To Appear in the Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition

Shape and Motion Carving in 6D

Sundar Vedula, Simon Baker, Steven Seitz, and Takeo Kanade
The Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15213

Abstract

The motion of a non-rigid scene over time imposes more constraints on its structure than those derived from images at a single time instant alone. An algorithm is presented for simultaneously recovering dense scene shape and scene flow (i.e., the instantaneous 3D motion at every point in the scene). The algorithm operates by carving away hexels, or points in the 6D space of all possible shapes and flows that are inconsistent with the images captured at either time instant, or across time. The recovered shape is demonstrated to be more accurate than that recovered using images at a single time instant. Applications of the combined scene shape and flow include motion capture for animation, retiming of videos, and non-rigid motion analysis.

1 Introduction

The image of a rigid object varies in a highly constrained manner as either the object or camera moves. This simple observation has led to a large body of techniques for reconstructing rigid scenes from multiple image sequences. (See, for example, [Waxman and Duncan, 1986, Young and Chellappa, 1999, Zhang and Faugeras, 1992].) The problem of modeling *non-rigid* scenes is far less well-understood, an unfortunate fact given that much of the real world moves non-rigidly. A major difficulty is the lack of general-purpose constraints that govern non-rigid motion. Previous research has therefore focused on specific objects and motions, such as global parametric models [Metaxas and Terzopoulos, 1993], cardiac motion [Pentland and Horowitz, 1991], articulated figures [Bregler and Malik, 1998], faces [Guenter *et al.*, 1998], and curves [Carceroni and Kutulakos, 1999].

Rather than making strong assumptions about the complexity of the motion, we propose instead to use two very general constraints, namely that (1) a fixed point on an object projects to pixels of approximately the same color in all images at consecutive time instants, and (2) the motion between frames is bounded. The first constraint is an extension of the notion of photo-consistency, introduced in [Seitz and Dyer, 1999], to time-varying scenes. The second constraint imposes a weak regularization criterion that improves reconstruction accuracy without penalizing complex shapes or motions. To enable recovery of dense shape and motion data, we assume that the scene is imaged at each time instant from at least two known viewpoints.

While shape models can be recovered using photo-consistency at a single time instant, we instead propose to integrate the computation of shape and motion using the combined images at two time instants. The advantages of this approach are two-fold. First, in addition to recovering shape, our approach computes *Scene flow* (defined as the instantaneous motion for every point on the surface, see [Vedula *et al.*, 1999]). Knowledge of motion data such as scene flow has numerous potential applications ranging from motion analysis tasks, to motion capture for character animation. Second, integrating the recovery of shape and motion into one procedure should produce superior shape estimates compared to models obtained from images taken at a single time instant.

We work in the 6D space of all possible scene shapes at two time instants, and the scene flow that relates them. We define a *hexel* as a point in this 6D space, characterized by the beginning and ending position of a fixed point on an object. The shape and motion of a scene is a 2D manifold in this 6D space (strictly only in the continuous case, and only to well-behaved surface patches and motions). We formulate the problem of simultaneously computing shape and motion as one of determining which points in the 6D space lie on this 2D space-time manifold. Our algorithm operates by *carving away* hexels that do not lie on the manifold. In particular, a hexel is carved if it corresponds to points at the two time instants that are not photo-consistent, i.e., whose projections do not agree in all images at both time instants.

This approach is closely related to the occupancy decisions made by volumetric stereo algorithms such as [Seitz and Dyer, 1999] and [Kutulakos and Seitz, 1999]. The key component of most volumetric algorithms is the representation of the volume of the scene. Many algorithms use the concept of a *disparity space* [Scharstein and Szeliski, 1998], the basis for which dates back to the work of Marr and Poggio [1979]. In several other approaches, the scene is represented as a set of rectangular voxels [Chen and Medioni, 1999] [De Bonet and Viola, 1999]. The volume can also be represented as the mesh graph defined by these voxels [Roy and Cox, 1998] or by sweeping a plane through it [Collins, 1996]. In this paper we work in a 6D space which represents the volume of a scene at two consecutive time instants. This allows us to enforce photo-consistency over both space and time simultaneously.

We demonstrate the validity of our approach empirically by recovering shape and instantaneous scene flow on real

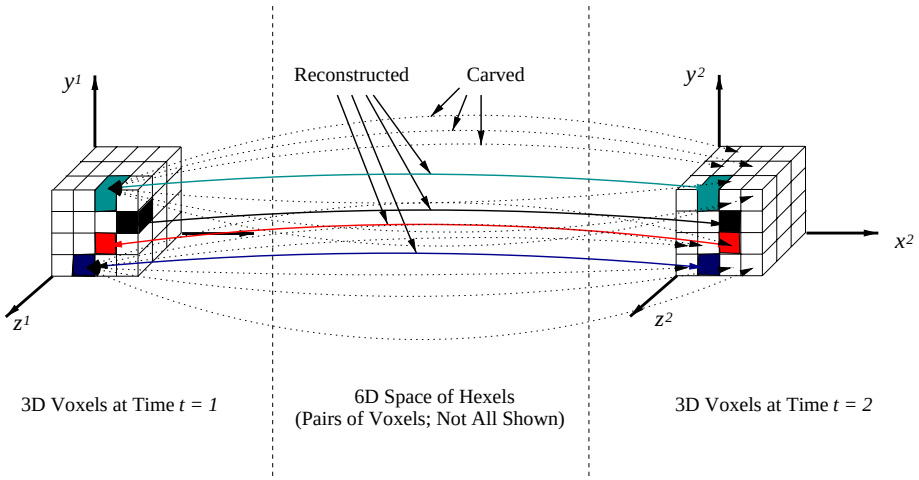


Figure 1: An illustration of the 6D space of hexels. A hexel can be regarded as a pair of corresponding 3D voxels, one at each time instant. The 6D hexel $(x^1, y^1, z^1, x^2, y^2, z^2)$ defines the two voxels: (x^1, y^1, z^1) at $t = 1$ and (x^2, y^2, z^2) at $t = 2$. It also defines the scene flow $(\Delta x^1, \Delta y^1, \Delta z^1) = (x^2 - x^1, y^2 - y^1, z^2 - z^1)$ that relates them. Estimating instantaneous scene shape and motion can be posed as deciding which hexels to carve and which to reconstruct. A hexel is reconstructed if the voxels that it corresponds to are photo-consistent, both separately at the two time instants, and together across time. If a hexel is reconstructed, the two voxels that it corresponds to are both reconstructed and the scene flow between them calculated as above.

image data. We compare our shape results to the shape computed at a single time instant by 3D space carving [Seitz and Dyer, 1999]. These results show that the recovered model using our 6D carving algorithm contains fewer outlier voxels. We also demonstrate that the scene flow that is computed is consistent with the motion.

The rest of this paper is organized as follows. Section 2 introduces the framework for representing shape and motion in a 6D space and defines 6D photo-consistency. Our 6D carving algorithm is presented in Section 3. Section 4 presents experimental results obtained by applying this algorithm to a set of image sequences from a multi-camera 3D room. Section 5 concludes the paper with a discussion.

2 The 6D Space of Scene Shape and Flow

If a scene point moves from $\mathbf{x}^1 = (x^1, y^1, z^1)$ at $t = 1$ to $\mathbf{x}^2 = (x^2, y^2, z^2) = (x^1 + \Delta x^1, y^1 + \Delta y^1, z^1 + \Delta z^1)$ at $t = 2$, there are two natural ways of representing the space of possible scene shapes and flows. One possibility is the asymmetric 6D space of all 6-tuples:

$$(x^1, y^1, z^1, \Delta x^1, \Delta y^1, \Delta z^1). \quad (1)$$

Here, the shape (x^1, y^1, z^1) at time $t = 1$ and the scene flow $(\Delta x^1, \Delta y^1, \Delta z^1)$ uniquely determine the shape at time $t = 2$, as above. The second possibility is the symmetric 6D space of 6-tuples:

$$(x^1, y^1, z^1, x^2, y^2, z^2). \quad (2)$$

Here, the shapes at the two times uniquely determine the flow $(\Delta x^1, \Delta y^1, \Delta z^1) = (x^2 - x^1, y^2 - y^1, z^2 - z^1)$.

We chose to treat both time instants equally and work in the symmetric space because it turned out to be more natural for our algorithm. One advantage of the asymmetric definition, however, is that the space can be kept smaller, since $\Delta\mathbf{x}$ usually has a much smaller range of magnitudes than $\mathbf{x}$. The asymmetric representation may therefore be more natural for other algorithms, particularly ones such as [Vedula *et al.*, 1999] that first compute shape at $t = 1$ and then compute scene flow and predict shape at $t = 2$.

A hexel is a 6D vector of the form in Equation (2), analogous to a voxel in 3D. Figure 1 contains an illustration of the 6D hexel space. The hexel space is the Cartesian product of the two voxel spaces; i.e., it is the set of all pairs of voxels, one at each time instant. A hexel is a single entity that defines one voxel at each time, and the scene flow that relates them. Therefore, it does not exist at either time, but simultaneously at both.

A surface that moves continuously from $t = 1$ to $t = 2$ is a 2D manifold in the 6D hexel space. The manifold implicitly includes information on the instantaneous motion, in addition to the shape of the surface. However, manifolds only exist in the continuous domain. Therefore, when we work with discrete hexels, we attempt to find an approximation of the manifold as a 2D subset of hexels.

2.1 6D Photo-Consistency

Suppose that the scene is imaged by the cameras $\mathbf{P}_i$. The image projection $\mathbf{u}_i = (u_i, v_i)$ of a scene point $\mathbf{x} = (x, y, z)$ by camera $\mathbf{P}_i$ is expressed by the relation $\mathbf{u}_i = \mathbf{P}_i(\mathbf{x})$. The images captured by the i^{th} camera at $t = 1$ and $t = 2$ are denoted $I_i^1(\mathbf{u}_i)$ and $I_i^2(\mathbf{u}_i)$ respectively.

A hexel $(x^1, y^1, z^1, x^2, y^2, z^2)$ is said to be *photo-consistent* if (x^1, y^1, z^1) and (x^2, y^2, z^2) project to pixels of approximately the same color in all of the images that they are visible in. Note that this definition of photo-consistency

is stronger than that introduced in [Kutulakos and Seitz, 1999] because it requires that the points have the same color: (1) from all viewpoints and (2) at both instants in time. The first requirement assumes a Lambertian surface model and the second assumes *brightness constancy*, a standard assumption for small scene motion.

Algebraically, we define hexel photo-consistency using the following measure:

$$\mathrm{Var}\left(\bigcup_{\mathrm{Vis}^1(\mathbf{x}^1)}\left\{I_i^1(\mathbf{P}_i(\mathbf{x}^1))\right\}\cup\bigcup_{\mathrm{Vis}^2(\mathbf{x}^2)}\left\{I_i^2(\mathbf{P}_i(\mathbf{x}^2))\right\}\right). \tag{3}$$

Here, $\mathrm{Var}(\cdot)$ is the variance of a set of numbers and $\mathrm{Vis}^t(\mathbf{x})$ is the set of cameras $\mathbf{P}_i$ for which $\mathbf{x}$ is visible at time t. Clearly, a smaller variance implies a greater likelihood that the hexel is photo-consistent.

Many other photo-consistency functions could be used instead, including robust measures. One particularly nice property of the variance, however, is that the 6D space-time photo-consistency function can be expressed as a combination of two 3D spatial photo-consistency functions. If we store the number of cameras $n^t = |\mathrm{Vis}^t(\mathbf{x}^t)|$, the sum of the intensities $S^t = \Sigma_i I_i^t(\mathbf{P}_i(\mathbf{x}^t)$, and the sum of the squares of the intensities $SS^t = \Sigma_i[I_i^t(\mathbf{P}_i(\mathbf{x}^t)]^2$ for the two spaces (x^t, y^t, z^t), $t = 1, 2$, then the photo-consistency of the hexel $(x^1, y^1, z^1, x^2, y^2, z^2)$ is:

$$\frac{SS^1 + SS^2 - (S^1 + S^2) * (S^1 + S^2)}{n^1 + n^2}. \tag{4}$$

Hence, the 6D photo-consistency measure in Equation (3) can be represented in memory as a collection of 3D functions, and can be quickly computed when needed from the stored representation using Equation (4).

2.2 6D Hexel Occupancy

The next step is to develop an algorithm to compute scene shape and flow from the photo-consistency function. In particular, we need to determine which hexels (see Figure 1) represent valid shape and flow; i.e., for which tuples $(x^1, y^1, z^1, x^2, y^2, z^2)$ there is a point (x^1, y^1, z^1) at time $t = 1$ that flows to (x^2, y^2, z^2) at time $t = 2$. Thus, the reconstruction problem can be posed as determining a binary hexel occupancy function in the discretized 6D space.

For each hexel there are therefore two possibilities; either it is reconstructed or it is carved. If it is reconstructed, the two voxels that it corresponds to are both reconstructed and the scene flow between them is computed. In this case, the hexel can be thought of as having a color; i.e., the color of the two voxels (which must be roughly the same if they are photo-consistent across time). If a hexel is carved however, it does not imply anything with regard to the occupancy of the two voxels; it just says that this particular match between the two is a bad one. For a voxel to be carved away, an entire 3D subspace of hexels (which corresponds to all possible flows for this voxel) has to be searched, and no valid match found.

We estimate the hexel occupancy function via a 6D carving algorithm; i.e., we initially assume that all of the hexels are occupied, meaning that no shape and flow hypotheses have been eliminated. We then remove any hexels that we can conclude are not part of the reconstruction. The result gives us an estimate of the scene shape at both time instants and the scene flow relating them.

Before describing our algorithm we briefly review the 3D carving algorithm of [Kutulakos and Seitz, 1999].

2.3 Review: Space Carving in 3D

In space carving the decision whether to carve a voxel is simple; a voxel is carved if its photo-consistency is above a threshold, otherwise the voxel is retained. The major contribution of the paper is its treatment of visibility. As can be seen from Equation (3), photo-consistency cannot be computed until the set of cameras that view the voxel is known. This visibility, in turn, depends on whether other voxels are carved or not. In space carving, decisions for the voxels are ordered in a way such that the decisions for all potential occluders are made before the decision for any voxel they may occlude.

A special case is when the scene and the cameras are separated by a plane. Then the carving decisions can be made in the correct order by sweeping the plane through the scene in the direction of the normal to the plane. To keep track of visibility, a collection of 1-bit masks are used, one for each input image. The masks keep track of which pixels are accounted for in the reconstruction. For each voxel considered, the color from a particular camera is only used if the mask at the pixel corresponding to the projection of the voxel has not already been set, implying that no other voxel along the line of sight is occupied. If the voxel is reconstructed (not carved) the pixels that it projects to are masked out in all of the cameras.

3 A 6D Slab Sweeping Algorithm

Our algorithm is a generalization of the 3D plane-sweep algorithm just described. It operates by sweeping a *slab* (a thickened plane) through the 3D volume of the scene. The slab is swept through the space simultaneously for both $t = 1$ and $t = 2$, as is illustrated in Figure 2. (In the figure, the slab sweeps from top to bottom.) For each position of the slab, all of the voxels on the top layer are considered and a decision made whether to carve them or not. We assume that the cameras and scene can be separated by a plane, so the shape and motion are simultaneously recovered at both times in a single pass sweeping the slab through the scene.

We also assume that there is an upper bound on the magnitude of the scene flow, and set the thickness of the slab to be this value. At any particular voxel, we therefore only

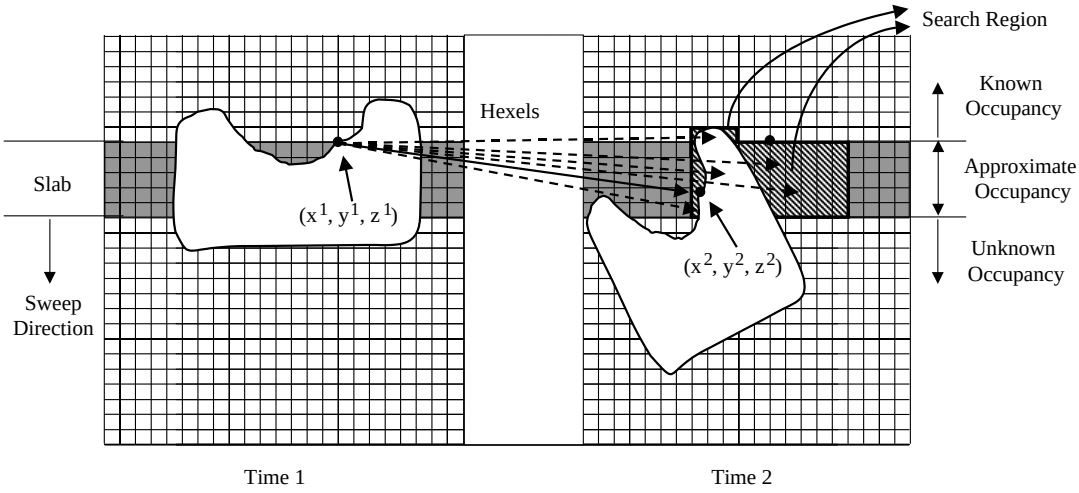


Figure 2: An illustration of the 6D Slab Sweeping Algorithm. A thick plane, or *slab*, is swept through the scene simultaneously for both $t = 1$ and $t = 2$. Occupancy decisions are made for each voxel on the top layer of the slab by searching the other time instant to see whether a matching voxel can be found. While the set of cameras visible to any voxel above the slab is known, the visibility of voxels in the slab (except for the top layer) needs to be approximated. See Section 3.1 for more details.

need to consider hexels for which the other endpoint is either in the slab, or above the slab by a distance less than the width of the slab. This saves us from having to consider the entire 3D subspace of hexels that have this voxel as an endpoint. We describe the algorithm for the voxel $\mathbf{x}^1 = (x^1, y^1, z^1)$ at $t = 1$, as is shown in the figure. The steps for $t = 2$ are of course the same, switching the roles of $t = 1$ and $t = 2$. For each position of the slab, we perform the following steps for each voxel in the top layer.

1. **Compute the visibility and color statistics:** The visibility of $\mathbf{x}^1$ is computed by projecting it into the cameras. If the pixel it projects to is already masked out (as in 3D space carving), $\mathbf{x}^1$ is occluded in that camera. Otherwise, the color of the pixel is incorporated into the color statistics n^1, S^1, and SS^1 needed to compute the photo-consistency using Equation 4.
2. **Determine the search region:** The search region is initialized to be a cube at the other time, centered on the corresponding voxel. The length, width, and height of the search region are all set equal to twice the maximum flow magnitude. This cube is then intersected with the known surface above the slab. (See Figure 2.) The search region defines a set of hexels to be searched, each of which corresponds to a possible hypothesis for the scene flow of $\mathbf{x}^1$.
3. **Compute all the hexel photo-consistencies:** The combination of $\mathbf{x}^1$ and each voxel in the search region corresponds to a hexel. For each such hexel, the visibilities and color statistics of the voxel being searched are computed and combined with those for $\mathbf{x}^1$ using Equation 4 to give the photo-consistency value for this hexel. (Computing the visibility for the voxel in the search region is non-trivial: if the voxel is above the slab, the visibility is known and was already computed when the voxel was in the top layer. But if the voxel is in the slab, it is impossible to compute the visibility without carving further into the scene. This step is involved and is described in Section 3.1.)
4. **Hexel carving decision:** The hexel with the best photo-consistency value is found. If its photo-consistency is above a threshold, all of the hexels in the search region are carved. This also means that the voxel $\mathbf{x}^1$ is carved. Otherwise, the best matching hexel is reconstructed, which means that $\mathbf{x}^1$ and the corresponding voxel at the other time are reconstructed and the scene flow between them computed. All of the other hexels are carved (a step which is equivalent to eliminating all of the other flow possibilities for $\mathbf{x}^1$.)
5. **Update the visibility masks:** If $\mathbf{x}^1$ was reconstructed (not carved) it is projected into all of the cameras. The masks at the projection locations are set. (The masks in the cameras for which $\mathbf{x}^1$ is not visible will already be set and so are not changed by this operation.)

3.1 Visibility Within the Slab

It is easy to show that it is impossible to determine the visibility below the top layer in the slab without first carving further into the slab.

Theorem 1 *Apart from the cameras which are occluded by the structure that has already been reconstructed above the slab, it is impossible to determine whether* **any** *of the other cameras are visible or not for voxels below the top layer*

(and that are not on the sides of the slab) until some occupancies in the top layer or below are known.

Proof: For any voxel below the top layer, if all of the voxels on the top layer and the sides of the slab turn out to be occupied, none of the cameras will be visible. On the other hand, if all of the voxels below the top layer turn out not to be occupied, all of the cameras that are not already occluded will be visible. □

Without making assumptions, such as that the visibility of a point does not change between its initial and its flowed position, we therefore have to carve into the slab to get an approximation of the visibility. We do this by performing a space carving in the slab, but with a high photo-consistency threshold. This gives us a thickened estimate of the surface because of the high threshold. A thickened surface will give an under-estimate of the visibility (at least on the surface) which is preferable[1] to an over-estimate of the visibility, which might arise if we chose a lower threshold and mistakenly carved away voxels on the true surface.

Space carving in the slab only defines the visibility on or above the thickened surface. To obtain estimates of the visibility below the thickened surface, we propagate the visibility down below the surface assuming that there are no local occlusions between the thickened surface and the true surface. This assumption is reasonable, either if the thickening of the surface caused by the high threshold carving is not too great, or if the range of motions (i.e., the size of the search region) is small relative to the size of local variations in the shape of the surface.

3.2 Properties of the Flow Field

The flow field produced by our algorithm will not, in general, be one-to-one (bijective). This property, however, is not desirable since it means that the shapes at the two times must contain the same number of voxels. Contractions and expansions of the surface may cause the "correct" discrete surfaces to have different numbers of voxels. Hence, bijectivity should not be enforced.

Ideally we want the flow field to be well defined for each visible surface voxel at both time instants. That is, the flow should be to a visible surface voxel at the other time. The algorithm described above does not guarantee that the flow field is well defined in this sense. It is possible for a voxel that is reconstructed in Step 4 to later appear in the top layer, but not be visible in any of the cameras. Such a voxel is not a surface voxel and should not have a flow.

To make sure that flow field is only defined between visible voxels on the two surfaces, we add a second pass to our algorithm. We perform the following two operations on any voxels that were reconstructed by Step 4 and which later turned out not to be visible in any of the cameras when they appeared in the top layer:

6. **Carve the hexel:** Since this interior voxel is one endpoint of a reconstructed hexel, that hexel (and this voxel) are carved.
7. **Find the next best hexel:** Find the corresponding voxel at the other time (i.e., the one at the other end of the hexel just carved) using the flow stored in this voxel. Repeat Steps 2–4 for that voxel to find the next best hexel (even if the photo-consistency of that hexel is above the threshold). Since the slab has passed through the entire scene once, the search region can be limited to voxels known to be on the surface.

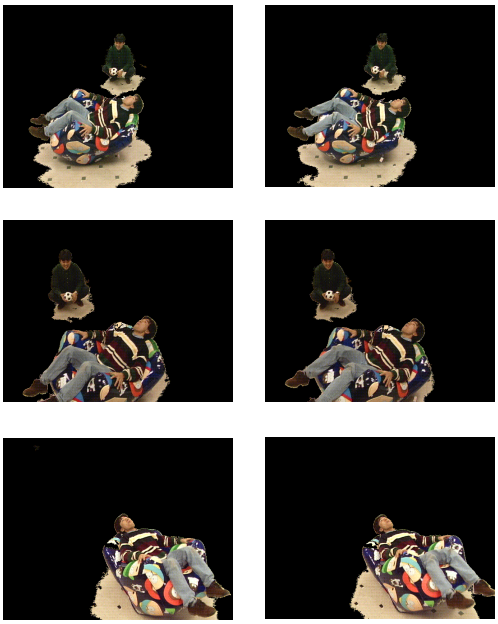
With this second pass, our algorithm does guarantee the property that the flow field is only defined for surface voxels. Note that guaranteeing this property is only possible in an efficient manner by assuming that if the best matching hexel in Step 4 turns out to have an endpoint that is not on the surface, then another hexel can be found in Step 7 that is also photo-consistent (and for which the other endpoint is a surface voxel). This assumption is reasonable since the other endpoint of the best matching hexel will likely be close to the surface to start with, and because the photo-consistency function should be spatially continuous.

This two-pass procedure may also be used to model any inherent ambiguities in the motion field, such as the *aperture problem*. For example, the aperture problem manifests itself in the form of multiple hexels passing the photo-consistency threshold test in Step 4 of the algorithm. Since all of these hexels yield photo-consistent hypotheses for the flow, the flow cannot be estimated without motion (smoothness) constraints. In our carving algorithm we choose the most photo-consistent hexel to yield a unique flow at every reconstructed voxel, followed by simple spatial averaging over a small volume (3x3x3). Further investigation of how to impose smoothness in the motion field, and in carving algorithms in general, is left as future work.

4 Experimental Results

We tested our 6D carving algorithm on a collection of 14 image pairs (one at $t = 1$, the other at $t = 2$). The image pairs are taken from different cameras viewing a dynamic event consisting of a man falling backwards in a chair. The input image pairs from 3 of the 14 cameras are shown for both time instants (after a simple background subtraction step) in Figure 3. The motion is very approximately a rotation, except for the legs (which are moving outward somewhat faster than the rest of the scene). The hexel grid was modeled as the Cartesian product of two 3D voxel grids of size 150x150x75. It therefore contains 150^4x$75^2 \approx 2.8 \times 10^{12}$ hexels. The maximum flow was set to be 8 voxels (corresponding to 20cm in the scene). Therefore the search region in Step 4 of the algorithm had a max-

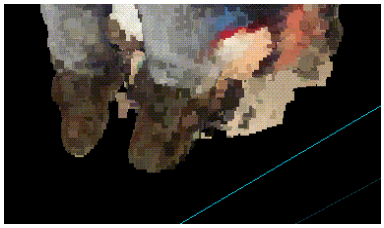
[1] An under-estimate of the visibility is preferable to an over-estimate because under-estimates cannot lead to false carving decisions. On the other hand, mistakenly using a camera could lead to the addition of an outlier color and the carving of a hexel in the correct reconstruction.



Time 1 Time 2

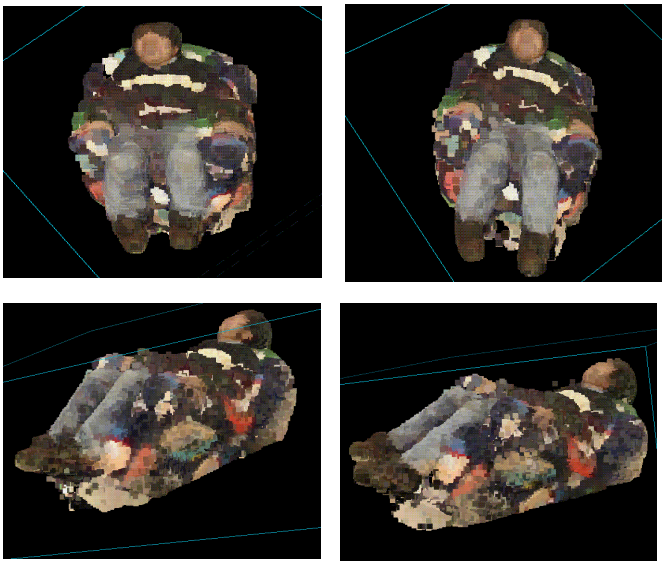
Figure 3: Input sequence: Person falling backwards while sitting on a chair (640x480 images, background subtraction used to approximately segment foreground)

(a) Close up of shape reconstructed by Space Carving



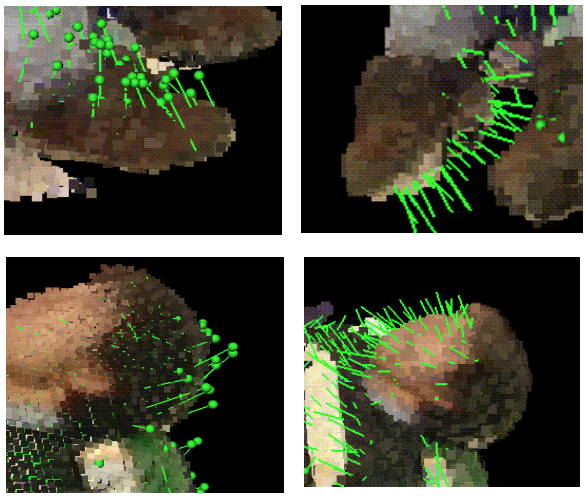
(b) Close up of shape reconstructed by 6D Carving

Figure 5: Comparison with 3D Space Carving. The 6D spatio-temporal constraints eliminate artifacts near the feet, and also remove other stray voxels that don't have a consistent motion.



Time 1 Time 2

Figure 4: Reconstructions obtained using 6D Shape and motion carving. Both shapes are recovered simultaneaously using the 6D slab sweeping algorithm



Time 1 Time 2

Figure 6: Examples of the computed scene flow, shown as needle maps with a sphere drawn at the tip of the flow vector. The backward motion of the head, and upward motion of the right foot are captured by the scene flow vectors.

imum size of 16x16x16 voxels. This reduces the number of hexels considered to 150^2x75x$16^3 \approx 6.9 \times 10^9$.

Figure 4 shows two views of the shapes obtained by the 6D carving algorithm at the two time instants. The shapes are obtained simultaneously after a 6D slab sweep, and a second pass to enforce the surface flow properties described in the previous section. (The reconstruction consists of approximately $60,000$ hexels. The running time was 20mins on an R10000 SGI O2 and uses 250MB of memory.) Figure 5 compares the reconstruction obtained by 3D Space Carving, with that obtained by our algorithm on a close-up of the right foot of the person. Since the foot is moving upwards, the additional spatio-temporal constraints in the 6D carving algorithm remove some of the artifacts. In addition, a number of the outlier voxels that are reconstructed by space carving are correctly carved by our algorithm because they do not have a consistent motion.

Figure 6 shows the computed scene flow. The scene flow vectors are displayed as needle maps with a sphere drawn at the $t = 2$ endpoint of each vector. For clarity, these are only displayed at a small subset of the voxels. Looking at the direction of the vectors and their position on the shapes at both time instants, it can be clearly seen that they capture the upward motion of the right foot and the backward motion of the head.

5 Discussion

We have described an algorithm for the simultaneous computation of scene shape and scene flow across two time instants, which operates by carving hexels in the 6D space of shapes and flows. A hexel is carved if the two points that it corresponds to project to image pixels that are not jointly photo-consistent across both times. We have shown that incorporating these spatio-temporal constraints can improve the reconstructed shape.

One major weakness that our algorithm shares with 3D space carving is that it does not guarantee that the recovered shape is continuous. In addition, because the flow is estimated individually for every voxel, the flow field is not as smooth as it would be otherwise. We are currently investigating which motion conditions provide the most additional information for improving shape. We are also interested in ways of enforcing smoothness by evolving local 2D representations of the scene under forces derived from the 6D photo-consistency measure, in a similar manner to that described in [Faugeras and Keriven, 1998].

References

[Bregler and Malik, 1998] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *CVPR '98*, pages 8–15, 1998.

[Carceroni and Kutulakos, 1999] R.L. Carceroni and K.N. Kutulakos. Multi-view 3D shape and motion recovery on the spatio-temporal curve manifold. In *7th ICCV*, 1999.

[Chen and Medioni, 1999] Q. Chen and G. Medioni. A volumetric stereo matching method: Applications to image-based modeling. In *CVPR '99*, 1999.

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

[De Bonet and Viola, 1999] J.S. De Bonet and P. Viola. Roxels: Responsibility weighted 3D volume reconstruction. In *7th ICCV*, 1999.

[Faugeras and Keriven, 1998] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods, and the stereo problem. *Trans. on Image Processing*, 7(3):336–344, 1998.

[Guenter *et al.*, 1998] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. *SIGGRAPH 98*, pages 55–66, 1998.

[Kutulakos and Seitz, 1999] K. Kutulakos and S. Seitz. A theory of shape by space carving. In *7th ICCV*, 1999.

[Marr and Poggio, 1979] D. C. Marr and T. Poggio. A computational theory of human stereo vision. *Proc. of the Royal Soc. of London*, B 204:301–328, 1979.

[Metaxas and Terzopoulos, 1993] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation thr-ough physics-based synthesis. *PAMI*, 15(6), 1993.

[Pentland and Horowitz, 1991] A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *PAMI*, 13(7):730–742, 1991.

[Roy and Cox, 1998] S. Roy and I.J. Cox. A maximum-flow formulation of the *n*-camera stereo correspondence problem. In *Sixth ICCV*, pages 492–499, 1998.

[Scharstein and Szeliski, 1998] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *IJCV*, 28(2):155–174, 1998.

[Seitz and Dyer, 1999] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by space coloring. *IJCV*, 35(2):1–23, 1999.

[Vedula *et al.*, 1999] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *7th ICCV*, 1999.

[Waxman and Duncan, 1986] A. Waxman and J. Duncan. Binocular image flows: Steps toward stereo-motion fusion. *PAMI*, 8(6):715–729, 1986.

[Young and Chellappa, 1999] G.S. Young and R. Chellappa. 3-D motion estimation using a sequence of noisy stereo images: Models, estimation, and uniqueness. *PAMI*, 12(8):735–759, 1999.

[Zhang and Faugeras, 1992] Z. Zhang and O. Faugeras. Estimation of displacements from two 3-D frames obtained from stereo. *PAMI*, 14(12):1141–1156, 1992.