Appeared in the 1998 Conference on Computer Vision and Pattern Recognition, Santa Barbara, CA, June 1998.

A Layered Approach to Stereo Reconstruction

Simon Baker *

Department of Computer Science
Columbia University
New York, NY 10027

Richard Szeliski and P. Anandan

Microsoft Research
Microsoft Corporation
Redmond, WA 98052

Abstract

We propose a framework for extracting structure from stereo which represents the scene as a collection of approximately planar layers. Each layer consists of an explicit 3D plane equation, a colored image with per-pixel opacity (a sprite), and a per-pixel depth offset relative to the plane. Initial estimates of the layers are recovered using techniques taken from parametric motion estimation. These initial estimates are then refined using a re-synthesis algorithm which takes into account both occlusions and mixed pixels. Reasoning about such effects allows the recovery of depth and color information with high accuracy, even in partially occluded regions. Another important benefit of our framework is that the output consists of a collection of approximately planar regions, a representation which is far more appropriate than a dense depth map for many applications such as rendering and video parsing.

1 Introduction

Although extracting scene structure using stereo has long been an active area of research, the recovery of accurate depth information still remains largely unsolved. Most existing algorithms work well when matching feature points or highly textured regions, but perform poorly around occlusion boundaries and in untextured regions.

A common element of many recent attempts to solve these problems is explicit modeling of the 3D volume of the scene [37, 13, 8, 25, 26, 30]. The scene volume is discretized, often in terms of equal increments of disparity, rather than into equally sized voxels. The goal is then to find the voxels which lie on the surfaces of the objects in the scene. The major benefits of such approaches include, the equal and efficient treatment of a large number of images [8], the possibility of modeling occlusions [13], and the detection of mixed pixels at occlusion boundaries to obtain sub-pixel accuracy [30]. Unfortunately, discretizing space volumetrically introduces a huge number of degrees of freedom, and leads to sampling and aliasing artifacts.

Another active area of research is the detection of parametric motions within image sequences [1, 34, 12, 9, 15, 14, 24, 5, 36, 35]. Here, the goal is the decomposition of the images into sub-images, commonly referred to as *layers,* such that the pixels within each layer move in a manner consistent with a parametric transformation. The motion of each layer is determined by the values of the parameters. An important transformation is the 8–parameter homography (collineation), because it describes the motion of a rigid planar patch as either it or the camera moves [11].

While existing techniques have been successful in detecting multiple independent motions, layer extraction for scene modeling has not been fully developed. One fact which has not been exploited is that, when simultaneously imaged by several cameras, each of the layers implicitly lies on a fixed plane in the 3D world. Another omission is the proper treatment of transparency. With a few exceptions (e.g. [27, 3, 18]), the decomposition of an image into layers that are partially transparent has not been attempted. In contrast, scene modeling using multiple partially transparent layers is common in the graphics community [22, 6].

In this paper, we present a framework for reconstructing a scene as a collection of approximately planar layers. Each of the layers has an explicit 3D plane equation and is recovered as a *sprite*, i.e. a colored image with per-pixel opacity (transparency) [22, 6, 32, 20]. To model a wider range of scenes, a per-pixel depth offset relative to the plane is also added. Recovery of the layers begins with the iteration of several steps based on techniques developed for parametric motion estimation, image registration, and mosaicing. The resulting layer estimates are then refined using a re-synthesis step which takes into account both occlusions and mixed pixels in a similar manner to [30].

Our layered approach to stereo shares many of the advantages of the aforementioned volumetric techniques. In addition, it offers a number of other advantages:

- The combination of the global model (the plane) with the local correction to it (the per-pixel depth offset) results in very robust performance. In this respect, the framework is similar to the *plane + parallax* work of [19, 23, 29], the *model-based stereo* work of [10], and the *parametric motion + residual optical flow* of [12].
- The output (a collection of approximately planar regions) is more suitable than a discrete collection of voxels for many applications, including, rendering [10] and video parsing [15, 24].

*The research described in this paper was conducted while the first author was a summer intern at Microsoft Research.

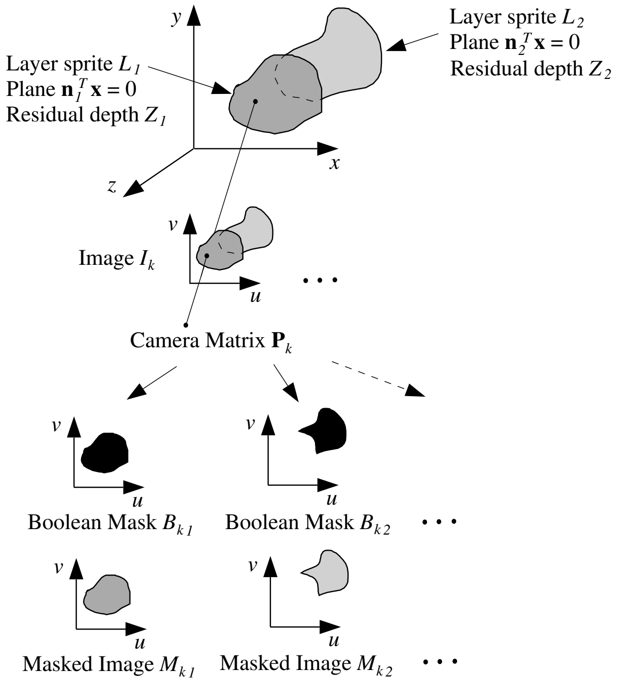

Figure 1: Suppose K images I_k are captured by K cameras $\mathbf{P}_k$. We assume that the scene can be represented by L sprite images L_l on planes $\mathbf{n}_l^T\mathbf{x} = 0$ with depth offsets Z_l. The boolean masks B_{kl} denote the pixels in image I_k from layer L_l and the masked images are given by $M_{kl} = B_{kl} \cdot I_k$.

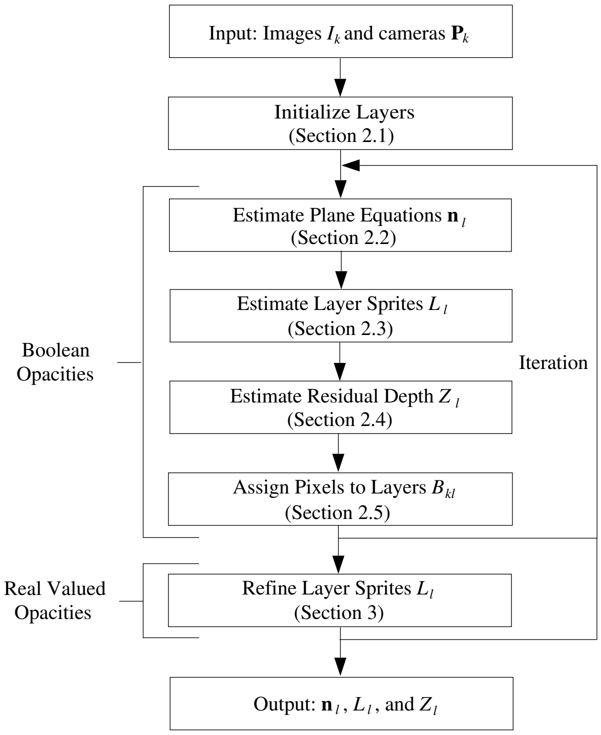

Figure 2: We wish to compute the layer sprites L_l, the layer plane vectors $\mathbf{n}_l$, the residual parallax Z_l, and the boolean mask images B_{kl}. After initializing, we iteratively compute each quantity in turn fixing the others. Finally, we refine the layer sprite estimates using a re-synthesis algorithm.

1.1 Basic Concepts and Notation

We use homogeneous coordinates for both 3D world coordinates $\mathbf{x} = (x, y, z, 1)^T$ and for 2D image coordinates $\mathbf{u} = (u, v, 1)^T$. The basic concepts of our framework are illustrated in Figure 1. We assume that the input consists of K images $I_1(\mathbf{u}_1), I_2(\mathbf{u}_2), \ldots, I_K(\mathbf{u}_K)$ captured by K cameras with known projection matrices $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_K$. (In what follows, we drop the image coordinates $\mathbf{u}_k$ unless they are needed to explain a warping operation explicitly.) We wish to reconstruct the world as a collection of L approximately planar layers. Following [6], we denote a layer sprite with *pre-multiplied opacities* by:

$$L_l(\mathbf{u}_l) \;=\; (\alpha_l \cdot r_l,\, \alpha_l \cdot g_l,\, \alpha_l \cdot b_l,\, \alpha_l) \tag{1}$$

where $r_l = r_l(\mathbf{u}_l)$ is the red band, $g_l = g_l(\mathbf{u}_l)$ is the green band, $b_l = b_l(\mathbf{u}_l)$ is the blue band, and $\alpha_l = \alpha_l(\mathbf{u}_l)$ is the opacity. We also associate a homogeneous vector $\mathbf{n}_l$ with each layer (which defines the plane equation of the layer via $\mathbf{n}_l^T\mathbf{x} = 0$) and a per-pixel residual depth offset $Z_l(\mathbf{u}_l)$.

Our goal is to estimate the layer sprites L_l, the plane vectors $\mathbf{n}_l$, and the residual depths Z_l. To do so, we wish to use techniques for parametric motion estimation. Unfortunately, most such techniques assume boolean-valued opacities α_l (i.e., unique layer assignments). We therefore split our framework into two parts. In the first part, described in Section 2, we assume boolean opacities to get a first approximation to the structure of the scene. If the opacities are boolean, each point in each image I_k is only the image of a point on one of the layers L_l. We therefore introduce boolean masks B_{kl} which denote the pixels in image I_k that are images of points on layer L_l. So, in addition to L_l, $\mathbf{n}_l$, and Z_l, we also need to estimate the masks B_{kl}. Once we have estimates of the masks, we immediately compute masked input images $M_{kl} = B_{kl} \cdot I_k$ (see Figure 1). In the second part of our framework, we use the initial estimates of the layers made by the first part as input into a re-synthesis algorithm which refines the layer sprites L_l, *including* the opacities α_l. This second step requires a generative or forward model of the image formation process and is discussed in Section 3.

In Figure 2 we illustrate the processing steps of the framework. Given any three of L_l, $\mathbf{n}_l$, Z_l, and B_{kl}, there are techniques for estimating the remaining one. The first part of our framework therefore consists of first initializing these four quantities, and then iteratively estimating each one while fixing the other three. After good initial estimates of the layers are obtained, we move on to the second part of the framework in which we use real valued opacities and refine the entire layer sprites, including the opacities.

2 Initial Computation of the Layers

2.1 Initialization of the Layers

Initialization of the layers is a difficult task, which is inevitably somewhat ad-hoc. A number of approaches have been proposed in the parametric motion literature:

- Randomly initialize a large number of small layers, which grow and merge until a small number of layers remain which accurately model the scene [34, 24, 5].
- Iteratively apply dominant motion extraction [15, 24], at each step applying the algorithm to the residual regions of the previous step.
- Perform a color segmentation in each image, match the segments, and use as the initial assignment [2].
- Apply a simple stereo algorithm to get an approximate depth map, and then fit planes to the depth map.
- Get a human to initialize the layers. (In many applications, such as model acquisition [10] and video parsing [24], the goal is a semi-automatic algorithm and limited user input is acceptable.)

In this paper, we assume a human has initialized the layers. As discussed in Section 5, fully automating the framework is left as future work.

2.2 Estimation of the Plane Equations

To compute the plane equation vector $\mathbf{n}_l$ we need to map the pixels in the masked images M_{kl} onto the plane defined by $\mathbf{n}_l^T\mathbf{x} = 0$. If $\mathbf{x}$ is a 3D world coordinate of a point and $\mathbf{u}_k$ is the image of $\mathbf{x}$ in camera $\mathbf{P}_k$, we have:

$$\mathbf{u}_k \ = \ \mathbf{P}_k\mathbf{x} \tag{2}$$

where equality is in the 2D projective space $\mathcal{P}^2$ [11]. Since $\mathbf{P}_k$ is of rank 3, it follows that:

$$\mathbf{x} \ = \ \mathbf{P}_k^*\mathbf{u}_k + s\mathbf{p}_k \tag{3}$$

where $\mathbf{P}_k^* = \mathbf{P}_k^T(\mathbf{P}_k\mathbf{P}_k^T)^{-1}$ is the pseudo-inverse of the camera matrix $\mathbf{P}_k$, s is an unknown scalar, and $\mathbf{p}_k$ is a vector in the null space of $\mathbf{P}_k$, (i.e. $\mathbf{P}_k\mathbf{p}_k = 0$). If $\mathbf{x}$ lies on the plane $\mathbf{n}_l^T\mathbf{x} = 0$ we have:

$$\mathbf{n}_l^T\mathbf{P}_k^*\mathbf{u}_k + s\mathbf{n}_l^T\mathbf{p}_k \ = \ 0. \tag{4}$$

Solving this equation for s, substituting into Equation (3), and rearranging yields:

$$\mathbf{x} \ = \ \left((\mathbf{n}_l^T\mathbf{p}_k)\mathbf{I} - \mathbf{p}_k\mathbf{n}_l^T\right)\mathbf{P}_k^*\mathbf{u}_k. \tag{5}$$

The importance of Equation (5) is that it allows us to map a pixel coordinate $\mathbf{u}_k$ in image M_{kl} onto the point on the plane $\mathbf{n}_l^T\mathbf{x} = 0$, of which it is the image. So, we can now map this point onto its image in another camera $\mathbf{P}_{k'}$:

$$\mathbf{u}_{k'} \ = \mathbf{P}_{k'}\left((\mathbf{n}_l^T\mathbf{p}_k)\mathbf{I} - \mathbf{p}_k\mathbf{n}_l^T\right)\mathbf{P}_k^*\mathbf{u}_k \ \equiv \ \mathbf{H}_{kk'}^l\mathbf{u}_k \tag{6}$$

where $\mathbf{H}_{kk'}^l$ is a homography (collineation of $\mathcal{P}^2$ [11]). Equation (6) describes the mapping between the two images which would hold if the pixels were all images of points on the plane $\mathbf{n}_l^T\mathbf{x} = 0$. Using this relation, we can warp all of the masked images onto the coordinate frame of one distinguished image[1] (w.l.o.g. image M_{1l}) as follows:

$$\left(\mathbf{H}_{1k}^l \circ M_{kl}\right)(\mathbf{u}_1) \ \equiv \ M_{kl}\left(\mathbf{H}_{1k}^l\mathbf{u}_1\right). \tag{7}$$

Here, $\mathbf{H}_{1k}^l \circ M_{kl}$ is the masked image M_{kl} warped into the coordinate frame of M_{1l}.

The property which we use to compute $\mathbf{n}_l$ is that, assuming the pixel assignments to the layers B_{kl} are correct, the world is piecewise planar, and the surfaces are Lambertian, the warped images $\mathbf{H}_{1k}^l \circ M_{kl}$ should agree with each other where they overlap. There are a number of functions which can be used to measure the degree of consistency between the warped images, including least squares [4] and robust measures [9, 24]. In both cases, the goal is the same: find the plane equation vector $\mathbf{n}_l$ which maximizes the degree of consistency. Typically, this extremum is found using some form of gradient decent, such as the Gauss-Newton method, and the optimization is performed in a hierarchical (i.e. pyramid based) fashion to avoid local extrema [4]. To apply this standard approach [31], we simply need to derive the Jacobian of the image warp $\mathbf{H}_{1k}^l$ with respect to the parameters of $\mathbf{n}_l$. This is straightforward from Equation (6) because we know the cameras matrices $\mathbf{P}_k$.

2.3 Estimation of the Layer Sprites

Before we can compute the layer sprites L_l, we need to choose 2D coordinate systems for the planes. Such coordinate systems can be specified by a collection of arbitrary (rank 3) camera matrices $\mathbf{Q}_l$.[2] Then, similarly to Equations (5) and (6), we can show that the image coordinates $\mathbf{u}_k$ of the pixel in image M_{kl} which is projected onto the pixel $\mathbf{u}_l$ on the plane $\mathbf{n}_l^T\mathbf{x} = 0$ is given by:

$$\mathbf{u}_k \ = \ \mathbf{P}_k\left((\mathbf{n}_l^T\mathbf{q}_l)\mathbf{I} - \mathbf{q}_l\mathbf{n}_l^T\right)\mathbf{Q}_l^*\mathbf{u}_l \ \equiv \ \mathbf{H}_k^l\mathbf{u}_k \tag{8}$$

where $\mathbf{Q}_l^*$ is the pseudo-inverse of $\mathbf{Q}_l$, and $\mathbf{q}_l$ is a vector in the null space of $\mathbf{Q}_l$. The homography $\mathbf{H}_k^l$ can be used to warp the image M_{kl} forward onto the coordinate frame of the plane $\mathbf{n}_l^T\mathbf{x} = 0$, the result of which is denoted $\mathbf{H}_k^l \circ M_{kl}$. Then, we can estimate the layer sprite (with boolean opacities) by *blending* the warped images:

$$L_l \ = \ \bigoplus_{k=1}^{K}\mathbf{H}_k^l \circ M_{kl} \tag{9}$$

[1] It is possible to add an extra 2D perspective coordinate transformation here. Suppose $\mathbf{H}$ is an arbitrary homography. We could warp each masked image onto $\mathbf{H} \circ \mathbf{H}_{1k}^l \circ M_{kl}(\mathbf{u}_1) \equiv M_{kl}(\mathbf{H}\mathbf{H}_{1k}^l\mathbf{u}_1)$. The addition of the homography $\mathbf{H}$ can be used to remove the dependence on one distinguished image, as advocated by Collins [8].

[2] A suitable choice for $\mathbf{Q}_l$ would be one of the camera matrices $\mathbf{P}_k$, in which case Equation (8) reduces to Equation (6). Another interesting choice is one in which the null space of $\mathbf{Q}_l$ is perpendicular to the plane defined by $\mathbf{n}_l$, and the pseudo-inverse maps the coordinate axes onto perpendicular vectors in the plane (i.e. a camera with a frontal image plane). Note that often we do not want a fronto-parallel camera, since it may unnecessarily warp the input images.

where $\bigoplus$ is the blending operator. There are a number of ways in which blending could be performed. One simple method would be to take the mean of the color values. A refinement would be to use a *feathering* algorithm such as [28], where the average is weighted by the distance of each pixel from the nearest invisible pixel (i.e. $\alpha = 0$) in M_{kl}. Alternatively, robust techniques could be used to estimate L_l. The simplest such example is the median operator, but more sophisticated alternatives exist.

An unfortunate effect of the blending in Equation (9) is that averaging tends to increase image blur. Part of the cause is non-planarity in the scene (which is modeled in Section 2.4), but image noise and resampling error also contribute. One simple method of compensating for this effect is to *deghost* the sprites [28]. Another solution is to use image enhancement techniques such as [16, 21, 7], which can even be used to obtain super-resolution sprites.

2.4 Estimation of the Residual Depth

In general, the scene will not be piecewise planar. To model any non-planarity, we allow the point $\mathbf{u}_l$ on the plane $\mathbf{n}_l^T\mathbf{x} = 0$ to be displaced slightly. We assume it is displaced in the direction of the ray through $\mathbf{u}_l$ defined by the camera matrix $\mathbf{Q}_l$. The distance it is displaced is denoted by $Z_l(\mathbf{u}_l)$, as measured in the direction *normal* to the plane. In this case, the homographic warps used in the previous section are not applicable, but using a similar argument, it is possible to show (see also [19, 23]) that:

$$\mathbf{u}_k \;=\; \mathbf{H}_k^l\mathbf{u}_l + w(\mathbf{u}_l)Z_l(\mathbf{u}_l)\mathbf{t}_{kl} \qquad (10)$$

where $\mathbf{H}_k^l \;=\; \mathbf{P}_k\left((\mathbf{n}_l^T\mathbf{q}_l)\mathbf{I} - \mathbf{q}_l\mathbf{n}_l^T\right)\mathbf{Q}_l^*$ is the planar homography of Section 2.3, $\mathbf{t}_{kl} = \mathbf{P}_k\mathbf{q}_l$ is the epipole, and it is assumed that the vector $\mathbf{n}_l = (n_x, n_y, n_z, n_d)^T$ has been normalized such that $n_x^2 + n_y^2 + n_z^2 = 1$. The term $w(\mathbf{u}_l)$ is a projective scaling factor which equals the reciprocal of $\mathbf{Q}_l^3\mathbf{x}$, where $\mathbf{Q}_l^3$ is the third row of $\mathbf{Q}_l$ and $\mathbf{x}$ is the world coordinate of the point. It is possible to write $w(\mathbf{u}_l)$ as a linear function of the image coordinates $\mathbf{u}_l$, but the dependence on $\mathbf{Q}_l$ and $\mathbf{n}_l$ is quite complicated and so the details are omitted. Equation (10) can be used to map plane coordinates $\mathbf{u}_l$ backwards to image coordinates $\mathbf{u}_k$, or to map the image M_{kl} forwards onto the plane. We denote the result of this warp by $(\mathbf{H}_k^l, \mathbf{t}_{kl}, Z_l) \circ M_{kl}$, or more concisely $\mathbf{W}_k^l \circ M_{kl}$.

Almost any stereo algorithm could be used to compute $Z_l(\mathbf{u}_l)$, although it would be preferable to use one favoring small disparities. Doing so essentially solves a simpler (or what Debevec *et. al* [10] term a *model-based*) stereo problem. To compute the residual depth map, we initially set $Z_l(\mathbf{u}_l)$ to be the value (in a range close to zero) which minimizes the variance of $(\mathbf{H}_k^l, \mathbf{t}_{kl}, Z_l) \circ M_{kl}$ across k. Afterwards a simple smoothing algorithm is applied to $Z_l(\mathbf{u}_l)$. Once the residual depth offsets have been estimated, the layer sprite images should be re-estimated using:

$$L_l \;=\; \bigoplus_{k=1}^{K} (\mathbf{H}_k^l, \mathbf{t}_{kl}, Z_l) \circ M_{kl} \;=\; \bigoplus_{k=1}^{K} \mathbf{W}_k^l \circ M_{kl} \quad (11)$$

rather than Equation (9).

2.5 Pixel Assignment to the Layers

The basis for the computation of the pixel assignments is a comparison of the warped images $\mathbf{W}_k^l \circ M_{kl}$ with the layer sprites L_l.[3] If the pixel assignment was correct (and neglecting resampling issues) these images should be identical where they overlap. Unfortunately, comparing these images does not yield any information outside the current estimates of the masked regions.

To allow the pixel assignments to grow, we take the old estimates of B_{kl} and enlarge them by a few pixels to yield new estimates $\tilde{B}_{kl}$. These new assignments can be computed by iterating simple morphological operations, such as setting $\tilde{B}_{kl} = 1$ for the neighbors of every pixel for which $B_{kl} = 1$. Enlarged masked images are then computed using:

$$\tilde{M}_{kl} = \tilde{B}_{kl} \cdot I_k \qquad (12)$$

and a new estimate of the layer sprite computed using:

$$\tilde{L}_l \;=\; \bigoplus_{k=1}^{K} \mathbf{W}_k^l \circ \tilde{M}_{kl}. \qquad (13)$$

(Here, Z_l is enlarged in $\mathbf{W}_k^l$ so that it declines to zero smoothly outside the old masked region.) One small danger of working with $\tilde{M}_{kl}$ and $\tilde{L}_l$ is that occluded pixels may be blended together with unoccluded pixels and result in poor estimates of the $\tilde{L}_l$. A partial solution to this problem is to use a robust blending operator such as the median. Another part of the solution is, during the blend, weight pixels for which $B_{kl} = 1$ more than those for which $B_{kl} = 0$ (and $\tilde{B}_{kl} = 1$). The weights should depend on the distance of the pixel from the closest pixel for which $B_{kl} = 1$, in a similar manner to the *feathering* algorithm of [28].

Given $\tilde{L}_l$, our approach to pixel assignment is as follows. We first compute a measure $P_{kl}(\mathbf{u}_l)$ of the likelihood that a pixel in $\mathbf{W}_k^l \circ \tilde{M}_{kl}(\mathbf{u}_l)$ is the warped image of the pixel $\mathbf{u}_l$ in the enlarged sprite $\tilde{L}_l$. There are a number of ways of defining P_{kl}. Perhaps the simplest is the *residual intensity difference* [24]:

$$P_{kl} \;=\; \|\tilde{L}_l - \mathbf{W}_k^l \circ \tilde{M}_{kl}\|. \qquad (14)$$

Another is the magnitude of the *residual normal flow*:

$$P_{kl} \;=\; \frac{\|\tilde{L}_l - \mathbf{W}_k^l \circ \tilde{M}_{kl}\|}{\|\boldsymbol{\nabla}\tilde{L}_l\|}. \qquad (15)$$

[3] Alternatively, we could compare the input images I_k with the layer sprite images warped back onto image coordinates $(\mathbf{W}_k^l)^{-1} \circ L_l$. This means comparing the input image with a twice resampled, blended image. Both blending and resampling tend to increase blur, so, even if the pixel assignment was perfect, these images may well differ substantially.

Locally estimated variants of the residual normal flow have been used by Irani and coworkers [16, 17, 15]. A final possibility would be to compute the optical flow between $\mathbf{W}_k^l \circ \tilde{M}_{kl}$ and $\tilde{L}_l$. Then a decreasing function of the magnitude of the flow could be used for P_{kl}.

Next, P_{kl} is warped back into the coordinate system of the input image I_k to yield:

$$\hat{P}_{kl} \;=\; (\mathbf{W}_k^l)^{-1} \circ P_{kl}. \tag{16}$$

This warping tends to blur P_{kl}, but this is acceptable since we will want to smooth the pixel assignment anyway.[4] The new pixel assignment can then be computed by choosing the best possible layer for each pixel:

$$B_{kl}(\mathbf{u}_k) \;=\; \begin{cases} 1 & \text{if } \hat{P}_{kl}(\mathbf{u}_k) \;=\; \min_{l'} \hat{P}_{kl'}(\mathbf{u}_k) \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

3 Layer Refinement by Re-Synthesis

In this section, we describe how the estimates of the layer sprites can be refined, now assuming that their opacities α_l are real valued. We begin by formulating a generative model of the image formation process. Afterwards, we propose a measure of how well the layers re-synthesize the input images, and show how the re-synthesis error can be minimized to refine the estimates of the layer sprites.

3.1 The Image Formation Process

We formulate the generative (forward) model of the image formation process using image compositing operations [6], i.e. by painting the sprites one over another in a back-to-front order. The basic operator used to overlay the sprites is the *over* operator:

$$F \odot B \;\equiv\; F + (1 - \alpha_F)B, \tag{18}$$

where F and B are the foreground and background sprites, and α_F is the opacity of the foreground [22, 6]. This definition of the over operator assumes pre-multiplied opacities, as in Equation (1). The generative model consists of the following two steps:

1. Using the camera matrices, plane equations, and residual depths, warp each layer backwards onto the coordinate frame of image I_k using the inverse of the operator in Section 2.4. This yields the *un-warped* sprite:

$$U_{kl} \;=\; (\mathbf{W}_k^l)^{-1} \circ L_l. \tag{19}$$

   Note that the opacities should be warped along with the color values [6].

2. Composite the un-warped sprites in back-to-front order (which can be computed from the plane equations):

$$S_k \;=\; \bigodot_{l=1}^{L} U_{kl} \;=\; U_{k1} \odot \cdots \odot U_{kl} \tag{20}$$

   to obtain the *synthesized* image S_k. If we have solved the stereo reconstruction problem, and neglecting resampling issues, S_k should match the input I_k.

This last step can be re-written as three simpler steps:

2a. Compute the *visibility* of each un-warped sprite [30]:

$$V_{kl} \;=\; V_{k(l-1)}\,(1 - \alpha_{k(l-1)}) \;=\; \prod_{l'=1}^{l-1}(1 - \alpha_{kl'}) \tag{21}$$

   where α_{kl} is the alpha channel of U_{kl}, and $V_{k1} = 1$.

2b. Compute the masked images, $M_{kl} = V_{kl}U_{kl}$.

2c. Sum up the masked images, $S_k \;= \sum_{l=1}^{L} M_{kl}$.

In these last three substeps, the visibility map makes the contribution of each sprite pixel to the image S_k explicit.

3.2 Minimization of Re-Synthesis Error

As mentioned above, if the layer estimates are accurate, the synthesized image S_k should be very similar to the input image I_k. Therefore, we refine the layer estimates by minimizing the prediction error:

$$\mathcal{C} = \sum_k \sum_{\mathbf{u}_k} \|S_k(\mathbf{u}_k) - I_k(\mathbf{u}_k)\|^2 \tag{22}$$

using a gradient descent algorithm. (In order to further constrain the space of possible solutions, we can add smoothness constraints on the colors and opacities [30].) Rather than trying to optimize over all of the parameters (L_l, $\mathbf{n}_l$, and Z_l) simultaneously, we only adjust the sprite colors and opacities in L_l, and then re-run the previous motion estimation steps to adjust $\mathbf{n}_l$ and Z_l (see Figure 2 and Section 2).

The derivatives of the cost function $\mathcal{C}$ with respect to the colors and opacities in $L_l(\mathbf{u}_l)$ can be computed using the chain rule [30]. In more detail, the visibility map V_{kl} mediates the interaction between the un-warped sprite U_{kl} and the synthesized image S_k, and is itself a function of the opacities in the un-warped sprites U_{kl}. For a fixed warping function $\mathbf{W}_k^l$, the pixels in U_{kl} are linear combinations of the pixels in sprite L_l. This dependence can either be exploited directly using the chain rule to propagate gradients, or alternatively the derivatives of $\mathcal{C}$ with respect to U_{kl} can be warped back into the reference frame of L_l [30].

[4] We may want to smooth P_{kl} even more, e.g. using an isotropic smoother such as a Gaussian. Other alternatives include, (1) performing a color segmentation of each input image and only smoothing within each segment in a similar manner to [2], and (2) smoothing P_{kl} less in the direction of the intensity gradient since strong gradients often coincide with depth discontinuities and hence layer boundaries.

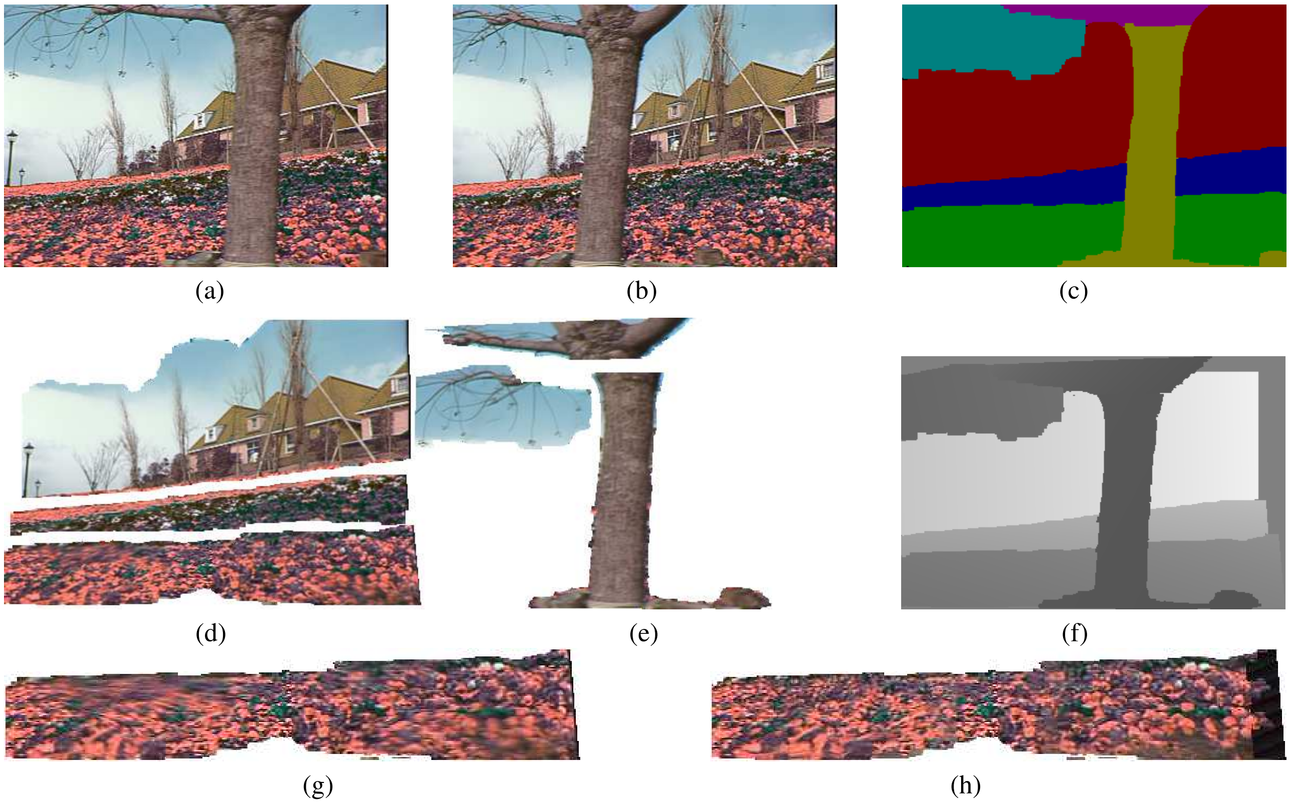

Figure 3: Results on the *flower garden* sequence: (a) first and (b) last input images; (c) initial segmentation into six layers; (d) and (e) the six layer sprites; (f) depth map for planar sprites (darker denotes closer); front layer before (g) and after (h) residual depth estimation.

Figure 4: Results on the *symposium* sequence: (a) third of five images; (b) initial segmentation into six layers; (c) recovered depth map (darker denotes closer); (d) and (e) the five layer sprites; (f) residual depth image for fifth layer.

(a) (b) (c)

Figure 5: 3D views of the reconstructed *symposium* scene: (a) re-synthesized third image (note extended field of view). (b) novel view without residual depth; (c) novel view with residual depth (note the "rounding" of the people).

4 Experiments

To validate our approach, we experimented on two multi-frame data sets. The first of these data sets is a standard motion sequence of a scene containing no independently moving objects. The second consists of 40 images taken simultaneously. The camera geometry is not given for either sequence, so we used point tracking and a standard structure from motion algorithm to estimate the camera matrices. Our experiments do not yet include the results of applying the layer refinement step described Section 3.

To initialize our algorithm, we first decided how many layers were required, and then performed a rough assignment of pixels to layers by hand. Various automated techniques for performing this initial labeling are described in Section 2.1. Next, the automatic hierarchical parametric motion estimation algorithm described in [31] was used to find the 8-parameter homographies between the layers and estimate the layer sprites. (For the experiments presented in this paper, we set $\mathbf{Q}_l = \mathbf{P}_1$, i.e. we reconstructed the sprites in the coordinate system of the first camera.) Using the computed homographies, we found the best plane estimate for each layer using a Euclidean structure from motion algorithm [33].

The results of applying these steps to the MPEG *flower garden* sequence are shown in Figure 3. Figures 3(a) and (b) show the first and last image in the subsequence we used (the first nine even images). Figure 3(c) shows the initial pixel labeling into seven layers. Figures 3(d) and (e) show the sprite images corresponding to each of the seven layers, re-arranged for more compact display. (These sprites are actually the ones computed after residual depth estimation.) Note that because of the blending that takes place during sprite construction, each sprite is larger than its footprint in any one of the input images. Figure 3(f) shows a depth map computed by painting every pixel with its corresponding grey coded Z value, where darker denotes closer.

Once we have recovered the initial geometric structure, we recompute the homographies by directly adjusting the plane equations, as described in Section 2.2. We then run the the residual depth estimation algorithm described in Section 2.4 and recompute the sprites. Since the correspondence is now much better across images, the resulting sprites are much less blurry. Figure 3(g) shows the original sprite obtained for the lower flower bed, while Figure 3(h) shows the same sprite after residual depth estimation.

Our second set of experiments uses five images of a 40-image stereo data set taken at a graphics symposium. Figure 4(a) shows the middle input image, Figure 4(b) shows the initial pixel assignment to layers, Figure 4(c) shows the recovered planar depth map, and Figure 4(f) shows the residual depth map for one of the layers. Figures 4(d) and (e) show the recovered sprites. Figure 5(a) shows the middle image re-synthesized from these sprites. Finally, Figures 5(b–c) show the same sprite collection seen from a novel viewpoint (well outside the range of the original views), first with and then without residual depth correction. The gaps in Figure 5 correspond to parts of the scene which where not visible in any of the five input images.

5 Discussion

We have presented a framework for stereo reconstruction which represents the scene as a collection of approximately planar layers. Each layer consists of a plane equation, a layer sprite image, and a residual depth map. The framework exploits the fact that each layer implicitly lies on a fixed plane in the 3D world. Therefore, we only need to recover three plane parameters per layer, independently of the number of images. We also showed how an initial estimate of the scene structure allows us to reason about image formation. We proposed a forward model of image formation, and derived a measure of how well the layers re-synthesize the input images. Optimizing this measure allows the layer sprites to be refined, and their opacities estimated.

Our initial results are very encouraging, however further work is required to complete an implementation of the entire framework. In particular, we are currently implementing the layer refinement algorithm described in Section 3. Other areas which we are exploring include automatic initialization of the layers and more sophisticated pixel assignment strategies.

Acknowledgements

We would like to thank Michael Cohen, Mei Han, and Jonathan Shade for fruitful discussions, Harry Shum for his implementation of the residual depth estimation algorithm, and the anonymous reviewers for many useful suggestions and comments.

References

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *PAMI*, 17(4):384–401, 1985.

[2] S. Ayer, P. Schroeter, and J. Bigün. Segmentation of moving objects by robust parameter estimation over multiple frames. In *3rd ECCV*, pages 316–327, 1994.

[3] J. R. Bergen, P. J. Burt, R. Hingorani, and S. Peleg. A three-frame algorithm for estimating two-component image motion. *PAMI*, 14(9):886–896, 1992.

[4] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *2nd ECCV*, pages 237–252, 1992.

[5] M.J. Black and A.D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *PAMI*, 18(10):972–986, 1996.

[6] J.F. Blinn. Jim Blinn's corner: Compositing, part 1: Theory. *IEEE Computer Graphics and Applications*, 14(5):83–87, September 1994.

[7] M.-C. Chiang and T.E. Boult. Local blur estimation and super-resolution. In *CVPR '97*, pages 821–826, 1997.

[8] R.T. Collins. A space-sweep approach to true multi-image matching. In *CVPR '96*, pages 358–363, 1996.

[9] T. Darrell and A.P. Pentland. Cooperative robust estimation using layers of support. *PAMI*, 17(5):474–487, 1995.

[10] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH '96*, pages 11–20, 1996.

[11] O.D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[12] S. Hsu, P. Anandan, and S. Peleg. Accurate computation of optical flow by using layered motion representations. In *ICPR '94*, pages 743–746, 1994.

[13] S.S. Intille and A.F. Bobick. Disparity-space images and large occlusion stereo. In *2nd ECCV*, 1994.

[14] M. Irani and P. Anandan. A unified approach to moving object detection in 2D and 3D scenes. In *12th ICPR*, pages 712–717, 1996.

[15] M. Irani, P. Anandan, and S. Hsu. Mosiac based representations of video sequences and their applications. In *5th ICCV*, pages 605–611, 1995.

[16] M. Irani and S. Peleg. Image sequence enhancement using multiple motions analysis. In *CVPR '92*, pages 216–221, 1992.

[17] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *2nd ECCV*, pages 282–287, 1992.

[18] S.X. Ju, M.J. Black, and A.D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *CVPR '96*, pages 307–314, 1996.

[19] R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: A parallax based approach. In *12th ICPR*, pages 685–688, 1994.

[20] J. Lengyel and J. Snyder. Rendering with coherent layers. In *SIGGRAPH '97*, pages 233–242, 1997.

[21] S. Mann and R.W. Picard. Virtual bellows: Constructing high quality stills from video. In *1st ICIP*, pages 363–367, 1994.

[22] T. Porter and T. Duff. Compositing digital images. *SIGGRAPH '84*, pages 253–259, 1984.

[23] H. S. Sawhney. 3D geometry from planar parallax. In *CVPR '94*, pages 929–934, 1994.

[24] H.S. Sawhney and S. Ayer. Compact representations of videos through dominant and multiple motion estimation. *PAMI*, 18(8):814–830, 1996.

[25] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. In *CVPR '96*, pages 343–350, 1996.

[26] S.M. Seitz and C.M. Dyer. Photorealistic scene reconstrcution by space coloring. In *CVPR '97*, pages 1067–1073, 1997.

[27] M. Shizawa and K. Mase. A unified computational theory of motion transparency and motion boundaries based on eigenenergy analysis. In *CVPR '91*, pages 289–295, 1991.

[28] H.-Y. Shum and R. Szeliski. Construction and refinement of panoramic mosaics with global and local alignment. In *6th ICCV*, 1998.

[29] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *CVPR '94*, pages 194–201, 1994.

[30] R. Szeliski and P. Golland. Stereo matching with transparency and matting. In *6th ICCV*, 1998.

[31] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and texture-mapped models. In *SIGGRAPH '97*, pages 251–258, 1997.

[32] J. Torborg and J.T. Kajiya. Talisman: Commodity realtime 3D graphics for the PC. In *SIGGRAPH '96*, pages 353–363, 1996.

[33] T. Viéville, C. Zeller, and L. Robert. Using collineations to compute motion and structure in an uncalibrated image sequence. *IJCV*, 20(3):213–242, 1996.

[34] J.Y.A. Wang and E.H. Adelson. Layered representation for motion analysis. In *CVPR '93*, pages 361–366, 1993.

[35] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR '97*, pages 520–526, 1997.

[36] Y. Weiss and E.H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *CVPR '96*, pages 321–326, 1996.

[37] Y. Yang, A. Yuille, and J. Lu. Local, global, and multilevel stereo matching. In *CVPR '93*, pages 274–279, 1993.