha_k^{st} = U_k^t U_k^s z_k^s + z_k^s - 1 \tag{11}$$

$$\beta_k^{st} = U_k^s z_k^s - U_k^t z_k^s + U_k^t \tag{12}$$

$$\gamma_k^{st} = U_k^t \tag{13}$$

Then, the interframe angles θ^{st} are estimated by

$$\theta^{st} = \arg\min_{\theta^{st}} \left\{ \sum_k \left\| \alpha_k^{st} \sin\theta^{st} + \beta_k^{st} \cos\theta^{st} - \gamma_k^{st} \right\|^2 \right\} \tag{14}$$

for all frames $s = 1, 2, \ldots M$ and their neighboring frame $t = s + 1$, *i.e.* consider only one frame ahead including wraparound.

4. Scale angles so that the sum equals 360°.

5. Go back to step 2, iterate until convergence.

The algorithm typically converges within twenty iterations and produces the interframe angles for the entire sequence.

3 Multiframe Depth Estimation

Once the inter-camera motion parameters have been estimated, one can proceed to derive dense depth maps from the given image sequence. An effective depth estimation scheme for view synthesis must handle several important issues. First, the scheme should be able to compute depth information for low-contrast objects in the scene. Second, it must overcome potential matching problems due to occlusion since the more interesting scenes consist of two or more objects. Finally,

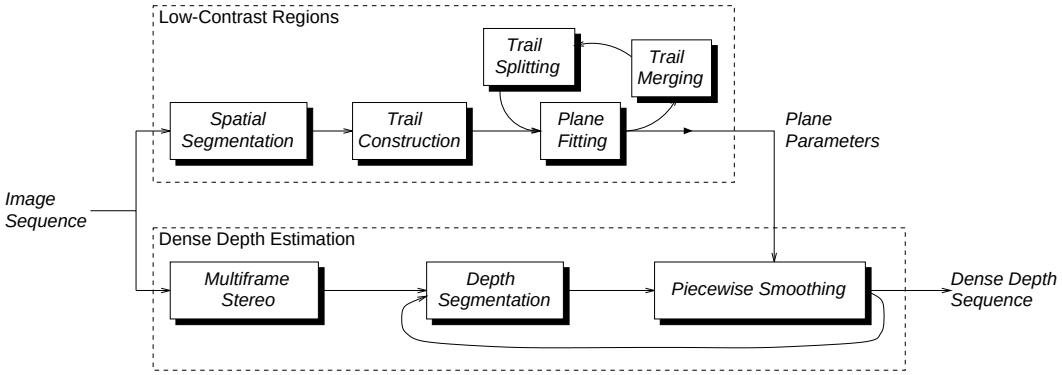


Figure 6: *Block diagram of the entire multiframe depth estimation procedure.*

it is particularly important that the depth maps are piecewise smooth since real-world objects tend to be spatially coherent.

There are a number of existing approaches to dense depth estimation. Anandan [1984] employs confidence measures to help identify points which are potentially occluded in one frame. Matthies *et al.* [1989] describe an incremental dense depth estimation technique using Kalman filtering. Anandan *et al.* [1993] propose a hierarchical scheme that uses multi-resolution matching to ensure that neighboring points have similar matches. Fua [1993] presents another multi-resolution approach which first derives accurate depth information at a sparse set of points and then interpolates these points to form a dense depth map. Chang and Zakhor [1997b] use adaptive block sizes to improve depth estimates for low-contrast regions and bidirectional matching to overcome occlusions. Although the resulting depth maps are reasonable, there is considerable variation in depth due to the lack of spatial coherence. Dynamic programming techniques like [Ohta and Kanade, 1985; Cox *et al.*, 1992; Falkenhagen, 1994; Intille and Bobick, 1994] provide a framework to explicitly smooth depth maps within regions while preserving discontinuities.

To address all three requirements mentioned at the beginning of the section, we propose a two-stage multiframe algorithm for dense depth estimation. As seen in Figure 6, the block diagram consists of two dashed boxes corresponding to the two stages of our multiframe depth estimation procedure. The top box handles low-contrast regions in the sequence and computes the depth associated with them. The bottom box then combines these results with the depth results for highly textured regions using a multiframe stereo algorithm.

Unlike previous approaches, our proposed algorithm addresses all of the above issues. First, it handles depth estimation for low-contrast regions by explicitly tracking these regions across frames and fitting surfaces that are consistent with the edge information rather than intensity information.

As described in Section 3.1, this technique results in good depth estimates in low-contrast regions.

Second, as shall be seen in Section 4, our multivalued representation overcomes occlusion by integrating multiple depth maps into a single coherent representation. Points visible in some frames but occluded in the reference coordinate system are fully recovered with this approach. Moreover, the proposed depth estimation algorithm uses a median-based multiframe stereo algorithm to mitigate potential matching errors due to occlusion for each depth map. The algorithm reliably estimates points which are visible in at least half the frames; Section 3.2.1 expands on the details of this algorithm.

Finally, our proposed algorithm imposes spatial coherence on the depth maps through an iterative dynamic programming scheme. In contrast to previous dynamic programming approaches, it is applied in both horizontal and vertical directions so that the entire depth map is piecewise smooth, not merely along epipolar lines. Also integral to the iteration is a depth segmentation step which identifies the boundaries to smooth within but not across. Furthermore, the previous low-contrast region results can be directly incorporated into this framework. The details are described below in Section 3.2.2.

3.1 Low-Contrast Regions

Many real world scenes contain regions of low contrast. Typical regions include walls and desks in indoor scenes and the sky and water in outdoor scenes. Such low-contrast regions are traditionally very troublesome for intensity-based depth estimation techniques because they lack any distinctive texture [Chang, 1994].

Without any prior knowledge, it is reasonable to assume that the low-contrast regions in the images correspond to actual objects in the scene modeled well by planar surfaces. We propose exploiting the edge information of the regions instead of using the limited texture information. In this case, the correct surface should line up the boundaries of all low-contrast regions corresponding to the same object. To proceed, the low-contrast regions in the sequence are identified and tracked across all the frames. Every tracked set of regions is fitted with the surface that is most consistent with the edge information. The depth information is simply given by the final surface through the regions. The steps of low-contrast region processing are shown in Figure 6 and are described in the following sections.

3.1.1 Spatial segmentation

The first step involves segmenting each image $I^i(\cdot)$, $i = 1 \ldots M$, into nonoverlapping regions based on intensity. A simple linking algorithm starts at every pixel in $I^i(\cdot)$ and recursively grows out regions of similar intensity [Haralick and Shapiro, 1985; Pal and Pal, 1993; Chang, 1999]. Regions smaller than 0.5% of the image are ignored since they are generally not robust for tracking. A separable 1-D median filter is applied to the segmentation map to clean up the results.

For each image $I^i(\cdot)$, the segmentation technique produces a set of disjoint regions denoted by S_k^i with index k. Each region S_k^i is the largest connected region with similar intensity. Occlusions in the scene and illumination differences are the two primary factors that affect the size, shape, and connectedness of the regions. They often cause an object in the scene to be separated into two or more regions. While there is not a one-to-one correlation between the regions S_k^i and the objects in the scene, the segmentation is still useful for identifying low-contrast problem areas for intensity-based depth estimation.

One may instead consider using a spatio-temporal video segmentation approach like [Wang and Adelson, 1994; Meier and Ngan, 1998; Wang, 1998; Shi *et al.*, 1998; Vass *et al.*, 1998] to track low-contrast regions across frames in the sequence. These types of algorithms tend to produce very coarse segmentations of objects in the scene. While applicable to video compression applications, such poor boundary localization is unacceptable for view synthesis applications.

3.1.2 Trail construction

Once the image sequence has been segmented into large distinct regions, the next step is to match these regions across multiple frames into trails. A *trail* $\mathcal{T}$ is defined to be a sequence of regions from successive frames, all corresponding to the same portion of the scene; for instance, a portion of a white wall tracked across several frames constitutes a trail. Any trail $\mathcal{T}$ can be represented in terms of the component regions as $\mathcal{T} = \{S_{K_i}^i\}_{i=i_s}^{i_f}$, where i_s is the starting frame of the trail, i_f is the ending frame of the trail, and K_i is a sequence of region ids across frames. The image sequence may be regarded as a series of trails corresponding to various low-contrast regions in the scene. Note there is a one-to-one correspondence between some, but not necessarily all, regions from one frame to those in the subsequent frame.

An iterative algorithm is used to perform the matching to derive the trails $\mathcal{T}$ for the image sequence. Each iteration consists of two primary stages: linking and validating. The linking stage examines the unlinked regions in every frame and finds the best matches in the previous and

subsequent frames.[4] Afterwards, the validation stage searches for matches that are consistent in both directions. The regions S_k^i and S_l^j are called consistent if and only if S_l^j is the best match in $I^j(\cdot)$ for S_k^i and vice versa. Consistent matches are then linked together to refer to the same region and subsequently removed from future consideration in this stage. The remaining inconsistent regions are viewed as unlinked and used in the next iteration. The algorithm repeats until no additional trails are found; typically only four or five iterations are required. Longer trails tend to be more robust for plane fitting and are thus preferred. Trails which are less than three frames in duration are ignored.

To quantify the matching step, every region S_k^i in $I^i(\cdot)$ is parametrized into four parameters: its mean intensity $S_{k\overline{I}}^i$, the number of pixels in the region S_{kN}^i, and the centroid of the region $(S_{kx_0}^i, S_{ky_0}^i)$. The region's shape is not considered as a parameter since it may be affected by occlusions in the scene. For each region in a given frame, the best matches in the previous frame and in the next frame are identified by minimizing the weighted norm of the four parameters. Specifically, matching region S_k^i in $I^i(\cdot)$ to region S_l^j in $I^j(\cdot)$ incurs a cost $J(i,k;j,l)$ given by

$$J(i,k;j,l) \quad = \quad a \left\| S_{k\overline{I}}^i - S_{l\overline{I}}^j \right\|^2 + b \left\| S_{kN}^i - S_{lN}^j \right\|^2 + c \left\| S_{kx_0}^i - S_{lx_0}^j \right\|^2 + d \left\| S_{ky_0}^i - S_{ly_0}^j \right\|^2 \quad (15)$$

where a, b, c, d are the relative weights associated with each parameter difference. The best match in $I^j(\cdot)$, $j = i+1$, is found by finding the index l which minimizes the cost $J(i,k;j,l)$. The mean intensity for the regions is weighted the most since the region is not expected to vary too much in intensity from frame to frame. Moreover, it is possible for the other parameters to vary significantly primarily due to occlusions, illumination changes, and the regions falling outside the field of view. Also, a region is skipped if the mean difference is too large or if $J(\cdot)$ exceeds a minimum threshold.

The above iteration may be further improved by using epipolar analysis to limit the number of regions to examine in the previous and next frames. For a given region S_k^i, its centroid $(S_{kx_0}^i, S_{ky_0}^i)$ is projected into camera $\mathcal{C}^j$ to compute the corresponding epipolar line $\mathcal{L}$ in $I^j(\cdot)$. Instead of checking every region S_l^j as a possible match for S_k^i, only the regions that intersect $\mathcal{L}$ are considered. Limiting the candidate regions helps to speed up the algorithm and also to reduce the number of spurious matches.

Figure 7 illustrates the two stages of the iterative algorithm applied to a five-frame example. Each image $I^i(\cdot)$ consists of a set of disjoint regions S_k^i, drawn abstractly as nodes in the figure.

[4] If the image sequence is obtained by a camera undergoing a closed motion path, the sequence wraps around itself, *i.e.* the first frame has the last frame as its previous frame and the last frame has the first frame as its subsequent frame. For nonclosed paths, there is no previous frame for the first frame and no subsequent frame for the last frame.

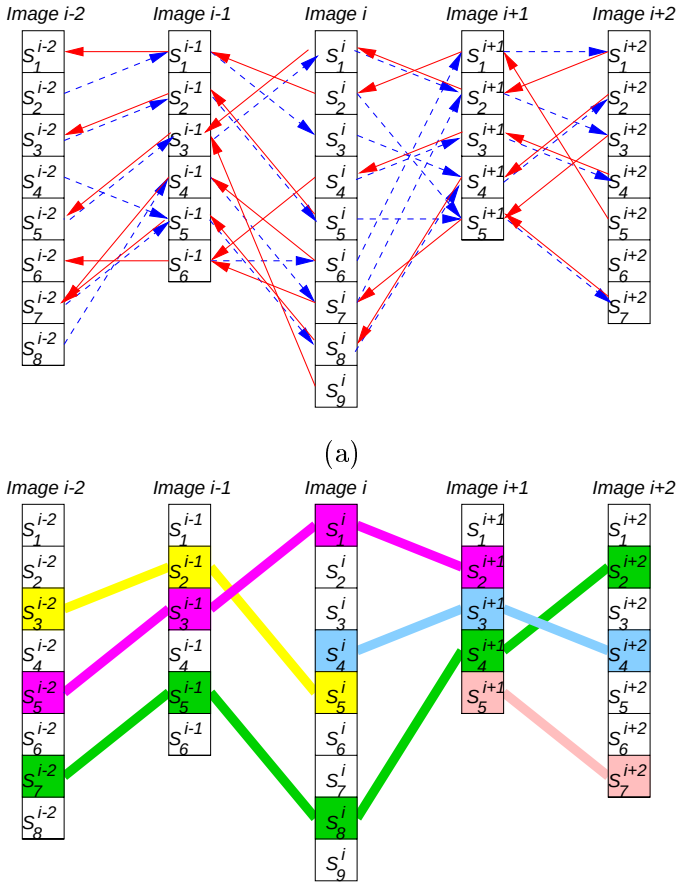


Figure 7: *Two stages of the iterative algorithm for constructing trails from a five-frame sequence: (a) linking regions to the best match in the previous and subsequent images; (b) validating the nodes to form five trails.*

The first stage, the linking stage, is shown in Figure 7 (a). Using the cost equation (15), every region is linked to its best match in the previous frame by a solid arrow and to its best match in the next frame by a dashed arrow. The regions need not have a match in the previous frame, the next frame, or either frames; for example, S_3^i has no previous link and S_9^i has no subsequent link. Moreover, it is possible to have multiple regions pointing to the same nodes, *e.g.* S_1^i and S_9^i both point to S_3^{i-1} as its best match in $I^{i-1}(\cdot)$.

The second stage of the iteration, shown in Figure 7 (b), consists of validating the links and analyzing every frame for doubly-linked nodes. The trails are implicitly constructed in this manner. In this iteration, a total of five trails were found: $\mathcal{T}_1 = \{S_3^{i-2}, S_2^{i-1}, S_5^i\}$, $\mathcal{T}_2 = \{S_5^{i-2}, S_3^{i-1}, S_1^i, S_2^{i+1}\}$, $\mathcal{T}_3 = \{S_7^{i-2}, S_5^{i-1}, S_8^i, S_4^{i+1}, S_2^{i+2}\}$, $\mathcal{T}_4 = \{S_4^i, S_3^{i+1}, S_5^{i+2}\}$, and $\mathcal{T}_5 = \{S_5^{i+1}, S_7^{i+2}\}$; these are indicated

(a) (b) (c)

Figure 8: *Three-frame example of spatial segmentation and trail construction from the Tea Box sequence: (a) Frame 38, (b) Frame 39, (c) Frame 40.*

by the different colored paths in the figure. Since $\mathcal{T}_5$ spans only two frames, it is discarded as an invalid trail. All remaining unlinked nodes are considered again for matching/linking in the next iteration, and the algorithm stops once no more trails are found.

The above steps have been applied in Figures 8 (a)–(c) to three frames from the Tea Box sequence described later in Section 3.4. Regions that span across these frames have been tracked and labeled with the same color to indicate they form the same trail. Points in black are not identified as low-contrast regions. Because the algorithm searches only immediate neighboring frames for connected regions, it is possible to inadvertently split off parts of a region. In Figure 8 (b), the post from the camera apparatus occludes part of the right cubicle wall in light grey. Since the post effectively separates the two observed regions of the cubicle wall, the algorithm divides the region in Figure 8 (a) into two parts in Figure 8 (b): one light grey region corresponding to the right side of the cubicle wall and one white region corresponding to the left side of the cubicle wall. Even though this splitting is not incorrect, it is preferred that the two trails be merged as one; trail merging is addressed in Section 3.1.4.

3.1.3 Plane fitting

The next step entails finding 3-D surfaces that are consistent with the trail information. Since intensities are not particularly useful in low-contrast regions, we instead focus on the edges of the regions. For each trail, the correct surface is expected to register the edge points the best. Since there is limited information found in the low-contrast regions, a reasonable assumption is that they may be well approximated by planes. Thus, the plane which maximizes the number of overlapping edge points is selected. Figures 9 (a) and (b) show a 2-D example of finding the correct plane to corroborate the edge data. A and B are the edge points of a low-contrast region in $\mathcal{C}^s$ while C and

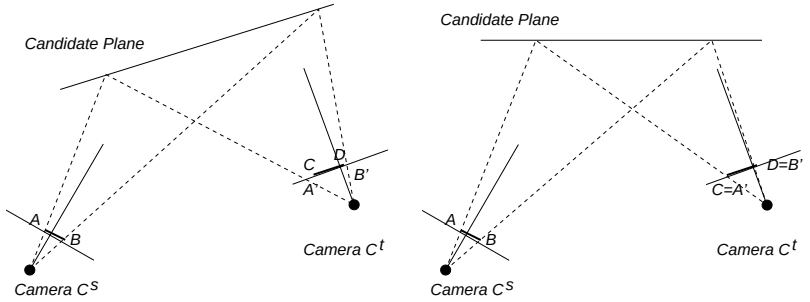


Figure 9: *2-D examples of fitting planes to maximize low-contrast region boundary overlap: (a) improper fitting and (b) proper fitting.*

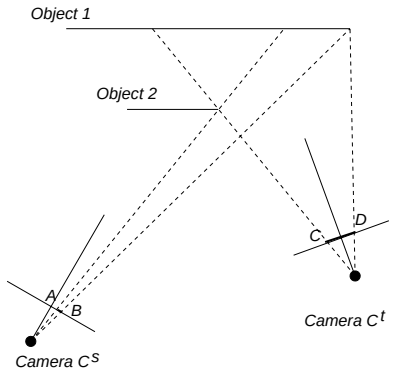


Figure 10: *2-D example of true edge points and false edge points due to occlusion.*

D are the corresponding edge points in $\mathcal{C}^t$. The projections of edge points A and B in $\mathcal{C}^s$ are A' and B' in $\mathcal{C}^t$ respectively. Notice with the plane in Figure 9 (b), the edge points A' and C overlap as do B' and D, suggesting that this plane is the correct surface unlike the plane in Figure 9 (a).

The edge points of every low-contrast region consist mainly of the true boundaries of the region but may also include false edges due to occlusion. As an example, consider Figure 10 which provides a 2-D example of two planar objects seen in two cameras. Suppose the low-contrast regions in both cameras correspond to the visible portion of Object 1. Notice in this case that edge points A and C are not the true boundary of the object, but rather a false edge arising from the occluding contour of Object 2. Clearly, if most of the edge points of a low-contrast region are false, the estimated plane will be erroneous. Because of this observation, it is assumed that the majority of the edge points are actual boundary points.

To find the optimal plane for a given trail, we exhaustively search the quantized space of possible plane parameters with respect to a reference frame. Since there is a convenient plane

(a) (b) (c)

Figure 11: *Three-frame example of trail merging from the Tea Box sequence: (a) Frame 38, (b) Frame 39, (c) Frame 40.*

parametrization with only three parameters (a, b, c) defined with respect to the image coordinates [Chang, 1999], this results in a three-dimensional search. For every trail, we consider only those candidate plane parameters which ensure that every frame in the trail observes the same side of the plane. For each set of parameter values, every edge point from one frame is projected into every other frame and the number of overlapping edge pixels is counted. The plane parameters which maximize this count represent the plane for the trail. In the end, there is an estimate of the plane parameters for every trail in the sequence.

3.1.4 Trail merging and splitting

To address potential problems from the previous steps, it is necessary to examine the trails for possible merging and splitting. The merging process begins by projecting each trail into every frame according to the current plane parameters. Two trails are merged if they satisfy the following criteria: (a) they have similar intensity; (b) the projections of the first trail overlap a significant portion of the second trail; and (c) the projections of the second trail overlap a significant portion of the first trail. This stage is repeated until no more trails can be merged. Figures 11 (a)–(c) show the effect of merging on the three-frame example. The two trails associated with the lower right cubicle wall have been automatically merged. While it is not required to merge multiple trails associated with the same object, merging provides more constraints and thus yields a more reliable plane fitting.

The splitting process may be needed when the spatial segmentation phase incorrectly merges two distinct regions together; this phenomenon can occur when the intensity of the two regions are very similar and they are spatially near each other without a well-defined boundary in between. Assuming that this problem occurs in less than half of the frames associated with the trail, one

can simply take the corresponding region in each frame and project the points using the estimated plane parameters to a common coordinate system. This projection forms the spatial extent of the trail in this common coordinate system. The number of projections at every pixel is then counted and pixels with fewer than 50% of the surrounding frames contributing are removed from the trail. If the cut-out regions are reasonably sized, it is very likely that these regions match an existing trail.

The last steps of plane fitting and trail merging and splitting are repeated until convergence as shown in Figure 6—usually only a couple of iterations are needed. The final result consists of the segmentation information and corresponding depth information for various low-contrast regions in the image sequence. This information is especially important when the image data consist of a large percentage of low-contrast regions and it will become useful when integrated with the multiframe depth estimation described in the next section.

3.2 Dense Depth Estimation

The previous section describes how to track low-contrast regions and estimate planes for them. A method is still needed for estimating depth for the remaining textured points in the sequence. We propose a multistage algorithm to accomplish this task. First, an initial dense depth map is formed by using a multiframe stereo algorithm. The depth map is then clustered into regions of similar depth. This segmentation information is fed into a dynamic programming algorithm which smoothes within clusters but not across them and integrates the results for low-contrast regions from Section 3.1. These steps are iterated to form a piecewise smooth depth map for various frames of the sequence. The block diagram for the proposed depth estimation algorithm is shown in Figure 6 and discussed in greater detail in the following sections.

3.2.1 Multiframe stereo

In order to derive an initial estimate of depth with respect to a particular frame, a variant of Okutomi and Kanade's multiple-baseline algorithm is used [1993]. Our approach consists of finding the inverse depths that minimize the median of component intensity errors or least median of squares [Rousseeuw and Leroy, 1987]. More precisely, suppose there are M images denoted by $I^i(\cdot)$, $i = 1, 2, \ldots M$, and let $k \in 1, 2, \ldots M$ be the index corresponding to the reference frame. As before, let $R = AR^{ki}A^{-1} = [r_l]$ and $T = AT^{ki} = (\Delta x, \Delta y, \Delta z)^T$ be the motion parameters relating frames k and i. According to Equations (3) and (4), a point $q = (x, y, 1)^T$ with inverse depth $\zeta^k(q)$

in frame k maps to the image coordinates $G^{ki}(q;\zeta^k(q))$ given by

$$G^{ki}(q;\zeta^k(q)) \quad = \quad \left(\frac{r_1 x + r_2 y + r_3 + \Delta x\ \zeta^k(q)}{r_7 x + r_8 y + r_9 + \Delta z\ \zeta^k(q)}, \frac{r_4 x + r_5 y + r_6 + \Delta y\ \zeta^k(q)}{r_7 x + r_8 y + r_9 + \Delta z\ \zeta^k(q)}, 1 \right)^T . \tag{16}$$

Then, our goal is to compute inverse depth $\zeta^k(p^k)$ for every desired pixel $p^k = (u^k, v^k, 1)^T$ in frame k using the following expression:

$$\zeta^k(p^k) \quad = \quad \arg\min_{\zeta^k(p^k)} \left(\text{median} \left\{ \sum_{q \in} \sum_{\mathcal{N}(p^k)} \left\| I^k(q) - I^i(G^{ki}(q;\zeta^k(p^k))) \right\|^2 \right\}_{i \neq k}^{M} \right) \tag{17}$$

where $\mathcal{N}(p^k)$ is a local neighborhood around p^k. Our algorithm computes the sum of intensity errors between points q near p^k in frame k and the predicted points $G^{ki}(q;\zeta^k(p^k))$ in frame i for all frames $i \neq k$ and then uses the median error as the overall cost for a particular candidate inverse depth $\zeta^k(p^k)$.

Our implementation of the multiple-baseline algorithm differs from Okutomi and Kanade's in a couple of ways. First, the objective function is formulated to compute the depth for sequences generated by arbitrary motion, rather than strict horizontal or vertical motion. Because large baselines may be used, occlusions in the scene will pose a larger problem in multiframe matching. The effects of occlusions are mitigated by using the median in Equation (17) instead of the sum of component intensity errors [Chang, 1999]. For points seen in over half of the frames, taking the median will remove the frames for which the points are occluded. Blindly including all frames in the minimization may lead to spurious results.

It seems intuitive to use this multiframe stereo algorithm alone to estimate dense depth for the sequence. Moreover, one could replace the depth results from this algorithm by the low-contrast region results of the previous section to achieve an improved depth map. A drawback of such an approach though is that it does not enforce piecewise smoothness. Smoothness is particularly important for view synthesis applications because (a) the real world consists of spatially coherent objects which are expected to move in a fairly rigid fashion, and (b) the human observer is very sensitive to artifacts due to nonrigid distortions. The next section improves on the depth results and imposes smoothness on the data set.

3.2.2 Piecewise smoothing

Given a jagged depth map, it is rather trivial to filter these estimates to produce a smoother map, however it is problematic to ensure that the depth discontinuities are well preserved. While it is preferred to identify these depth boundaries *a priori*, it is quite difficult to locate them among

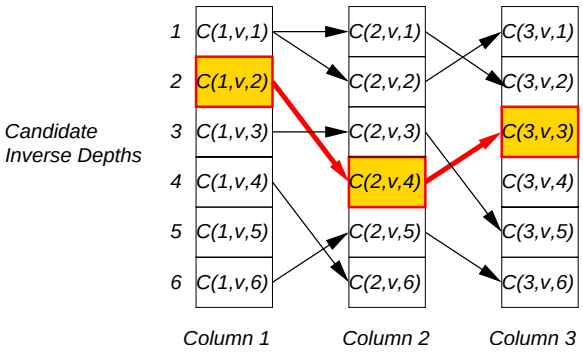


Figure 12: *Example of using dynamic programming to find the minimum path.*

noisy depth estimates. We apply a two-step iterative algorithm to impose smoothness on the depth estimates. The algorithm first segments each depth map to identify coarse region boundaries and then smoothes the depth information within these boundaries but not across them. These two steps are alternated to improve the localization of the true boundaries.

The first step is to find coherent regions in the depth maps through segmentation. Every depth map is clustered into nonoverlapping regions of similar inverse depth. The recursive algorithm described in Section 3.1.1 is used to identify large connected regions. Instead of grouping by similar intensities, the modified algorithm seeks out regions of similar inverse depth.

The second step uses dynamic programming (DP) to estimate depth in a multiframe setting subject to smoothness constraints. Previous DP approaches [Ohta and Kanade, 1985; Cox *et al.*, 1992; Falkenhagen, 1994; Intille and Bobick, 1994] estimate piecewise smooth image displacements to solve the correspondence problem between only two images. In contrast, the proposed DP algorithm finds minimum solution paths through inverse depth data from multiple frames. Inverse depth has the desirable quality of better resolution for closer objects. In addition to enforcing piecewise smoothness, the dynamic programming framework allows us to integrate the low-contrast results from Section 3.1.

Without loss of generality, consider the data associated with a single row in the image—the same principles may be applied to other rows and columns. We quantize the allowable inverse depth space into finitely many values and, for every pixel location along the row, the inverse depth can take on one of these values. Denote $\mathcal{D}$ to be the set of allowable inverse depths and suppose the members of $\mathcal{D}$ are indexed from 1 to $|\mathcal{D}|$. Let $C(u, v, d)$ be the cost of choosing inverse depth index $d \in \mathcal{D}$ for pixel (u, v). Consider the three-column example shown in Figure 12. Suppose the set of candidate inverse depths $\mathcal{D} = \{0.2, 0.4, 0.6, 0.8, 1.0, 1.2\}$ is indexed from 1 to 6, respectively. Let g_j be the inverse depth index for pixel (j, v) and denote $G = \{g_j\}$ to be the set of these indices for a row

of pixels indexed by column j. Then, the solution path G through the 2-D data incurs a cost $T(G)$ equal to the sum of component costs at every pixel in the row, or $T(G) = \sum_j C(j, v, g_j)$. In the example, the highlighted path $G = \{2, 4, 3\}$ has a cost of $T(G) = C(1, v, 2) + C(2, v, 4) + C(3, v, 3)$. The dynamic program then simply finds the path G which minimizes $T(G)$.

The component cost $C(u, v, d)$ consists of several terms and, omitting the arguments, may be written as

$$C \quad = \quad J + C_c|\Delta e| + C_d + C_i|\Delta d| \tag{18}$$

The first term J is simply the median intensity error in Equation (17) evaluated at pixel (u, v) with inverse depth index d. The next term $C_c|\Delta e|$ controls the amount of smoothing within the same region given by the above depth segmentation. C_c is large within a depth cluster, small at the end of clusters, and zero between clusters. $|\Delta e|$ is the absolute difference between the candidate inverse depth index and the previous pixel's index. Clearly, this term increases when the current index differs greatly from the previous pixel's. The third term C_d is an additional cost incurred by any large jumps in inverse depth between successive pixels. The final term $C_i|\Delta d|$ penalizes for deviations from the current depth map and the previous iteration's depth map. C_i is a scale factor which is updated every iteration through the dynamic program while $|\Delta d|$ is the absolute difference between the candidate index and the previous iteration's index at pixel (u, v).

The dynamic program works as follows. For a given pixel (u, v), every candidate in $\mathcal{D}$ is considered for pixel $(u + 1, v)$ and only the minimum candidate $d \in \mathcal{D}$ is retained. This process is repeated for every pixel in row v, thereby constructing a trellis of possible solution paths. In the end, the path that produces the smallest overall cost is chosen as the inverse depths for the row.

This DP framework also factors in the results for low-contrast regions. In this case, the dynamic program reduces the candidate set $\mathcal{D}$ to only the values obtained from the plane fitting stage in Section 3.1. Without such a modification, it is very likely to arbitrarily smooth within these regions since the median intensity error J is a shallow function of inverse depth index d. Moreover, the plane fitting stage produces reasonably accurate depth for low-contrast regions, and these values should be used in the final results.

To improve inter-row and inter-column coherence, the dynamic program is executed along the vertical and horizontal directions, *i.e.* for every column slice and row slice respectively. This is in contrast to other DP algorithms [Falkenhagen, 1994; Intille and Bobick, 1994] which lack spatial coherence across rows, resulting in horizontal streaks in the depth map [Chang, 1999]. The scale factor C_i is increased at every iteration to enforce convergence and reduce the number of iterations

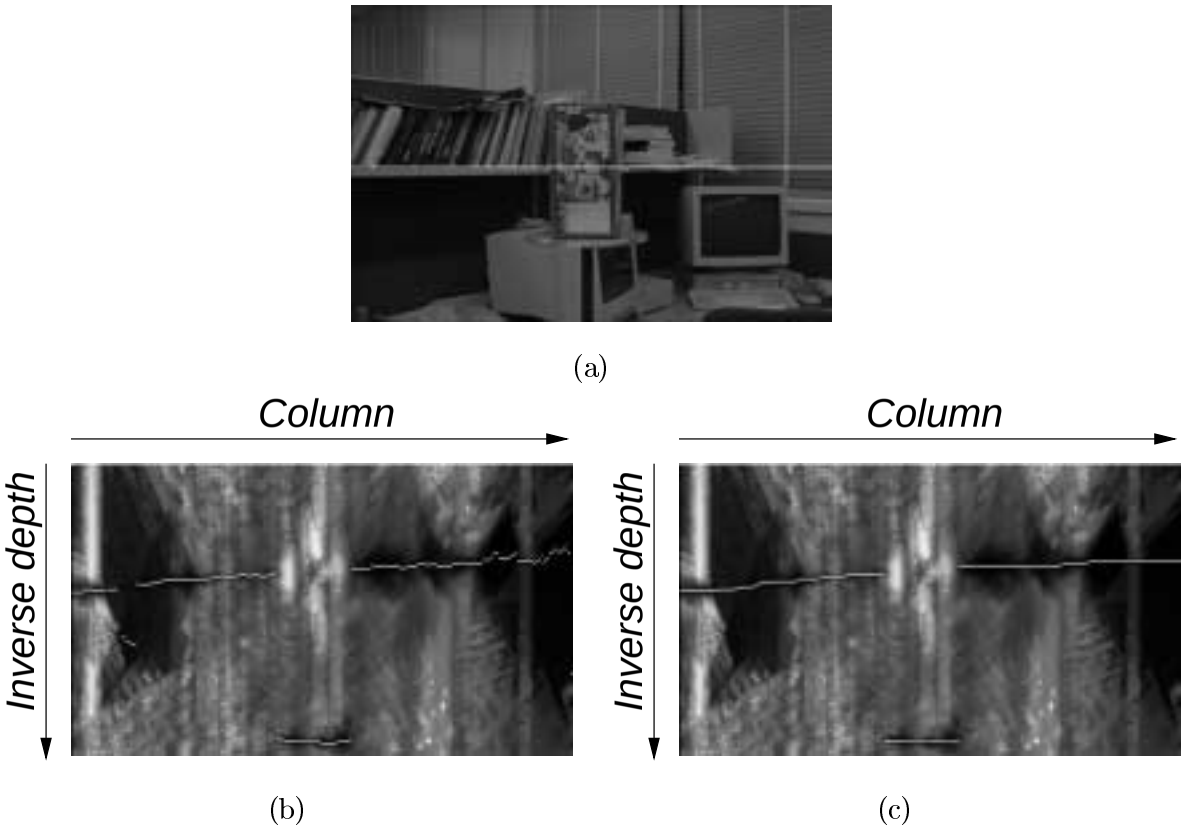


Figure 13: *Comparison between multiframe stereo algorithm and the proposed DP scheme. (a) Frame 51 of the Tea Box sequence with row 129 highlighted; (b) Plot of the error function with the minimum highlighted for every column; (c) Plot of the error function with the computed DP path highlighted.*

needed; typically, only a handful of iterations are necessary to converge to the final result.

To compare the performance of the proposed DP algorithm and multiframe stereo alone, consider the example shown in Figures 13 (a)–(c). The multiframe stereo algorithm described in Section 3.2.1 produces an error measure for a given pixel location in the image and for a particular inverse depth. Suppose we are interested in estimating depth for the pixels in row 129 of Frame 51 of the Tea Box sequence, highlighted in white in Figure 13 (a). For this row of pixels, one can construct the 2-D array of stereo errors where the horizontal axis consists of the columns in the row and the vertical axis denotes the allowable values of inverse depth. As shown in Figure 13 (b), the multiframe stereo algorithm alone results in selecting the minimum values of every column in this 2-D error array. Since no explicit spatial coherence has been exploited, the resulting path, shown in white, appears to be extremely jagged. In contrast, adding the above dynamic programming

algorithm helps to smooth the inverse depth path immensely. The path in Figure 13 (c) is much less jagged, and yet does not compromise the sharp discontinuities of the foreground object.

3.3 Complete Algorithm

To summarize the last two sections, the following steps shown in Figure 6 are performed in order to produce piecewise smooth dense depth estimates.

Algorithm:

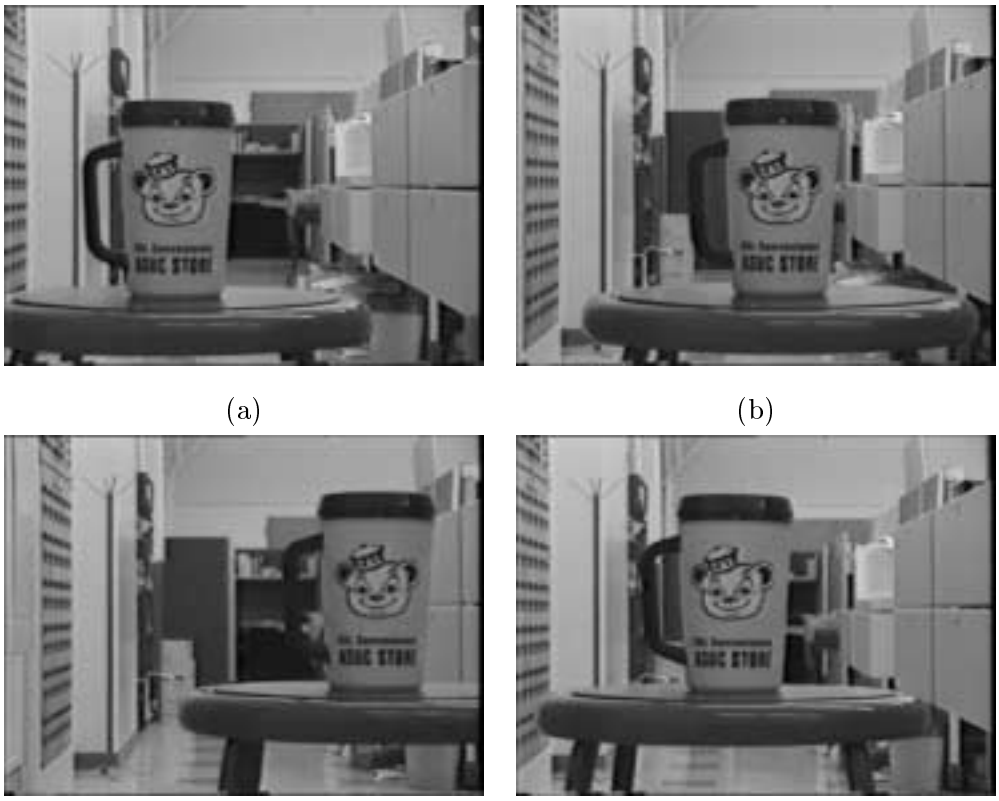
1. Perform low-contrast region segmentation, tracking, and plane fitting (Section 3.1).
2. Obtain initial depth estimates with the proposed multiframe stereo algorithm (Section 3.2.1).
3. Segment the depth maps to form initial regions
4. Apply dynamic programming along vertical and horizontal directions to enforce piecewise smoothness and to factor in low-contrast region results (Section 3.2.2).
5. Go back to step 3 and iterate until convergence.

The above steps provide an automatic method for deriving dense depth maps for frames in the image sequence.

3.4 Results

Two real-world sequences are considered to demonstrate the performance of the aforementioned algorithms. The first sequence consists of a mug sitting on a stool filmed by a camcorder undergoing unknown horizontal translation at two different elevations. Nine 320×240 frames are chosen from one elevation of the so-called "Mug" sequence and four from the other. Frames 0, 4, 9, and 11 from the Mug sequence are shown in Figures 14 (a)–(d) respectively. The planar translation parameters of all the frames are estimated using the algorithm described in Section 2.

The segmentation and tracking algorithm from Section 3.1 is used to identify low-contrast regions in the Mug sequence and to estimate depth for these regions. Figures 15 (a)–(d) show an example of tracking the low-contrast regions across the corresponding frames in Figures 14 (a)–(d), respectively. The algorithm indicates tracked regions with similar colors; all the objects shown have been correctly matched.



(a) (b)

(c) (d)

Figure 14: *Typical frames from the Mug sequence: (a) Frame 0, (b) Frame 4, (c) Frame 9, (d) Frame 11.*

Dense depth maps may be obtained for every frame by applying the algorithms in Section 3.2. Figures 16 (a)–(d) show examples of dense depth maps[5] corresponding to Figures 14 (a)–(d) using only the multiframe stereo algorithm described in Section 3.2.1. While the mug and stool are somewhat discernible, it is evident that the boundaries have been poorly localized. Also, the low-contrast regions like the background wall and the drawers have spurious depth estimates. In contrast, Figures 17 (a)–(d) show the resulting depth maps after performing the low-contrast region tracking of Section 3.1 and the iterative piecewise smoothing of Section 3.2.2. It should be clear that the proposed multiframe depth estimation algorithm produces much smoother depth maps. Also, the estimates of depth in low-contrast regions are quite reasonable.

For a second real-world sequence, a digital camera undergoing unknown orbital motion of radius 65 cm is used to capture 102 378×252 frames of the "Tea Box" sequence. The scene consists of a

[5]The depth maps have been quantized to 256 grey values where the depth is inversely related to the brightness. The maps have also been individually histogram equalized to improve visibility and to show the contrast between the object and the surrounding background.

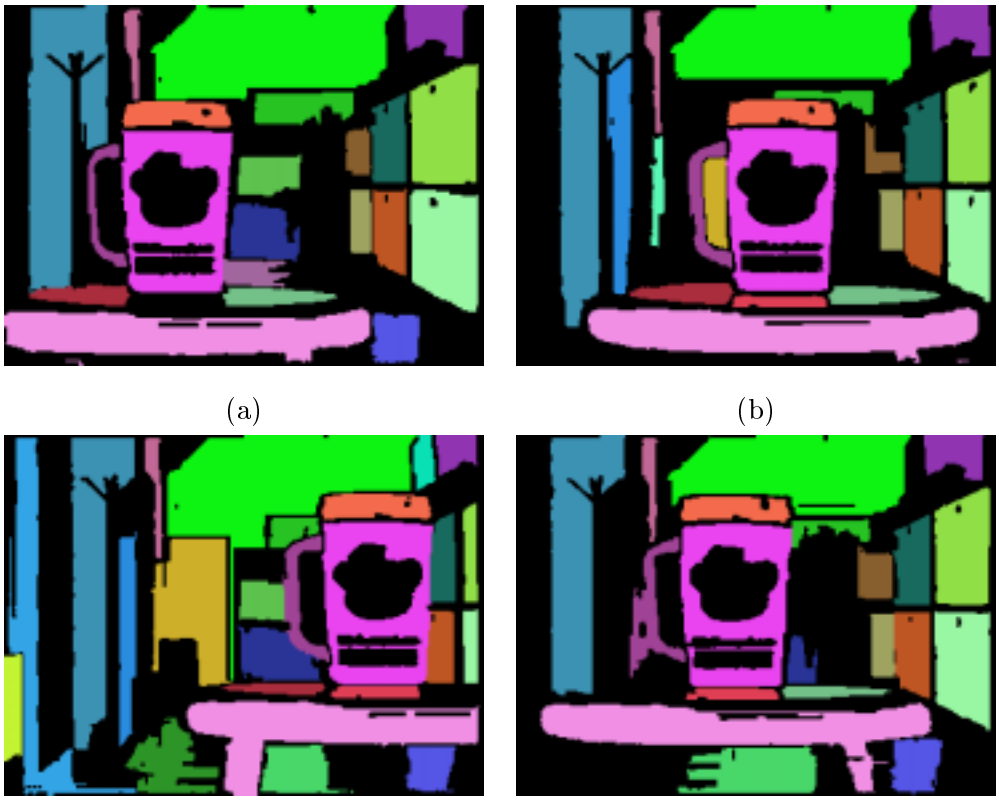


Figure 15: *Low-contrast segmentation and tracking results for Mug sequence: (a) Frame 0, (b) Frame 4, (c) Frame 9, (d) Frame 11.*

tea box placed atop the camera apparatus in a typical office environment. A subset of frames from this sequence is shown in Figures 18 (a)–(d). Notice the large percentage of low-contrast regions in the scene. The inter-camera angles are estimated between every frame using the approach in Section 2.

Figures 19 (a)–(d) show the results of the low-contrast region segmentation and tracking algorithm from Section 3.1 applied to the Tea Box sequence. Similar to the results for the Mug sequence, the algorithm is quite successful in tracking regions across the frames of the sequence. The corresponding dense depth maps using the proposed multiframe depth estimation algorithm are shown in Figures 20 (a)–(d). Again, they are reasonable given the complexity of the scene.

These depth maps, along with their corresponding intensity frames, can be used directly in any view interpolation scheme to synthesize new views. However, the multivalued representation described in the next section is a more compact way to represent these data while still capable of view synthesis.

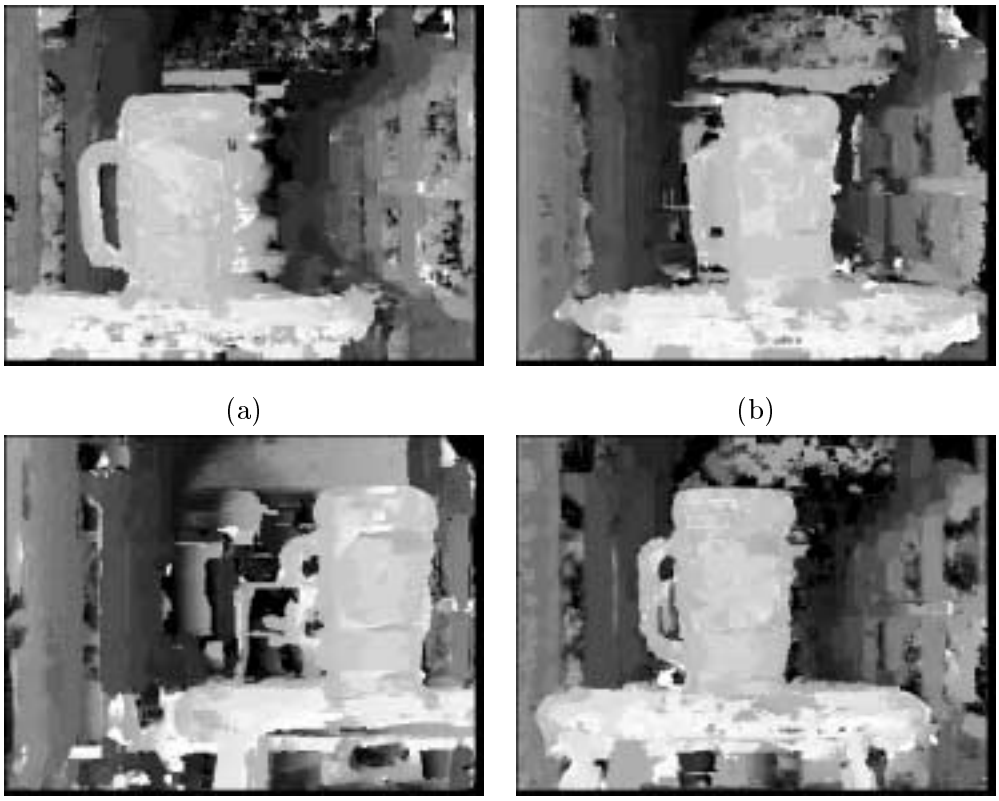


Figure 16: *Dense depth maps computed for the Mug sequence using only multiframe stereo: (a) Frame 0, (b) Frame 4, (c) Frame 9, (d) Frame 11.*

4 Multivalued Representation

From the previous section, the scene may be modeled as a set of intensity maps with corresponding dense depth information. One approach to further compact this representation is to warp all the information to a reference coordinate system, commonly one of the images from the data set. Redundant points in the scene are then discarded and only the essential information is retained. By doing so, the previously occluded regions from the reference viewpoint may be recovered with the information from different viewpoints. This observation automatically leads to the different levels in the MVR.

One may wonder why grouping by occlusions is a much better solution than grouping by affine motions. One primary reason is that the occlusion levels come naturally from the dense depth information. Because of visibility limitations, real-world scenes typically do not have more than three occlusion levels which contrasts the possibly large number of layers in layered representations. Also, there should be fewer occlusions in the representation with each successive level progressively

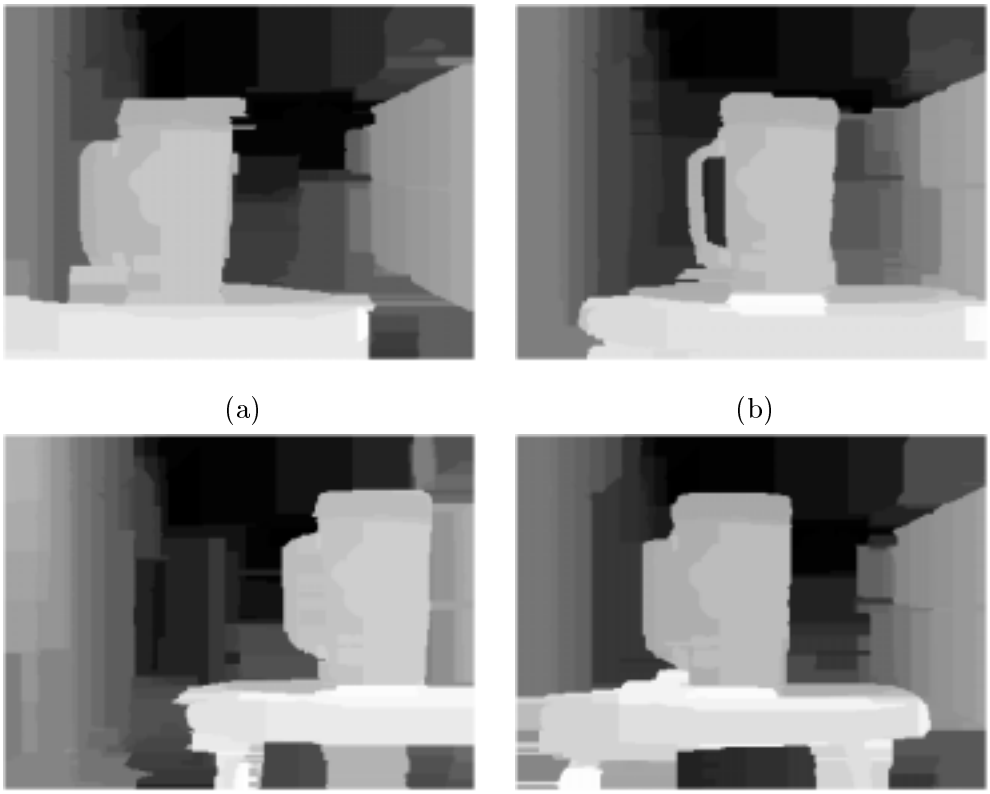


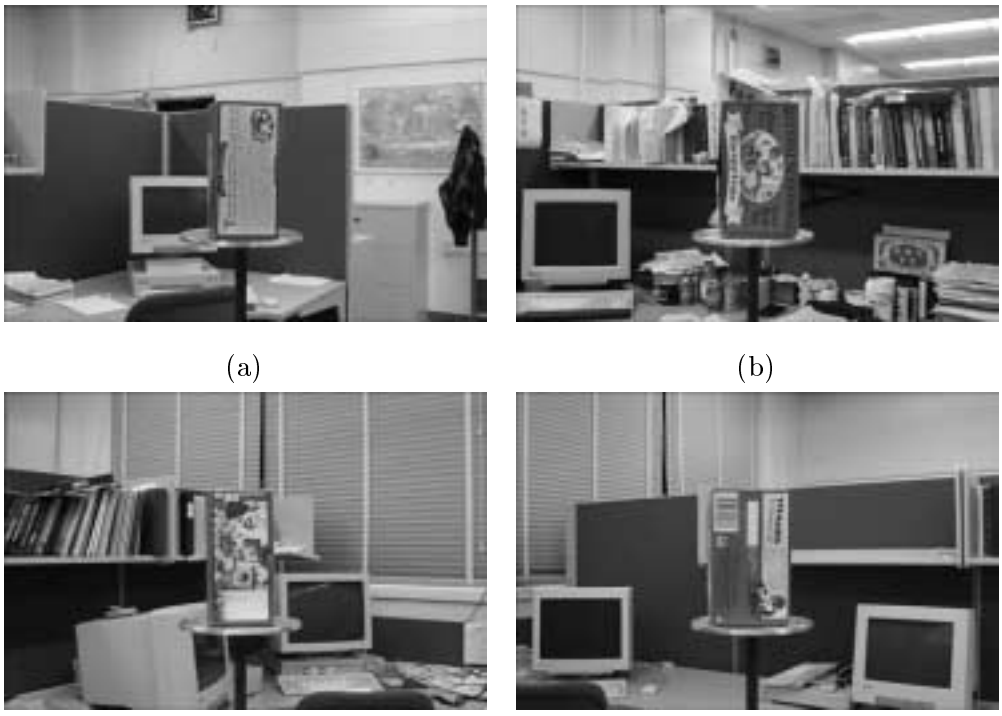
Figure 17: *Dense depth maps computed for the Mug sequence using multiframe stereo with low-contrast region processing and piecewise smoothing: (a) Frame 0, (b) Frame 4, (c) Frame 9, (d) Frame 11.*

smaller in size. This reduction helps improve the compactness of the MVR.

The section is outlined as follows. Section 4.1 details the steps to construct a MVR from a set of depth maps. Section 4.2 discusses issues in synthesizing novel views with MVRs. Finally, Section 4.3 presents results of MVRs and synthesized views for two real-world sequences.

4.1 Analysis Issues

It might appear straightforward to construct a MVR from a set of depth maps simply by warping all the data to a common reference frame and clustering the projected results. Certainly, this statement is true when one starts out with accurate depth maps and known camera motion. In practice, however, constructing a MVR from real-world data requires special processing. Inaccuracies in estimated camera motion and dense depth maps may lead to an inconsistent MVR if the data are merely warped and clustered. If left unchecked, the MVR will consist of neighboring pixels



(a) (b)

(c) (d)

Figure 18: *Typical frames from the Tea Box sequence: (a) Frame 2, (b) Frame 25, (c) Frame 54, (d) Frame 79.*

that might not have consistent depth values. This MVR will then yield synthesized views with objectionable artifacts, *e.g.* straight edges in the reference frame transform to jagged ones or objects with smoothly varying depth appear quite distorted.

We propose the following construction algorithm to ensure a consistent multivalued representation and to mitigate rendering artifacts. The steps needed to build a multivalued representation from a set of depth maps are shown in Figure 21 and described in detail in the following sections. Section 4.1.1 describes warping the depth information into the reference frame. Section 4.1.2 discusses refinements of the projected depths along edge segments. Finally, Section 4.1.3 computes the final MVR depth and intensity information using a dynamic programming algorithm to enforce spatial coherence.

4.1.1 Depth warping and clustering

We propose the following steps to construct the MVR with respect to a particular reference frame. The depth maps are first projected to this reference coordinate system. Equations (3) and (4) provide the necessary 3-D transformations from local depth maps into the reference frame. While

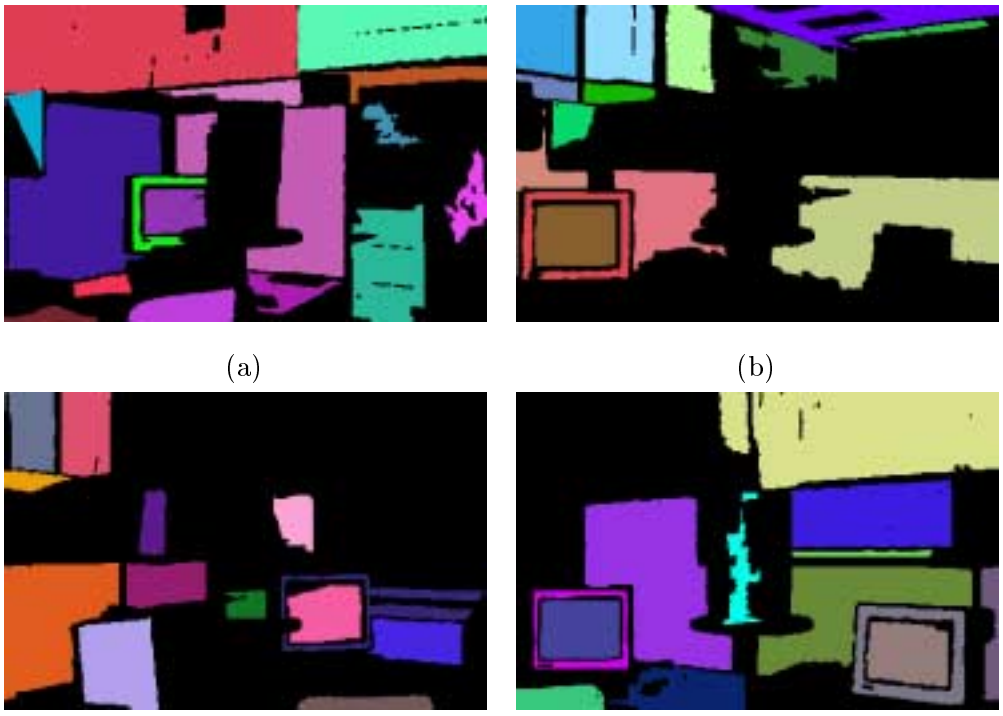


Figure 19: *Low-contrast segmentation and tracking results for Tea Box sequence: (a) Frame 2, (b) Frame 25, (c) Frame 54, (d) Frame 79.*

every frame in the sequence may be used, frames that are too far away from the reference frame are ignored; these frames will lead to projective distortion and will also require a lot more processing time. Usually, the ten-to-twenty frames nearest the reference coordinate system are sufficient. Every depth projection is assigned to the nearest pixel location in the final MVR. Each pixel in the MVR then has a collection of projected depths which is individually clustered and sorted through a k-means algorithm. The number of levels in the MVR is determined by examining the clusters in the MVR and ensuring that each level is reasonably sized and connected.

A five-frame example is shown in Figure 22 consisting of the depth maps for two rectangular objects obtained by a simple horizontal translation. The depth of the foreground object is shown in white while the background object is drawn in grey. Suppose the middle frame is the reference frame and the one for which we will construct a MVR. As described above, the depths in every frame are projected into this reference frame. Consider the pixel location (u, v) in the reference frame, labeled C in the diagram. This location corresponds to the upper right corner of the foreground object as well as to an occluded point on the background object. The corresponding points in the other four frames are A, A', B, B', D, and E. Assuming the depth maps are reasonably accurate,

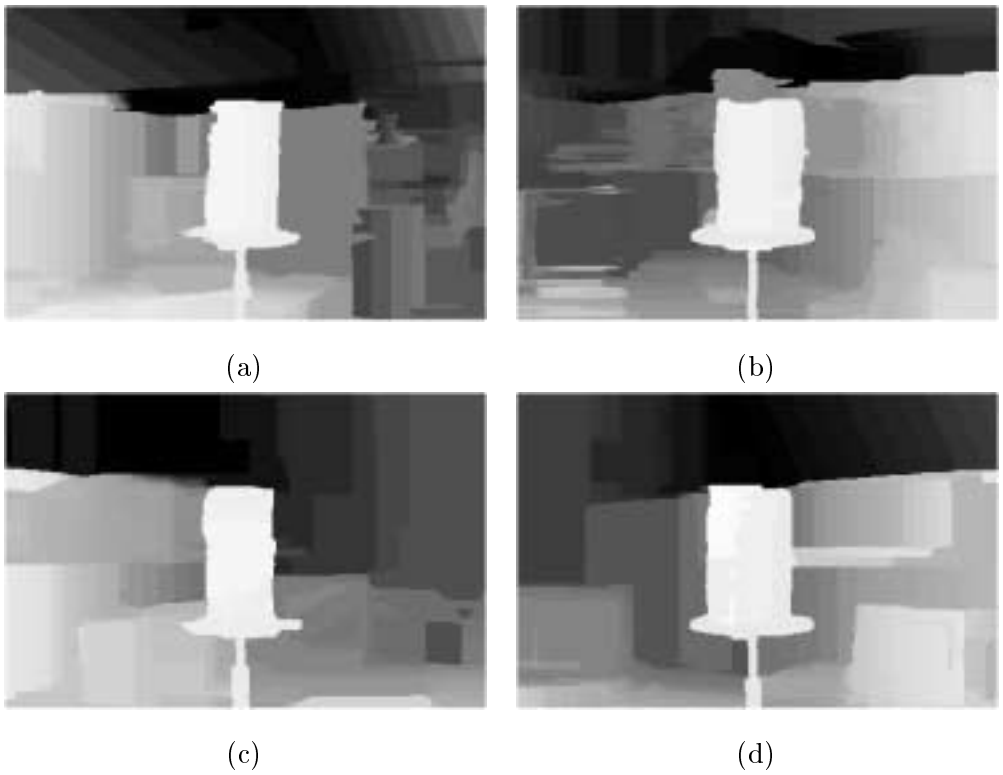


Figure 20: *Dense depth maps computed for the Tea Box sequence: (a) Frame 2, (b) Frame 25, (c) Frame 54, (d) Frame 79.*

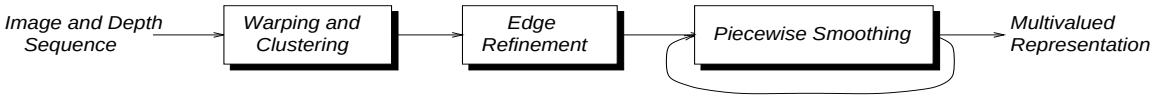


Figure 21: *Block diagram of the entire multivalued representation construction procedure.*

these points should all warp to (u, v) in the reference frame. Thus, the list of projected points associated with (u, v) is the set $\{A, A', B, B', C, D, E\}$. Through clustering, this list is separated into two groups: $\{A, B, C, D, E\}$ for the foreground point and $\{A', B'\}$ for the background point. One proceeds in a similar fashion to obtain a clustered list of depths for every MVR pixel location.

4.1.2 Edge refinement

Since many scenes consist of man-made objects with well-defined straight edges, it is important to refine the depth estimates along the more prominent edges in the representation. If the depths associated with these edges are not linear, then they will result in distorted edges which are very striking to the viewer. Of course, it is very hard to localize these potentially troublesome edges before actually forming the representation.

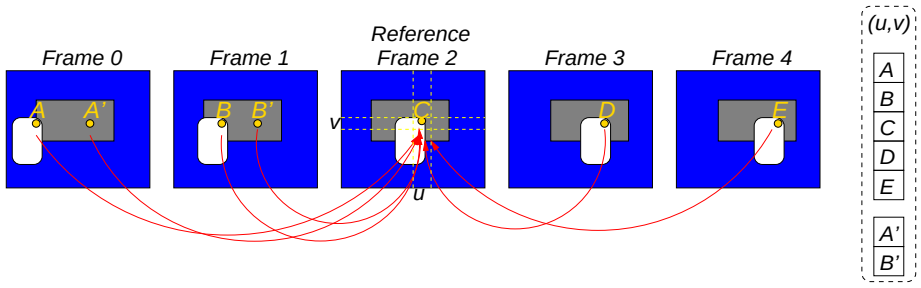


Figure 22: *Example of constructing MVR at pixel* (u, v) *from five depth maps.*

The original intensity data set serves as the best starting point to locate the edges. First, an intensity-based edge detection scheme using the derivative of the 2-D Gaussian function is applied to every frame in the data set [Lim, 1990]. The connected straight edge segments in the scene of sufficient length are retained and modeled as ideal edges. The component depth maps are refined along parallel bands close to each ideal edge using a simple least-squares technique [Chang and Zakhor, 1999]. These ideal edges are then warped into the MVR coordinate system and all corresponding edges from different frames are merged together. The depth results along parallel bands close to the merged edges in the MVR are again refitted using least squares. The final result are much straighter edges in the scene and in the synthesized views.

4.1.3 Piecewise smoothing

Even though the component depth maps are piecewise smooth, there is no guarantee that they will remain that way after projection onto the reference frame. An iterative dynamic programming algorithm similar to the one described in Section 3.2.2 is thus performed on every cluster to ensure piecewise smooth levels. In this case, the component cost $C(u, v, d)$, again omitting the arguments, is written as

$$C \quad = \quad J + C_c|\Delta\zeta| \tag{19}$$

Recall that J is the median intensity error as described before, C_c is a penalty factor, and $\Delta\zeta$ is the absolute difference between the candidate inverse depth and the previous pixel's inverse depth. This dynamic program finds the minimum solution paths through the frontmost cluster of every pixel. The result forms the depth surface for Level 0 of the MVR. These clusters are then removed from future consideration and the process continues until the depth surface associated with all the levels has been determined. Since only a small number of frames contribute to the MVR, there are fewer inverse depth candidates per cluster and hence the required search time for this DP

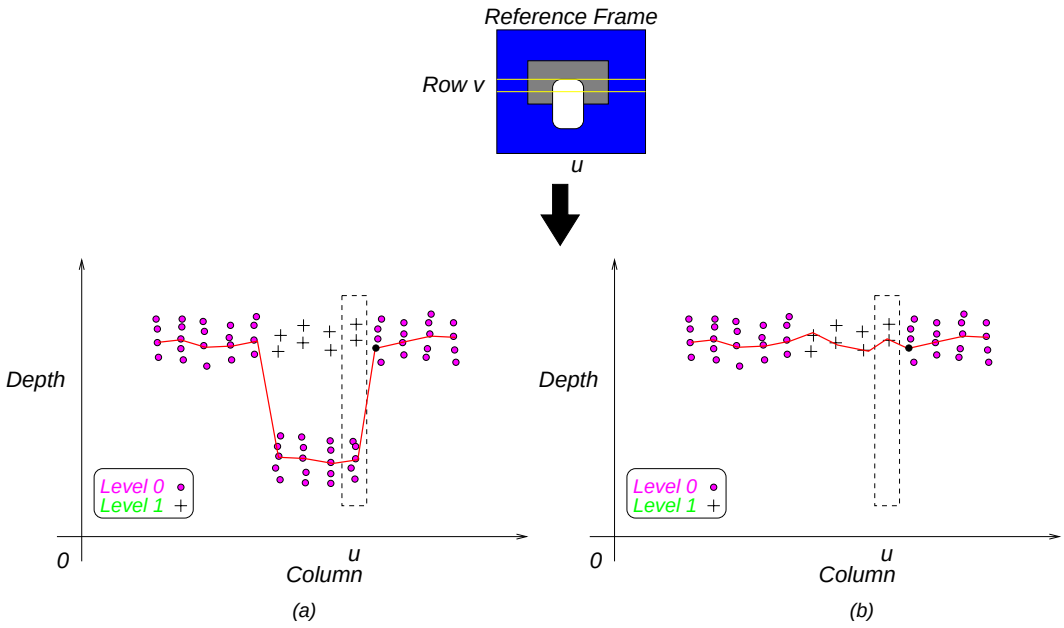


Figure 23: *Example of using dynamic programming for piecewise smoothing.*

algorithm is much shorter than compared to the previous DP algorithm. Each path is also made to pass through the above edge-based refinements. This step ensures that straight lines in the original image sequence result in straight lines in the depth domain, thus minimizing perceptual distortion in the reconstructed views.

Figure 23 illustrates smoothing on the two-plane case from before. The plots represent the projected depths associated with every column in row v; notice the two distinct clusters of depth corresponding to the two planar objects. The circular points comprise the candidate Level 0 depth surface while the cross points form the candidate Level 1 depth surface. The DP algorithm finds the minimum solution path through the frontmost cluster in every column, resulting in the path shown in Figure 23 (a) corresponding to the depth surface of Level 0. The frontmost cluster of every column with multiple clusters is removed and the DP algorithm is run again, this time resulting in the path shown in Figure 23 (b) which is equivalent to a mixture of Level 0 and 1 depth surfaces.

The piecewise smoothing step is applied in both horizontal and vertical directions to improve coherence in the final MVR. It is repeated for a few iterations until convergence to produce the depth for the MVR. The intensity information is found by inverse warping the depth information back into the original frames and using the median of the intensities. The final result consists of a single multivalued array of intensities and depths corresponding to the reference frame.

4.2 Synthesis Issues

Once the multivalued representation has been obtained, it may not be entirely clear how to reconstruct the original images or synthesize new views of the scene. One problem is that there are multiple levels in the representation of differing sizes. A solution is to consider rendering slices of the representation. A *slice* is a dense single-valued array with the same dimensions as the overall MVR obtained by combining points from different levels. In particular, slice k consists of all the points from Level k, as well as points from Level $k-1$ for pixels with no defined values in Level k, and so forth, until a dense array is formed. Note that slice 0 is precisely Level 0 of the representation. One simply needs to render all slices to render a new view.

Rendering a particular slice is relatively straightforward. Each slice is regarded as a deformable mesh of quadrilateral patches, and the appropriate 3-D transformation is applied to the vertices. The results are then rendered using standard computer graphics scanline algorithms with z-buffering. Patches which are partially occluded or become backfacing are discarded. A segmentation algorithm is used to identify patches which transcend depth boundaries. These patches are also ignored to minimize smearing artifacts in the synthesized view [Chang and Zakhor, 1997b]. Small cracks which may arise due to slight inconsistencies in the projections are filled in using a simple interpolation algorithm. It is possible for larger holes to remain in the final view; they occur because there is insufficient information in the corresponding region in the MVR.

In this manner, one can easily produce a virtual flythrough of the captured scene from a single MVR. However, only the front sides of objects can be adequately recovered since the MVR, as described so far, is a 2.5-D representation similar to other layered representations; backfacing views of the scene cannot be synthesized from a single MVR. Another artifact of such representations is their inability to represent planes which intersect the camera origin.

To address this problem, we propose using multiple MVRs at distinct viewpoints around the scene in a view interpolation scheme. Assuming the 3-D transformations among all the viewpoints are known, one can simply produce view estimates from each of the MVRs and then combine the results to form the desired view. Only the MVRs close in viewing angle to the desired view are considered and the contributions from these MVRs are weighted based on the difference in viewing angle. With this approach, even if backfacing views of the scene cannot be synthesized from a given MVR, other MVRs in the set can be used to provide the required view, thus enabling a complete virtual flyaround.

This rendering scheme is simple to implement and is a natural extension of standard graphics

(a) (b)

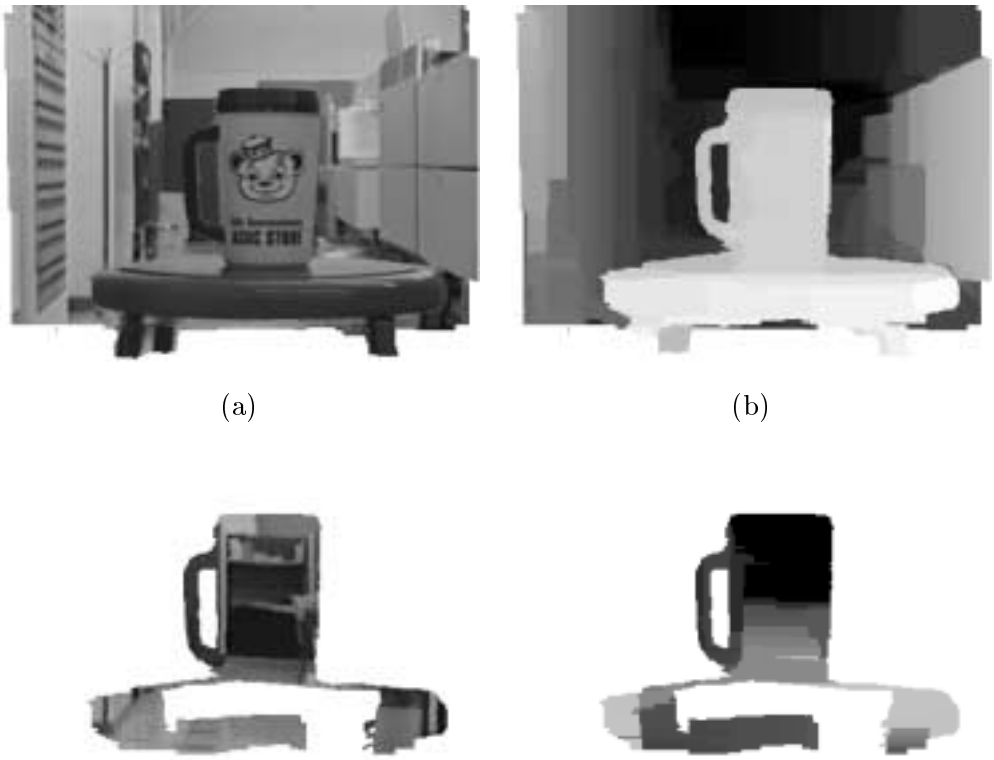
Figure 24: *Example of reconstructing Frame 16 of the Tea Box sequence from (a) a single MVR with respect to Frame 22 and (b) multiple MVRs.*

algorithms. We believe rendering multiple slices is acceptable for MVRs because many real-world data sets will consist of at most three levels. The scheme produces reasonable results but is by no means the fastest approach; the unoptimized software renderer used in this paper generates frames every five seconds. Immediate speed improvements can occur by optimizing the rendering software and by utilizing hardware acceleration. For even faster rates, such as for truly interactive flythrough applications, it may be better to use more efficient ordering and rendering algorithms like [McMillan, 1995; Shade *et al.*, 1998].

Figures 24 (a) and (b) compare reconstructed views generated from a single MVR and multiple MVRs. Suppose we consider a single MVR defined with respect to Frame 22 of the Tea Box sequence; the details of this representation are shown in Section 4.3. Using only this single MVR to reconstruct Frame 16 produces the result in Figure 24 (a). The overall quality of the view is quite good, however a striking artifact is that the the tea box is missing its right side. The artifact arises becauses the single MVR is incapable of representing this type of plane. Suppose instead that multiple MVRs from the Tea Box sequence are used. In this case, the reconstructed Frame 16 shown in Figure 24 (b) leads to a complete tea box with both of its sides intact.

4.3 Results

To verify its effectiveness, the MVR algorithm is applied to the sequences from the previous section. For the Mug sequence, the center frame (Frame 4) is selected as the primary reference frame for the representation. The algorithm automatically determines that only two levels are needed for this scene. The intensity and depth information in Level 0 are shown in Figures 25 (a) and (b) respectively. Points shown in white correspond to regions without intensity and depth. The shape



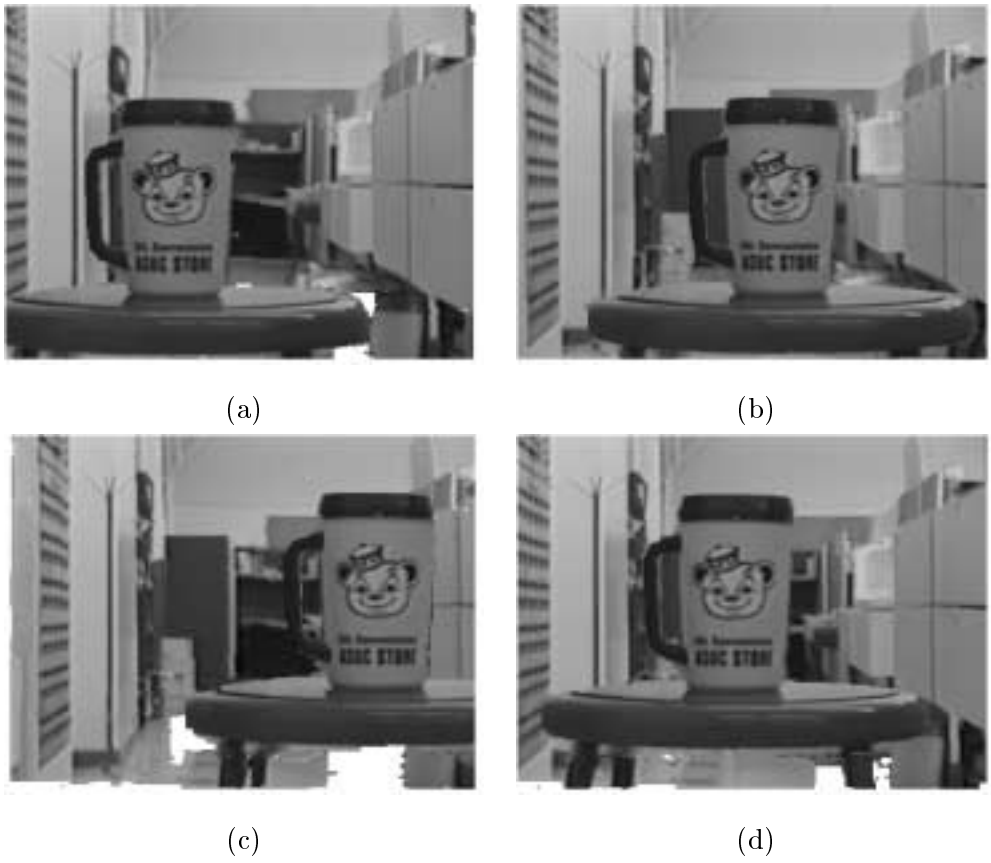
(a) (b)

(c) (d)

Figure 25: *Recovered information for Levels 0 and 1 of the Mug MVR: (a) intensity and (b) depth maps for Level 0; (c) intensity and (d) depth maps for Level 1.*

(a) (b)

Figure 26: *Points from Level 1 of the Mug MVR are combined with points from Level 0 to put the representation in context: (a) intensity and (b) depth.*



(a) (b)

(c) (d)

Figure 27: *Examples of reconstructed views using the Mug MVR: (a) Frame 0, (b) Frame 4, (c) Frame 9, (d) Frame 11.*

of the mug and the stool have been recovered quite well. Notice that the left and right sides of the scene descend in depth as expected. Also, the dimensions of the original image in Figure 14 have been expanded automatically and the points seen along the borders have been recovered and added on. The legs of the stool have even been extended by the algorithm.

Figures 25 (c) and (d) show the recovered information in Level 1 of the MVR. All of the information corresponds to points that are located behind the mug and stool. The cubicle and the wall are both recovered from behind the mug since they are seen in some of the original images. Moreover, most of the ground obscured by the stool is revealed in this level. The white region in the middle appears because these points were not visible, and thus not captured in the original image set. By filling in points from Level 0 for the pixels with no defined values in Level 1, as in Figures 26 (a) and (b), it appears that the mug and most of the stool have been removed. Notice that the bottom portion of the legs and part of the stool remain because the regions behind them were occluded in the original images.

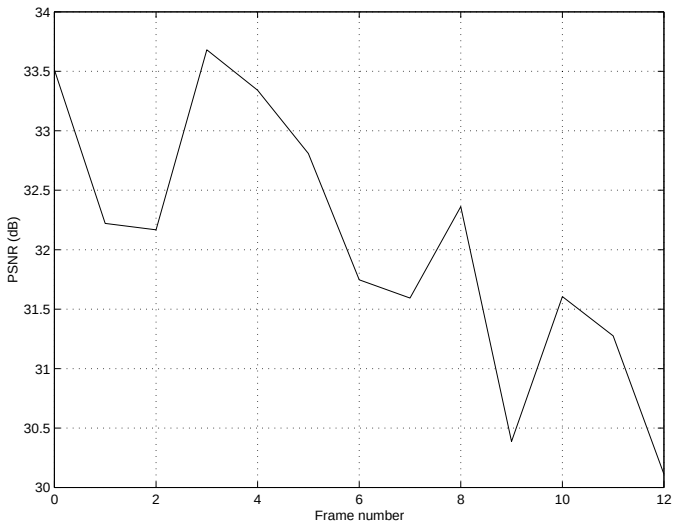


Figure 28: *Graph of PSNRs for reconstructed images using the Mug MVR.*

It is worth noting that typical LR and LDI approaches would create more than two layers for this scene since the objects cannot be separated into just two planar layers. A separate layer is required for the mug, the stool, the left wall, the background, the drawers to the right, and the ground. While a large number of layers in itself is not problematic, there is the potential for the borders between layers not to line up, thus resulting in gaps in the final synthesized views.

It is difficult to quantify the accuracy of the Mug MVR since we are working with a real-world scene without actual depth measurements. A reasonable measure of the approach is the quality of reconstructed images. The reconstruction techniques of Section 4 are applied to generate the original images. As an example, Frames 0, 4, 9, and 11 have been reconstructed in Figures 27 (a)–(d). Notice that the reconstructed quality is quite good. Figure 28 shows the PSNR of the entire sequence; the average PSNR of all of the reconstructed images is 32.063 dB. Notice that the PSNR for the higher elevation frames, Frames 9 to 12, are lower than the other frames because of the larger motions involved.

Synthesized views of the scene not originally captured by the camcorder may be generated in a similar manner. Figure 29 shows four novel views of the scene obtained by a virtual camera undergoing arbitrary motion. The resulting images are reasonable and provide a sense of depth.

The white regions in the above synthesized images occur because the representation has no information about them. One way to minimize these regions is to consider a motion trajectory such as orbital motion. Eight MVRs are constructed from different portions of the Tea Box sequence, each with a maximum angular range of approximately 45 degrees. For space reasons, only two of

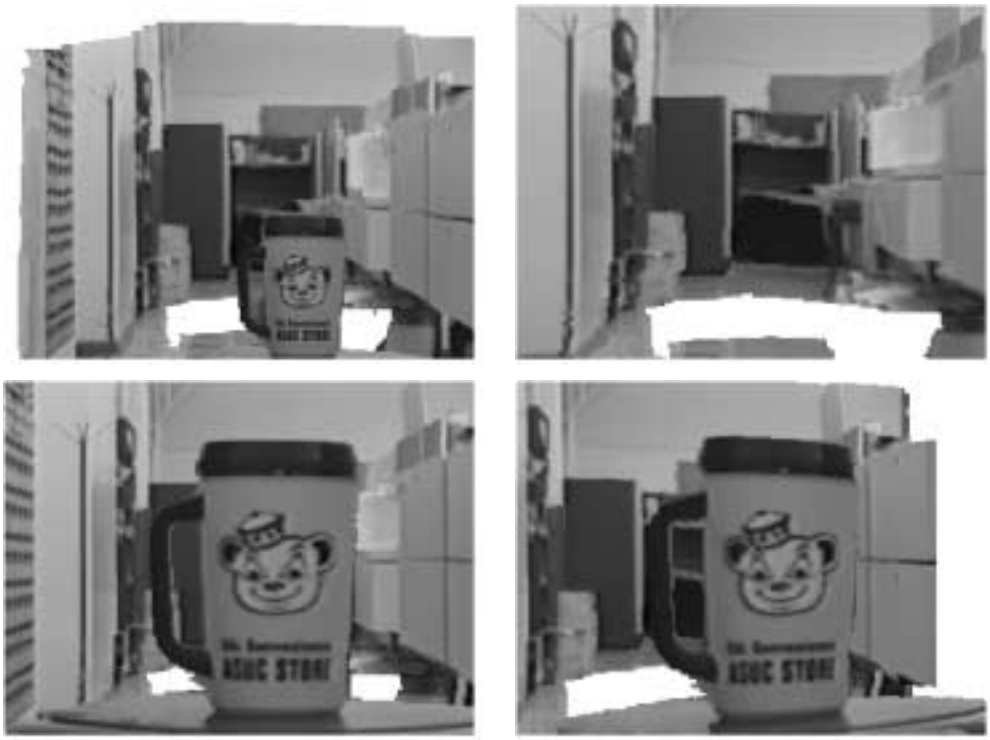


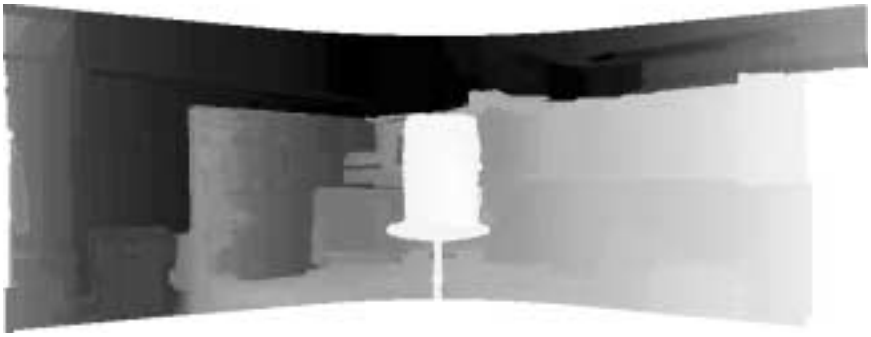
Figure 29: *Examples of synthesized views using the Mug MVR.*

the eight MVRs shall be presented in this paper; the remaining six MVRs are similar.

The reference frame for the first MVR is Frame 22 and its nearest subsequence of neighboring frames is selected. Integrating the information captured in any one of the subsequences, one obtains a much higher resolution view coinciding with the reference viewpoint. Figures 30 (a)–(d) show the intensity and depth information for Levels 0 and 1 computed by the proposed algorithm with respect to the reference frame. The occluded lamp and wall behind the tea box are recovered in Level 1. More interestingly, the two streaks above and to the right of the tea box are actually correct since they correspond to the regions occluded by the envelope sticking up and the bookshelf partition in Figure 30 (a). Notice that the algorithm results in a 3-D corrected mosaic of the input image subsequence. Figures 31 (a) and (b) show the effect of combining the two levels; it appears as though the entire tea box has been removed from the scene.

A second MVR defined with respect to Frame 86 is shown in Figures 32 (a)–(d). As with the previous MVR, the algorithm determines only two levels are needed to represent this subsequence. The proposed algorithm recovers the information occluded by the tea box. The extra horizontal strip above the tea box corresponds to part of the background wall that is visible over the cubicle wall in some of the original frames. Figures 33 (a) and (b) combine the points from Level 1 with

(a)

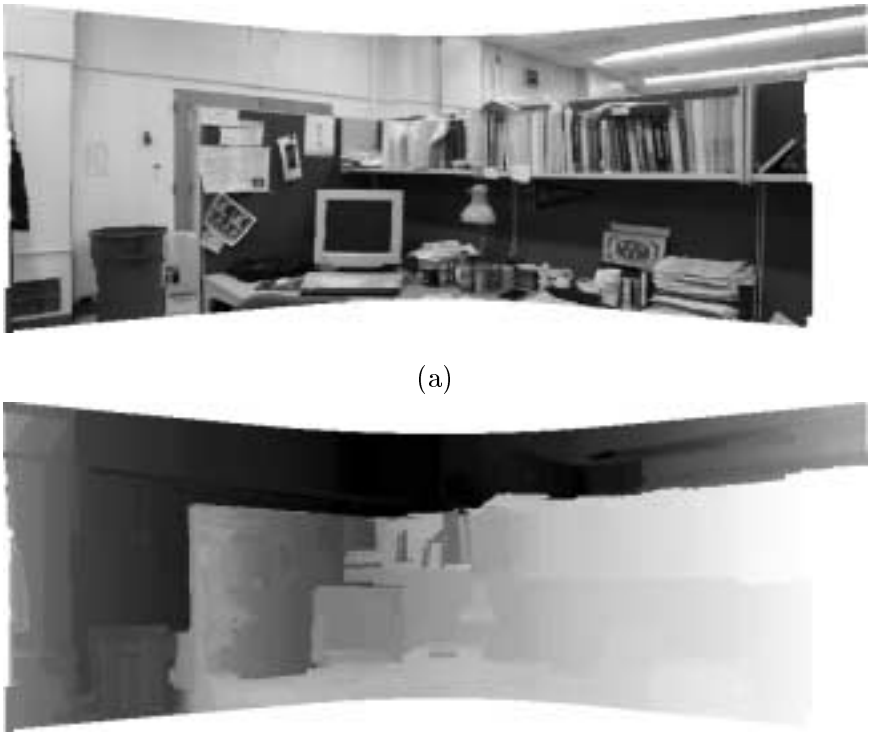


(b)

(c)

(d)

Figure 30: *Recovered information for the Tea Box MVR with respect to Frame 22: (a) intensity and (b) depth maps for Level 0; and (c) intensity and (d) depth maps for Level 1.*



(a)

(b)

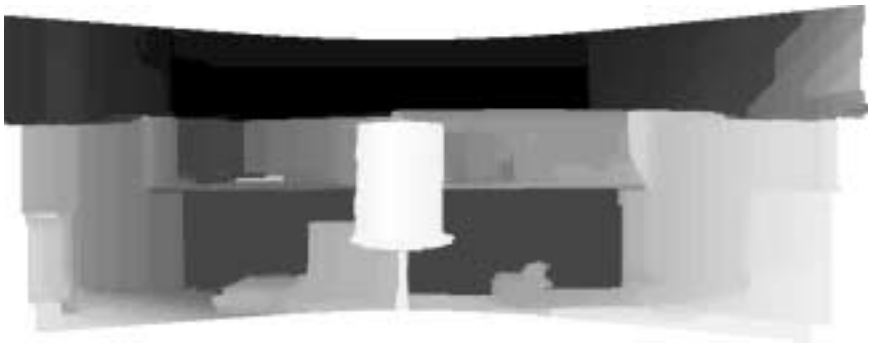
Figure 31: *Points from Level 1 of the Tea Box MVR with respect to Frame 22 are combined with points from Level 0 to put the representation in context: (a) intensity and (b) depth.*

those in Level 0 to look as if the tea box and apparatus have disappeared.

As discussed earlier, any one of the Tea Box MVRs can be used to synthesize views locally around the MVR's reference frame. However, it is more interesting to interpolate views between multiple MVRs to construct a complete virtual flyaround. The multiple MVRs may be used to accurately reconstruct the input images; Figures 34 (a)–(d) are reconstructed examples corresponding to Figures 18 (a)–(d). Figure 35 shows the graph of PSNRs for all reconstructed images using the multiple MVRs simultaneously for reconstruction. Notice that the PSNRs approach a local maxima around each of the eight reference frames. The average PSNR of 24.762 dB is somewhat lower than that of the Mug sequence due to the considerably large motions involved.

While reconstructed images are important, the power of the multivalued representation is in synthesizing new views of the scene. Figure 36 shows a set of novel views not restricted to the original orbital path. These views are constructed using multiple MVRs where the contributions of each MVR is weighted by its proximity to the viewpoint. Notice the dramatically different viewpoints that can be synthesized. A more impressive effect is obtained by viewing the synthesized

(a)

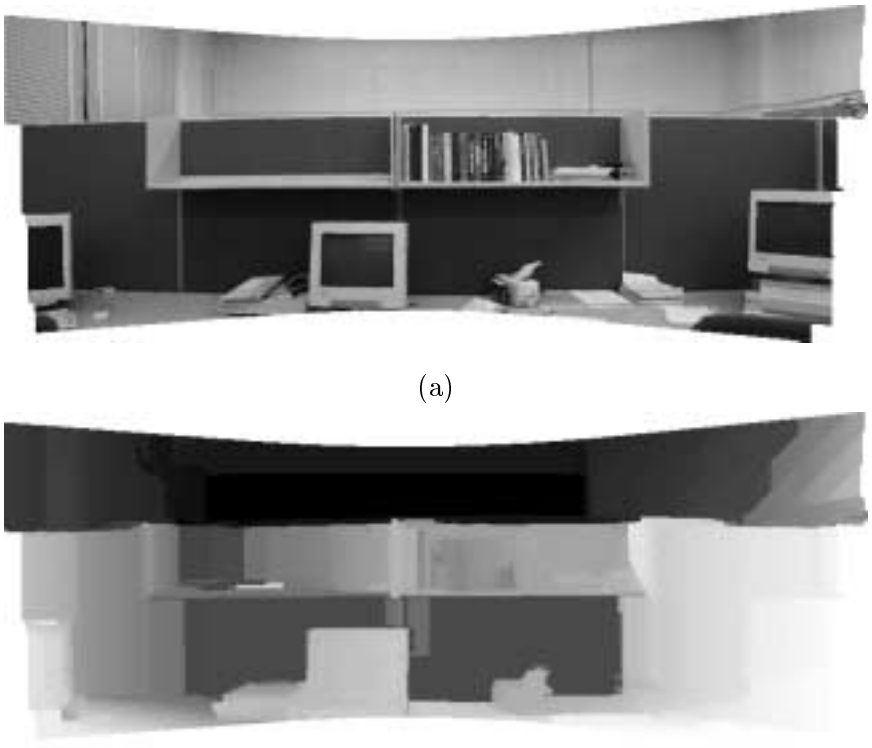


(b)

(c)

(d)

Figure 32: *Recovered information for the Tea Box MVR with respect to Frame 86: (a) intensity and (b) depth maps for Level 0; and (c) intensity and (d) depth maps for Level 1.*



(a)

(b)

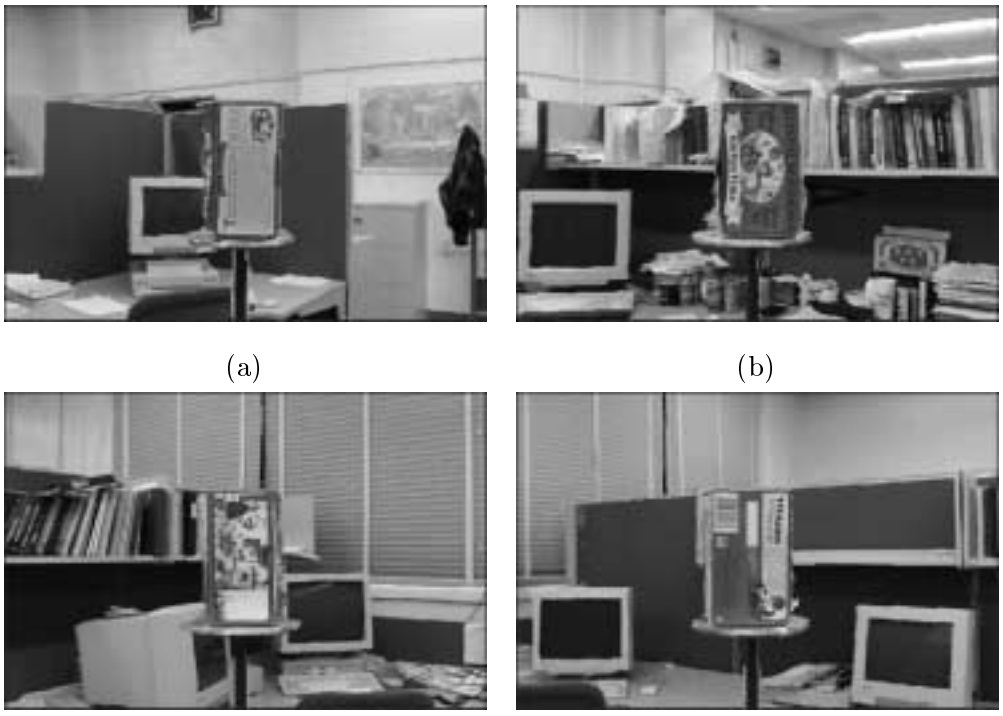
Figure 33: *Points from Level 1 of the Tea Box MVR with respect to Frame 86 are combined with points from Level 0 to put the representation in context: (a) intensity and (b) depth.*

views as an animated sequence.

5 Summary and Conclusions

We have proposed a multivalued representation to address the problem of compact representation for image reconstruction and new view synthesis. Instead of grouping by coherent affine motions, MVR organizes the input data into levels of occlusions. The MVR is automatically constructed with respect to a single reference frame from a set of dense depth maps. A multiframe depth estimation algorithm has been shown to be effective in incorporating all frames simultaneously to estimate depth, even in the traditionally difficult low-contrast regions. Since MVR has similar traits to layered representations, it accumulates information seen in the union of frames as well as minimizes the redundancy of the overall representation. Since MVR is primarily a depth-based representation, it is capable of overcoming problems of occlusion during synthesis.

It is worth noting some key differences between MVR and LR. First, LR consists of boundary,



(a) (b)

(c) (d)

Figure 34: *Examples of reconstructed views using the Tea Box MVRs: (a) Frame 2, (b) Frame 25, (c) Frame 54, (d) Frame 79.*

intensity, and affine motion information for each layer whereas MVR consists of boundary, intensity, and depth information for each level. Because dense depth is estimated for every frame, the MVR framework is much more complex than its LR counterpart. However, the tradeoff is that the depth information allows MVR to synthesize new views more naturally than LR. If however the scene consists of only planar objects, there is a direct relationship between MVR levels and layers. Also, LR cannot be used in situations where each object in the scene cannot be mapped to a 2-D affine motion; MVR does not have this limitation.

This paper demonstrated the effectiveness of the representation for two types of constrained camera motion. It must be pointed out however that the proposed system is certainly not limited to these types of motion. In fact, it may be extended directly to arbitrary motions, provided accurate motion estimates can be obtained for every frame. The results throughout the paper suggest that the MVR provides a powerful extension to typical depth-based representations and related applications.

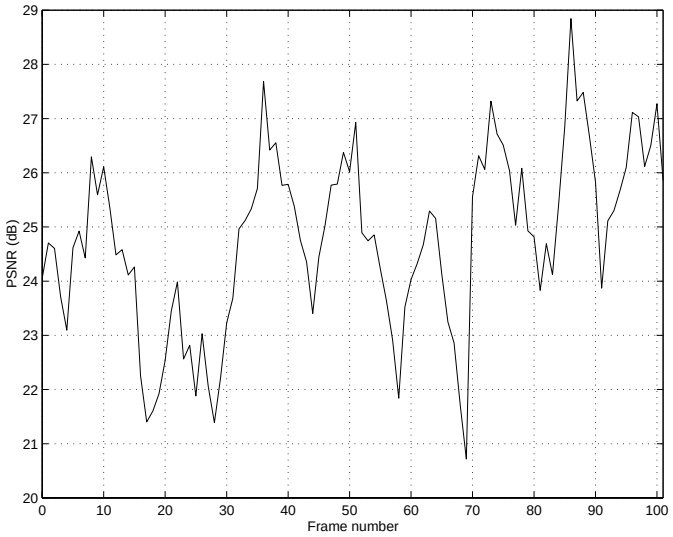


Figure 35: *Graph of PSNRs for reconstructed images using the Tea Box MVRs.*

References

[Anandan *et al.*, 1993] P. Anandan, J. R. Bergen, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In M. I. Sezan and R. L. Lagendijk, editor, *Motion Analysis and Image Sequence Processing*, chapter 1. Kluwer Academic Publishers, 1993.

[Anandan, 1984] P. Anandan. Computing dense displacement fields with confidence measures in scenes containing occlusion. In *Proceedings of the SPIE: Intelligent Robots and Computer Vision*, volume 521, pages 184–194. Cambridge, MA, 5–8 Nov. 1984.

[Baker *et al.*, 1998] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Proceedings of CVPR*, pages 434–441. Santa Barbara, CA, June 1998.

[Chang and Zakhor, 1997a] N. L. Chang and A. Zakhor. Multivalued representations for image reconstruction and new view synthesis. Qualifying Examination Proposal, University of California at Berkeley, January 1997. Also Technical Report, Video and Image Processing Lab, Feb. 1997.

[Chang and Zakhor, 1997b] N. L. Chang and A. Zakhor. View generation for three-dimensional scenes from video sequences. *IEEE Trans. on Image Proc.*, 6(4):584–598, Apr. 1997.

[Chang and Zakhor, 1998] N. L. Chang and A. Zakhor. Finite sensor effects for estimating structure-from-motion. In *Proceedings of ICIP*, volume 1, pages 918–922. Chicago, IL, 5–8 Oct. 1998.

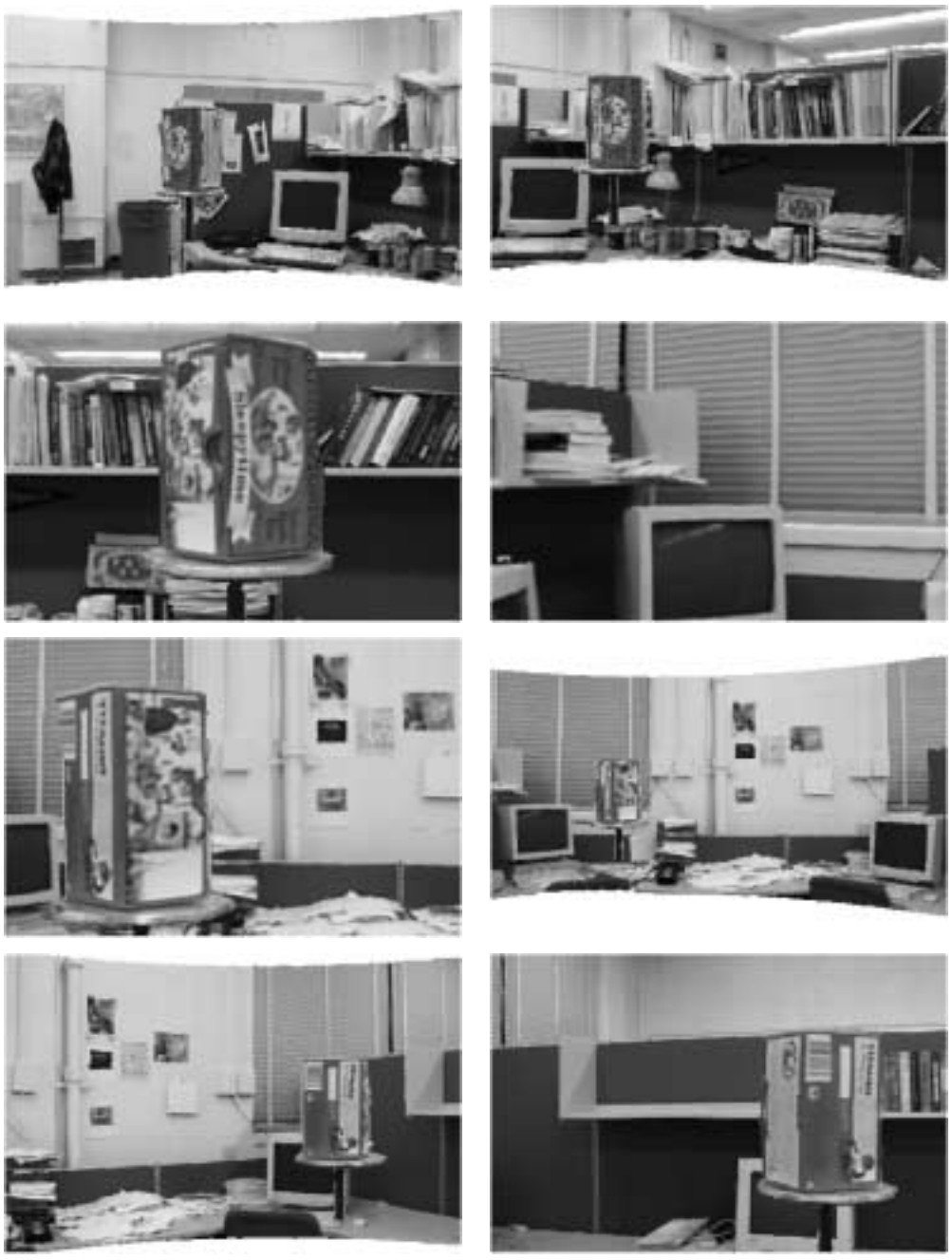


Figure 36: *Examples of synthesized views using the Tea Box MVRs.*

[Chang and Zakhor, 1999] N. L. Chang and A. Zakhor. A multivalued representation for view synthesis. In *Proceedings of ICIP (Invited paper)*, volume 2, pages 505–509. Kobe, Japan, 25–28 Oct. 1999.

[Chang, 1994] N. L. Chang. View reconstruction from uncalibrated cameras for three-dimensional scenes. Master's thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 1994.

[Chang, 1999] N. L. Chang. *Depth-Based Representations of Three-Dimensional Scenes for View Synthesis.* PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 1999. URL: `www-video.eecs.berkeley.edu/~nlachang/MVR`.

[Chen and Williams, 1993] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of SIGGRAPH*, pages 279–288. New York, NY, 1–6 Aug. 1993.

[Cox *et al.*, 1992] I. J. Cox, S. Hingorani, B. M. Maggs, and S. B. Rao. Stereo without disparity gradient smoothing: a bayesian sensor fusion solution. In *Proceedings of BMVC*, pages 337–346. Leeds, UK, 22–24 Sept. 1992.

[Darrell and Pentland, 1995] T. Darrell and A. P. Pentland. Cooperative robust estimation using layers of support. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 17(5):474–487, May 1995.

[Debevec, 1996] P. E. Debevec. *Modeling and Rendering Architecture from Photographs.* PhD thesis, Computer Sciences Division, University of California at Berkeley, 1996.

[Falkenhagen, 1994] L. Falkenhagen. Depth estimation from stereoscopic image pairs assuming piecewise continuous surfaces. In *Workshops in Computing, Image Processing for Broadcast and Video Production*, pages 115–127. Hamburg, 1994.

[Faugeras, 1994] O. D. Faugeras. *Three-Dimensional Computer Vision.* MIT Press, Cambridge, MA, 1994.

[Fua, 1993] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1):35–49, Winter 1993.

[Gortler *et al.*, 1996] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH*, pages 43–54. New Orleans, LA, 4–9 Aug. 1996.

[Haralick and Shapiro, 1985] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, Jan. 1985.

[Hartley, 1997] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 19(6):580–93, June 1997.

[Intille and Bobick, 1994] S. S. Intille and A. F. Bobick. Disparity-space images and large occlusion stereo. Technical Report 220, MIT Media Lab Perceptual Computing Group, May 1994.

[Kanade *et al.*, 1997] T. Kanade, P. W. Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, 4(1):34–47, Jan-Mar. 1997.

[Kang and Szeliski, 1997] S. B. Kang and R. Szeliski. 3-d scene data recovery using omnidirectional multibaseline stereo. *International Journal of Computer Vision*, 25(2):167–183, 1997.

[Koch, 1993] R. Koch. Automatic reconstruction of buildings from stereoscopic image sequences. In *Proceedings of EUROGRAPHICS*, pages 339–350. Barcelona, Spain, 6–10 Sept. 1993.

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

[Levoy and Hanrahan, 1996] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH*, pages 31–42. New Orleans, LA, 4–9 Aug. 1996.

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

[Longuet-Higgins, 1981] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 10–16 Sept. 1981.

[Matthies *et al.*, 1989] L. Matthies, T. Kanade, and R. Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–238, 1989.

[Maybank, 1993] S. Maybank. *Theory of Reconstruction from Image Motion.* Spring-Verlag, Berlin, 1993.

[McMillan and Bishop, 1995] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of SIGGRAPH*, pages 39–46. Los Angeles, CA, 6–11 Aug. 1995.

[McMillan, 1995] L. McMillan. A list-priority rendering algorithm for redisplaying projected surfaces. Technical Report 95-005, University of North Carolina, 1995.

[Meier and Ngan, 1998] T. Meier and K. N. Ngan. Automatic segmentation of moving objects for video object plane generation. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):525–538, Sept. 1998.

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

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

[Okutomi and Kanade, 1993] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 15(4):353–363, Apr. 1993.

[Pal and Pal, 1993] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, Sept. 1993.

[Rousseeuw and Leroy, 1987] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, New York, 1987.

[Sawhney and Ayer, 1996] H. S. Sawhney and S. Ayer. Compact representations of videos through dominant and multiple motion estimation. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 18(8):814–830, Aug. 1996.

[Seitz and Dyer, 1996] S. M. Seitz and C. R. Dyer. View morphing. In *Proceedings of SIGGRAPH*, pages 21–30. New Orleans, LA, 4–9 Aug. 1996.

[Shade *et al.*, 1998] J. Shade, S. Gortler, L. W. He, and R. Szeliski. Layered depth images. In *Proceedings of SIGGRAPH*. Orlando, FL, July 1998.

[Shi *et al.*, 1998] J. Shi, S. Belongie, T. Leung, and J. Malik. Image and video segmentation: The normalized cut framework. In *Proceedings of ICIP*, volume 1, pages 943–947. Chicago, IL, 4–7 Oct. 1998.

[Shum *et al.*, 1995] H. Y. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(9):854–867, 1995.

[Shum *et al.*, 1998] H. Y. Shum, M. Han, and R. Szeliski. Interactive construction of 3-d models from panoramic mosaics. In *Proceedings of CVPR*, pages 427–433. Santa Barbara, CA, 23–25 June 1998.

[Tomasi and Kanade, 1992] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization. *International Journal of Computer Vision*, 9(2):137–154, Nov. 1992.

[Tsai and Huang, 1984] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Trans. on Patt. Anal. Mach. Intell.*, PAMI–6(1):13–27, Jan. 1984.

[Tsai, 1987] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3-d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA–3(4):323–344, Aug. 1987.

[Vass *et al.*, 1998] J. Vass, K. Palaniappan, and X. Zhuang. Automatic spatio-temporal video sequence segmentation. In *Proceedings of ICIP*, volume 1, pages 958–962. Chicago, IL, 4–7 Oct. 1998.

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

[Wang, 1998] D. Wang. Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):539–546, Sept. 1998.

[Weiss and Adelson, 1996] Y. Weiss and E. H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proceedings of CVPR*, pages 321–326. San Francisco, CA, 18–20 June 1996.

[Zhang *et al.*, 1995] Z. Zhang, R. Deriche, O. D. Faugeras, and Q. T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1–2):87–119, Oct. 1995.

Constructing a Multivalued Representation for View Synthesis

Nelson L. Chang
Imaging Technology Department
Hewlett-Packard Laboratories
1501 Page Mill Road, MS 4U-6
Palo Alto, CA 94304 USA
email: nlachang@hpl.hp.com

Avideh Zakhor
EECS Department
University of California
Berkeley, CA 94720 USA
email: avz@eecs.Berkeley.EDU

Footnotes

1. An alternate view is that the scene is captured by a moving camera at M discrete locations.

2. The matrix $AR^{st}A^{-1}$ in Equation (2) has been labeled R out of convenience. Even though it reflects the original 3-D rotation with calibration effects, R is obviously not a true rotation matrix since it is not orthogonal.

3. The indices s and t are assumed to be a part of r_k, Δx, Δy, and Δz. They have been omitted in the equations for clarity. The indices will be used later in the text when the parameters' explicit dependence on s and t needs to be emphasized.

4. If the image sequence is obtained by a camera undergoing a closed motion path, the sequence wraps around itself, *i.e.* the first frame has the last frame as its previous frame and the last frame has the first frame as its subsequent frame. For nonclosed paths, there is no previous frame for the first frame and no subsequent frame for the last frame.

5. The depth maps have been quantized to 256 grey values where the depth is inversely related to the brightness. The maps have also been individually histogram equalized to improve visibility and to show the contrast between the object and the surrounding background.