# A Multi-View Approach to Motion and Stereo

Richard Szeliski
*Microsoft Research*
*Redmond, WA 98052-6399*
*szeliski@microsoft.com*

## Abstract

*This paper presents a new approach to computing dense depth and motion estimates from multiple images. Rather than computing a single depth or motion map from such a collection, we associate motion or depth estimates with* each *image in the collection (or at least some subset of the images). This has the advantage that the depth or motion of regions occluded in one image will still be represented in some other image. Thus, tasks such as novel view interpolation or motion-compensated prediction can be solved with greater fidelity. Furthermore, the natural variation in appearance between different images can be captured. To formulate motion and structure recovery, we cast the problem as a global optimization over the unknown motion or depth maps, and use robust smoothness constraints to constrain the space of possible solutions. We develop and evaluate some motion and depth estimation algorithms based on this framework.*

## 1. Introduction

Stereo and motion have long been central research problems in computer vision. Early work was motivated by the desire to recover depth maps and coarse shape and motion models for robotics and object recognition applications. More recently, depth maps obtained from stereo (or alternately dense correspondence maps obtained from motion) have been combined with texture maps extracted from input images in order to create realistic 3-D scenes and environments for virtual reality and virtual studio applications. Unfortunately, the quality and resolution of most of today's algorithms falls quite short of that demanded by these new applications, where even isolated errors in correspondence become readily visible when composited with synthetic graphical elements.

One of the most common errors made by these algorithms is a mis-estimation of depth or motion near occlusion boundaries. Traditional correspondence algorithms assume that every pixel has a corresponding pixel in all other images. Obviously, in occluded regions, this is not so. Furthermore, if only a single depth or motion map is used, it is impossible to predict the appearance of the scene in regions which are occluded (Figure 1). Other problems include dealing with untextured or regularly textured regions, and with viewpoint-dependent effects such as specularities or shading.

One novel approach to tackling these problems is to build a 3D volumetric model of the scene [15, 18]. The scene volume is discretized, often in terms of equal increments of disparity. The goal is then to find the voxels which lie on the surfaces of the objects in the scene. The benefits of such an approach include the equal and efficient treatment of a large number of images [5], the possibility of modeling occlusions [9], and the detection of mixed pixels at occlusion boundaries [18]. Unfortunately, discretizing space volumetrically introduces a large number of degrees of freedom and leads to sampling and aliasing artifacts. To prevent a systematic "fattening" of depth layers near occlusion boundaries, variable window sizes [10] or iterative evidence aggregation [14] can be used. Sub-pixel disparities can be estimated by finding the the analytic minimum of the local error surface [13] or using gradient-based techniques [12], but this requires going back to a single depth/motion map representation.

Another active area of research is the detection of parametric motions within image sequences [20, 3, 21]. Here, the goal is to decompose the images into sub-images, commonly referred to as *layers*, such that the pixels within each layer move with a parametric transformation. For rigid scenes, the layers can be interpreted as planes in 3D being viewed by a moving camera, which results in fewer unknowns. This representation facilitates reasoning about occlusions, permits the computation of accurate out-of-plane displacements, and enables the modeling of *mixed* or *transparent* pixels [1]. Unfortunately, initializing such an algorithm and determining the appropriate number of layers is not straightforward.

Thus, all current correspondence algorithms have their limitations. Single depth or motion maps cannot represent occluded regions not visible in the reference image and usually have problems matching near discontinuities. Volumetric techniques have an excessively large number of degrees of freedom and have limited resolution, which can lead to sampling or aliasing artifacts. Layered motion and stereo

**Figure 1. Slice through a motion sequence spatio-temporal volume. A standard estimation algorithm only estimates the motion at the center frame (⇒), whereas our multi-view approach produces several additional estimates (→). A layered motion model would use two (or more) layers to describe this motion, whereas a volumetric approach would assign one voxel to each "streak".**

algorithms require combinatorial search to determine the correct number of layers and cannot naturally handle true three-dimensional objects (they are better at representing "cutout" scenes). Furthermore, none of these approaches can easily model the variation of scene or object appearance with respect to the viewing position.

In this paper, we propose a new representation which overcomes most of these limitations. Rather than estimating a single depth or motion map, we associate a depth or motion map with *each* input image (or some subset of them, Figure 1). Furthermore, we try to ensure consistency between these different estimates using a *motion compatibility* constraint, and reason about occlusion relationships by computing pixel *visibilities*.

Our new approach is motivated by several target applications. One application is *view interpolation*, where we wish to generate novel views from a collection of images with associated depth maps. The use of multiple depth maps and images allows us to model partially occluded regions and to model view-dependent effects (such as specularities) by blending images with taken from nearby viewpoints [6]. Another application is *motion-compensated frame interpolation* (e.g., for video compression, rate conversion, or de-interlacing), where the ability to predict bi-directionally (from both previous and future keyframes) yield better prediction results [11]. A third application is as a low-level representation from which segmentation and layer extraction (or 3D model construction) can take place.

The remainder of the paper is structured as follows. In Section 2, we present our basic multi-view image matching framework. In Section 3, we explain in more details the cost functions used to formulate our estimation problem, and present our new method for estimating visibility in multi-image sequences. Section 4 presents our estimation algorithm, which combines ideas from hierarchical estimation, correlation-style search, and sub-pixel motion estimation. We present some experimental results in Section 5. We conclude the paper with a discussion of these results, and a list of topics for future research. A more comprehensive set of references and experimental results can be found in the extended version of this paper [19].

# 2. The multi-view framework

Assume that we are given a collection of images $\{I_t(\mathbf{x}_t)\}$, where $I_t$ is the image at time or location $t$, and $\mathbf{x}_t = (x_t, y_t)$ indexes pixels in image $I_t$. A simple way to formulate a multi-view matching criterion is

$$\mathcal{C}(\{\mathbf{u}_s\}) = \sum_s \sum_{t \in \mathcal{N}(s)} w_{st} \sum_{\mathbf{x}_s} \rho\left(I_s(\mathbf{x}_s) - I_t(\mathbf{x}_t)\right). \quad (1)$$

The images $I_s$ are considered the *keyframes* (or *key-views*) for which we will estimate a motion or depth estimate $\mathbf{u}_s(\mathbf{x}_s)$ (see Section 2.1 for a description of our motion models). The decision as to which images are keyframes is problem-dependent, much like the selection of I and P frames in video compression [11]. For 3D view interpolation, one possible choice of keyframes would be a collection of *characteristic views*.

Images $I_t, t \in \mathcal{N}(s)$ are *neighboring frames* (or views), for which we will require that corresponding pixel brightnesses (or colors) agree. The pixel coordinate $\mathbf{x}_t$ corresponding to a give keyframe pixel $\mathbf{x}_s$ with flow/depth $\mathbf{u}_s$ can be computed according to the motion model (Section 2.1). The constants $w_{st}$ are the *inter-frame weights* which dictate how much neighboring frame $t$ will contribute to the estimate of $\mathbf{u}_s$.

Corresponding pixel brightness or color differences are passed through a robust penalty function $\rho$, which is discussed in more detail in Section 2.2. In the case of color images, we currently pass each color channel separately through the robust penalty function. A better approach would be to compute a reasonable color-space distance between pixels, and pass this through a robust penalty.

## 2.1. Motion models

Given our basic matching criterion, a variety of motion models can be used, depending on the imaging/acquisition setup and the problem at hand. Bergen *et al.* [2] present a variety of instantaneous motion models in a unified estimation framework. Szeliski and Coughlan [17] present a similar set of motion models for finite motion. In this paper, we will focus on two motion models: *constant flow* (uniform velocity), and *rigid body motion*.

The constant flow motion model assumes a (locally) constant velocity,

$$\mathbf{x}_t = \mathbf{x}_s + (t - s)\mathbf{u}_s(\mathbf{x}_s). \quad (2)$$

This model is appropriate when processing regular video with a relatively small sliding window of analysis.

The rigid motion model assumes that the camera is moving in a rigid scene or observing a single rigid moving object,

but does *not* assume a uniform temporal sampling rate (e.g., the pictures can be taken by different cameras). The motion model is given by

$$\mathbf{x}_t = \mathcal{P}\left(\mathbf{M}_{ts}\hat{\mathbf{x}}_s + \mathbf{e}_{ts}d_s(\mathbf{x}_s)\right), \quad (3)$$

where $\mathbf{M}_{ts}$ is a *homography* describing a global parametric motion, $d_s(\mathbf{x}_s)$ is a per-pixel displacement that adds some motion towards the *epipole* $\mathbf{e}_{ts}$, and $\mathcal{P}(x, y, z) = (x/z, y/z)$ is the perspective projection operator.

For a calibrated camera with intrinsic viewing matrix $\mathbf{V}_t$, we have $\mathbf{M}_{ts} = \mathbf{V}_t\mathbf{R}_t\mathbf{R}_s^{-1}\mathbf{V}_s^{-1}$ and $\mathbf{e}_{ts} = \mathbf{V}_t\mathbf{R}_t(\mathbf{c}_s - \mathbf{c}_t)$, where $\mathbf{R}_t$ is the camera's orientation and $\mathbf{c}_t$ is its position in space. In this case, $\mathbf{M}_{ts}$ is the homography corresponding to the plane at infinity. If all of the cameras live in the same plane with their optical axes perpendicular to the plane, $d_s$ is the *inverse depth* (sometimes called the *disparity* [10]) of a pixel.

### 2.2. Robust penalty functions

In order to account for *outliers* among the pixel correspondences (e.g., because pixels might be occluded in some images), we use a robust matching criterion. Black and Rangarajan [4] provide a nice survey of robust statistics applied to image matching and image segmentation problems.

In our work, we use a *contaminated Gaussian* distribution, which is a mixture of a Gaussian distribution and a uniform distribution. The probability function is

$$p(x; \sigma, \epsilon) = Z^{-1}(1 - \epsilon)\exp(-x^2/(2\sigma^2)) + \epsilon, \quad (4)$$

where $\sigma$ in the standard deviation of the *inlier* process, $\epsilon$ is the probability of finding an *outlier*, and $Z$ is a normalizing constant. The robust penalty function is the negative log likelihood,

$$\rho(x; \sigma, \epsilon) = -\log\left((1 - \epsilon)\exp(-x^2/(2\sigma^2)) + \epsilon\right). \quad (5)$$

Our main motivation for using a contaminated Gaussian is to explicitly represent and reason about the inlier and outlier processes separately. For example, in Section 3.3 we propose a robust controlled smoothness constraint where the strength of the constraint depends on the neighboring pixel color similarity. In Appendix A we show that pixel color similarity affects the outlier probability but not the inlier variance. Using a contaminated Gaussian gives us a principled way to incorporate these effects.

## 3. Optimization criteria

The actual cost function we use consists of three terms,

$$\mathcal{C} = \mathcal{C}_I + \mathcal{C}_T + \mathcal{C}_S, \quad (6)$$

where $\mathcal{C}_I$ measures the *brightness compatibility*, $\mathcal{C}_T$ measures the temporal *flow compatibility*, and $\mathcal{C}_S$ measures the *flow smoothness*. Below, we give more details on each of these three terms.

### 3.1. Brightness compatibility

The brightness compatibility term measures the degree of agreement in brightness or color between corresponding pixels,

$$\mathcal{C}_I(\{\mathbf{u}_s\}) = \sum_s \sum_{t \in \mathcal{N}(s)} w_{st} \sum_{\mathbf{x}_s} v_{st}(\mathbf{x}_s) e_{st}(\mathbf{x}_s), \quad (7)$$

where

$$e_{st}(\mathbf{x}_s) = \rho_I\left(I_s(\mathbf{x}_s) - \gamma_{st} I_t(\mathbf{x}_t) - \beta_{st}; \sigma_I, \epsilon_I\right). \quad (8)$$

Compared with Equation (1), we have added a visibility factor $v_{st}(\mathbf{x}_s)$, which encodes whether pixel $\mathbf{x}_s$ is *visible* in image $I_t$ (Section 3.4). We have also generalized the robust penalty to allow for a global bias ($\beta_{st}$) and gain ($\gamma_{st}$) change.

### 3.2. Flow compatibility

The controlled flow compatibility constraint,

$$\mathcal{C}_T(\{\mathbf{u}_s\}) = \sum_s \sum_{t \in \mathcal{N}(s)} w_{st} \sum_{\mathbf{x}_s} v_{st}(\mathbf{x}_s) c_{st}(\mathbf{x}_s), \quad (9)$$

with

$$c_{st}(\mathbf{x}_s) = \rho_T\left(|\mathbf{u}_s(\mathbf{x}_s) - \mathbf{u}_t(\mathbf{x}_t)|; \sigma_T, \epsilon_T, \right) \quad (10)$$

enforces *mutual consistency* between flow estimates at different neighboring keyframes.

For the constant flow motion model, the variance $\sigma_T$ can be used to account for drift in the velocities (acceleration). For a rigid scene, we don't expect any drift. However, the $d_s$'s may actually be related by a projective transformation [16]. For a scene with object far enough away or for cameras arranged in a plane perpendicular to their optical axes, this is not a problem.

### 3.3. Flow smoothness

The final cost term we use is a controlled flow smoothness constraint,

$$\mathcal{C}_S(\{\mathbf{u}_s\}) = \sum_s \sum_{\mathbf{x}_s} f_s(\mathbf{x}_s), \quad (11)$$

with

$$f_s(\mathbf{x}_s) = \sum_{\mathbf{x}' \in \mathcal{N}_4(\mathbf{x})} \rho_S(|\mathbf{u}_s(\mathbf{x}) - \mathbf{u}_s(\mathbf{x}')|; \sigma_S, \epsilon_S(\mathbf{x}, \mathbf{x}')). \quad (12)$$

The value of the outlier probability is based on the brightness/color difference between neighboring pixels

$$\epsilon_S(\mathbf{x}, \mathbf{x}') = \Psi(I_s(\mathbf{x}) - I_s(\mathbf{x}')).$$

Appendix A derives the form of this $\Psi$ function and justifies the dependence of the outlier probability on the local intensity variation.

### 3.4. Determining visibility

One of the most novel aspects of our multi-view matching framework is the explicit use of visibility to prevent the matching of pixels into areas which are occluded. This kind of visibility computation is commonly used in a number of computer graphics algorithms, e.g., algorithms for computing shadows and algorithms for recovering texture maps from images. It is also used in some more recent stereo matching algorithms [15, 18, 1].

When working with rigid motion and depth/disparity estimates, the visibility computation is fairly straightforward. Consider two images, $I_s$ and $I_t$. We wish to compute whether pixel $\mathbf{x}_s$ in image $I_s$ is visible at location $\mathbf{x}_t$ in image $I_t$. If $\mathbf{x}_s$ is visible, the values of $d_s(\mathbf{x}_s)$ and $d_t(\mathbf{x}_t)$ should be the same.[1] If $\mathbf{x}_s$ is occluded, then $d_t(\mathbf{x}_t) > d_s(\mathbf{x}_s)$ (assuming $d = 0$ at infinity and positive elsewhere in front of the camera). We therefore have

$$v_{st}(\mathbf{x}_s) = ((d_t(\mathbf{x}_s) - d_s(\mathbf{x}_t)) \leq \delta), \tag{13}$$

where $\delta$ is a threshold to account for errors in estimation and warping.

When we have general 2-D flow, the situation is more complicated. In general, we cannot determine whether an occluding layer will be moving slower or faster than a pixel in a occluded layer. (For a translating camera, we can always pan the camera is such a way that either case occurs.) Therefore, the best we can do is to simply compare the flow estimates, and infer that a pixel may be invisible if the two velocities disagree,

$$v_{st}(\mathbf{x}_s) = (\|\mathbf{u}_s(\mathbf{x}_s) - \mathbf{u}_t(\mathbf{x}_t)\| \leq \delta). \tag{14}$$

Regardless of the motion model, we also set $v_{st}(\mathbf{x}_s) = 0$ whenever the corresponding pixel $\mathbf{x}_t$ is outside the boundaries of $I_t$, i.e., $\mathbf{x}_t \notin I_t$.

## 4. Estimation algorithm

In order to determine the best possible algorithm characteristics and to compare different design choices and algorithm components, we have developed a general-purpose framework which combines ideas from hierarchical estimation [2], correlation-style search [13, 10], and sub-pixel motion/disparity estimation [12, 13].

Our algorithm operates in two phases. During an initialization phase, we estimate the flows independently for each keyframe. Since we do not yet have any good motion estimates for other frames, the flow compatibility term $\mathcal{C}_T$ is ignored, and no visibilities are computed (i.e., $v_{st} = 1$). In the second phase, we enforce flow compatibility and compute visibilities based on the current collection of flow estimates $\{\mathbf{u}_s\}$.

[1] See the discussion in Section 3.2 of how disparities may have to be re-mapped between images in certain camera configurations.

### 4.1. Computing initial estimates

Our algorithm is hierarchical, i.e., the matching can occur at any level in a multi-resolution pyramid, and results from coarser levels can be used to initialize estimates at a finer level. Hierarchical matching both results in a more efficient algorithm, since fewer pixels are examined at coarser levels, and usually results in better quality estimates, since a wider range of motions can be searched and a better local minimum can be found.

Within each level, we can use one of two techniques to improve the current estimates: explicit correlation-style search, or Lucas-Kanade style gradient descent. Due to space limitations, we do not describe the second approach, since it can be derived using the techniques described in [2, 17].

In correlation-style search, we evaluate several motion or disparity hypotheses at once, and then locally pick the one which results in the lowest local cost function. To rank the hypotheses, we evaluate the local error function $e_{st}(\mathbf{x}_s, \hat{\mathbf{u}}_s)$ given in Equation (8) (the dependence on $\hat{\mathbf{u}}_s$ is made explicit). The flow hypotheses $\hat{\mathbf{u}}_s$ are obtained from $\hat{\mathbf{u}}_s = \mathbf{u}_s + \mathbf{\Delta u}_s$, where $\mathbf{u}_s$ is the current estimate, $\mathbf{\Delta u}_s = (iS, jS)$, $S$ is a step size, and $i = -N \ldots N, j = -N \ldots N$ is a $(2N+1) \times (2N+1)$ search window. For rigid motion, only a 1-D search window of size $(2N+1)$ over possible $d$ values is used.

In other words, we take the current flow field $\mathbf{u}_s$ and add a fixed step in $(u, v)$ before performing the re-sampling (warping) of image $I_t$. This is similar to the iterative re-warping algorithms described in [2, 17], as opposed to algorithms based on shifting a square correlation window [12, 13, 10].

The values of the local error function $e_{st}(\mathbf{x}_s, \hat{\mathbf{u}}_s)$ are usually not sufficient to reliably determine a winning $\hat{\mathbf{u}}_s$ at each pixel. Traditionally, two approaches have been used to overcome this problem. The first is to aggregate evidence spatially, using square windows (of potentially variable size) [10], convolution, pyramid-based smoothing [2], or non-linear diffusion [14]. In this paper, we use spatial convolution

$$\tilde{e}_{st}(\mathbf{x}_s, \hat{\mathbf{u}}_s) = e_{st}(\mathbf{x}_s, \hat{\mathbf{u}}_s) * W(\mathbf{x}), \tag{15}$$

where $W(\mathbf{x})$ is an iterated separable $(1/4, 1/2, 1/4)$ convolution kernel.

The other major approach to local ambiguity is the use of smoothness constraints [8]. Our algorithm uses the smoothness constraints described in Section 3.3. In our current implementation, we disable smoothness constraints when performing the initial estimate (with $\mathbf{u}_s = 0$), and then enable them and reduce the amount of spatial aggregation.

To find the best flow hypothesis at each pixel, we sum up the spatially aggregated error function for each temporal neighbor and add in the smoothness term to obtain a local cost function

$$\mathcal{C}_L(\mathbf{x}_s, \hat{\mathbf{u}}_s) = \sum_{t \in \mathcal{N}(s)} w_{st} \tilde{e}_{st}(\mathbf{x}_s, \hat{\mathbf{u}}_s) + f_s(\mathbf{x}_s, \hat{\mathbf{u}}_s), \tag{16}$$

Based on this set of cost estimates, we pick the $\hat{\mathbf{u}}_s$ with the lowest (best) cost at each pixel. (This corresponds to the "winner-take-all" step of many stereo algorithms.)

To obtain motion estimates with better accuracy, we compute a *fractional* motion estimate by fitting a quadratic cost function to the cost function values around the minimum and analytically computing its minimum [13].

### 4.2. Multi-View Estimation

Once we have computed an initial set of motion estimates $\{\mathbf{u}_s\}$, we can now compute visibilities $v_{st}(\mathbf{x}_s)$ and add in the motion compatibility constraint $\mathcal{C}_T$. For those pixels which are not visible in a given frame, i.e., $v_{st}(\mathbf{x}_s) = 0$, what cost function should we assign? This issue arises not only when performing multi-view estimation, but even in the initial independent motion estimation stage, whenever pixels are mapped outside the boundaries of an image, i.e., $\mathbf{x}_t \notin I_t$.

One possibility is to not pay any penalty, i.e., to set $e_{st}(\mathbf{x}_s, \hat{\mathbf{u}}_s)$ (and $c_{st}$) to 0 whenever $v_{st}(\mathbf{x}_s) = 0$. Unfortunately, this encourages pixels near image borders to have large, outward-going flows. Another possibility is to set $e_{st}(\mathbf{x}_s, \hat{\mathbf{u}}_s) = \rho(\infty)$. Unfortunately, this encourages pixels near image borders to have inward-directed flows.

The solution we have devised is to use the visibility field $v_{st}$ as a mask for a morphological fill operation. In other words, we replace entries in $\mathcal{C}_L(\mathbf{x}_s, \hat{\mathbf{u}}_s)$ with their neighbors' values whenever $v_{st}(\mathbf{x}_s) = 0$. The actual filling algorithm we use is a variant of the multiresolution push/pull algorithm described in [7].

The multi-view estimation algorithm can be repeated several times, at each iteration obtaining better estimates of motion and visibility. We currently perform this *sweeping* through the keyframes, instead of performing a single global estimation, because it is easier to implement and requires less memory.

## 5. Experiments

We have applied our multi-view matching algorithm to a number of image sequences, both where the camera motion is known (based on tracking points and computing structure from motion), and where the flow is uniform over time (video sequences). Figures 2 and 3 show some representative results and illustrate some of the features of our algorithm.

In both sets of figures, images (a–c) show the first, middle, and last image in the sequence (we used the first 4 even images from the *flower garden* sequence and 5 out of 40 images from the *symposium* sequence). The depth maps estimated by the initial, independent analysis algorithm (Section 4.1) are shown in images (e–g). The final results of applying our multi-view estimation algorithm (Section 4.2) with flow smoothness, flow compatibility, and visibility estimation are shown in images (i–k). Notice the improved quality of the estimates obtained with the multi-view estimation algorithm, especially in regions that are partially occluded. For example, in Figure 2, since the tree is moving from right to left, the occluded region is to the left of the tree in the first image, and to the right of the tree in the last one. Notice how the opposite edge of the trunk (where disocclusions are occurring) looks "crisp".

Image (d) in both Figures shows the results of warping one image based on the flow computed in another image. Displaying these warped images as the algorithm progresses is a very useful way to debug the algorithm and to assess the quality of the motion estimates. Without visibility computation, image (d) shows how the pixels in occluded regions draw their colors somewhere from the foreground regions (e.g., the tree trunk in Figure 2 and the people's heads in Figure 3).

Images (h) and (i) show the warped images with invisible pixels flagged as black (the images were generated after the initial and final estimation stages, and hence correspond to the flow fields shown to their left). Notice how the algorithm correctly labels most of the occluded pixels, especially after the final estimation. Notice, also, that some regions without texture such as the sky sometimes erroneously indicate occlusion. Using more smoothing or adding a check that occluder and occludees have different colors could be used to eliminate this problem (which is actually harmless, if we are using our matcher for view interpolation or motion prediction applications).

## 6. Discussion

The experimental results we have obtained so far are encouraging, but still leave room for improvement. In particular, the smoothness of the final estimates and the *sharpness* of the motion discontinuities is not as high as that obtainable with layered motion estimates [20, 21, 1]. This is particularly true in occluded regions: layered models will apply the layer's motion to the occluded regions, while we use a weak smoothness constraint.

In future work, we would like to apply our multi-view framework to view interpolation and motion-based prediction. Since a novel view does not come with an associated $\mathbf{u}_t$ map, we must first forward warp motion estimates from neighboring keyframes, using z-buffering to resolve ambiguities when several pixels map to the same destination [16]. The novel view can then be generated as a *blend* of original views, taking into account pixel visibilities.

We would also like to add super-resolution and de-noising to our multi-view estimation framework. Rather than optimizing our original brightness compatibility constraint, we would minimize $\rho(\hat{I}_s(\mathbf{x}_s) - I_t(\mathbf{x}_t))$ where $\hat{I}_s$ is a de-noised and possibly super-resolved image (texture) estimate. We would also like to investigate explicitly representing discontinuities, since these are where many of the errors occur.

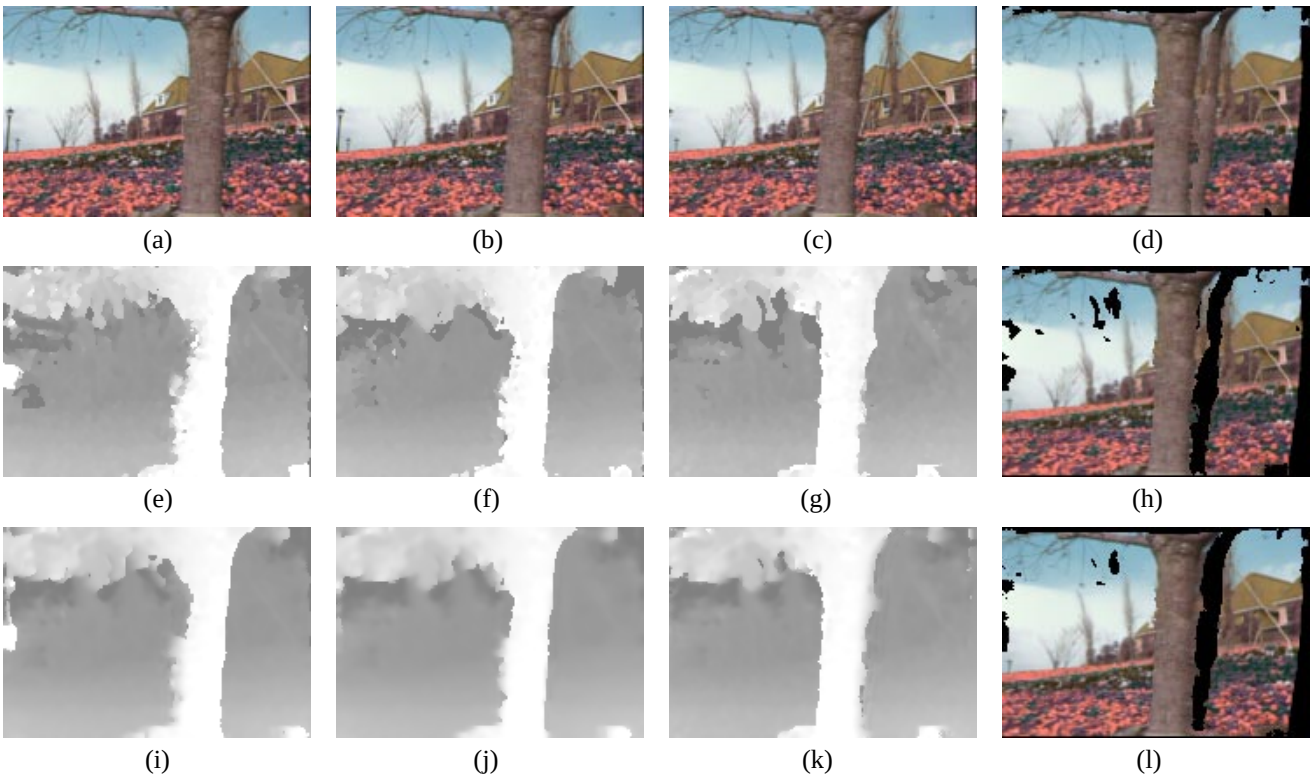


**Figure 2. Results on the *flower garden* sequence: (a–c) first, second, and fourth (last) frame; (e–g) initial depth estimates; (i–k) refined (multi-view) depth estimates. Warped (resampled) images: (d) after initial estimate; (h) with visibility computation; (l) with refined estimates.**


**Figure 3. Results on the *symposium* sequence: (a–c) first, third, and fifth (last) frame; (e–g) initial depth estimates; (i–k) refined (multi-view) depth estimates. Warped (resampled) images: (d) after initial estimate; (h) with visibility computation; (l) with refined estimates.**

# 7. Conclusions

In this paper, we have developed a novel multi-view framework for estimating dense motion and correspondence maps. Our framework simultaneously produces estimates for a subset of the input images, thereby representing motion or depth in partially occluded regions and explicitly modeling the variation in appearance between different views. Our framework is based on minimizing a three part cost function, which consists of a brightness compatibility, a motion compatibility, and a motion smoothness term. Our novel motion smoothness terms uses the presence of color/brightness discontinuities to modify the probability of motion smoothness violation (outlier). We have also developed some novel techniques for determining the visibility of pixels in neighboring images, and used this visibility to affect the values of the brightness and motion compatibility constraints.

While our preliminary experimental results look encouraging, there remains much work to be done in developing truly accurate and robust correspondence algorithms. We believe that the development of such algorithms will be crucial in promoting a wider use of image-based modeling in novel applications such as virtual environments, image-based rendering, and video processing.

# References


[1] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *CVPR'98*, pp. 434–441, Santa Barbara, June 1998.

[2] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV'92*, pp. 237–252, Santa Margherita, May 1992.

[3] M. J. Black and A. D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Trans. Patt. Anal. Mach. Intell.*, 18(10):972–986, October 1996.

[4] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Intl. J. Comp. Vision*, 19(1):57–91, 1996.

[5] R. T. Collins. A space-sweep approach to true multi-image matching. In *CVPR'98*, pp. 358–363, San Francisco, California, June 1996.

[6] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics (SIGGRAPH'96)*, pp. 11–20, August 1996.

[7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Computer Graphics (SIGGRAPH'96)* pp. 43–54, August 1996.

[8] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[9] S. S. Intille and A. F. Bobick. Disparity-space images and large occlusion stereo. In *ECCV'94*, Stockholm, Sweden, May 1994. Springer-Verlag.

[10] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Trans. Patt. Anal. Mach. Intell.*, 16(9):920–932, Sept. 1994.

[11] D. Le Gall. MPEG: A video compression standard for multimedia applications. *CACM*, 34(4):44–58, April 1991.

[12] B. D. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In *IJCAI-81*, pp. 674–679, Vancouver, 1981.

[13] L. H. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *Intl. J. Comp. Vision*, 3:209–236, 1989.

[14] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *Intl. J. Comp. Vision*, 28(2):155–174, July 1998.

[15] S. M. Seitz and C. M. Dyer. Photorealistic scene reconstrcution by space coloring. In *CVPR'97*, pp. 1067–1073, San Juan, Puerto Rico, June 1997.

[16] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *Computer Graphics (SIGGRAPH'98) Proceedings*, pp. 231–242, Orlando, July 1998.

[17] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. *Intl. J. Comp. Vision*, 22(3):199–218, March/April 1997.

[18] R. Szeliski and P. Golland. Stereo matching with transparency and matting. In *ICCV'98*, pp. 517–524, Bombay, January 1998.

[19] R. Szeliski. A Multi-View Approach to Motion and Stereo. Technical Report MSR-TR-99-19, Microsoft Research, May 1999. http://www.research.microsoft.com/pubs/.

[20] J. Y. A. Wang and E. H. Adelson. Layered representation for motion analysis. In *CVPR'93*, pp. 361–366, New York, New York, June 1993.

[21] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR'97*, pp. 520–526, San Juan, Puerto Rico, June 1997.


# A. An image-dependent flow smoothness constraint

To develop our constraint, assume that we know the prior probability $p_D$ that two neighboring pixels straddle a motion discontinuity (i.e., that they live on different surfaces). The distribution of the brightness or color differences between two neighboring pixels depends on the event $D$ that they live on different surfaces, i.e., we have two distributions $p(I_s(\mathbf{x}) - I_s(\mathbf{x}')|\overline{D})$ and $p(I_s(\mathbf{x}) - I_s(\mathbf{x}')|D)$. These distributions can either be guessed (say as contaminated Gaussians, with the probability of outliers much higher in the case of $D$), or estimated from labelled image data.

Given these distributions and the prior probability $p_D$, we can apply Bayes' Rule to calculate $\Psi(I_s(\mathbf{x}) - I_s(\mathbf{x}')) = p(D|I_s(\mathbf{x}) - I_s(\mathbf{x}'))$. (This function will typically start at some small probability $\epsilon_0$ for small color differences, and increase to a final value $\epsilon_1$ for large differences.) This posterior probability of a motion discontinuity can then be plugged in as the local value of $\epsilon_S$ in our controlled motion continuity constraint (12).