Space-time Isosurface Evolution for Temporally Coherent 3D Reconstruction

Bastian Goldluecke Marcus Magnor
Max-Planck-Institut für Informatik
Graphics-Optics-Vision
Saarbrücken, Germany
{bg,magnor}@mpii.de

Abstract

We model the dynamic geometry of a time-varying scene as a 3D isosurface in space-time. The intersection of the isosurface with planes of constant time yields the geometry at a single time instant. An optimal fit of our model to multiple video sequences is defined as the minimum of an energy functional. This functional is given by an integral over the entire hypersurface, which is designed to optimize photo-consistency. A PDE-based evolution derived from the Euler-Lagrange equation maximizes consistency with all of the given video data simultaneously. The result is a 3D model of the scene which varies smoothly over time. The geometry reconstructed by this scheme is significantly better than results obtained by space-carving approaches that do not enforce temporal coherence.

1. Introduction

Our goal is to reconstruct temporally coherent geometry using multi-video data from only a handful of cameras distributed around a scene. The geometry models obtained that way enable us to render the dynamic scene from arbitrary viewpoints in high quality, using image-based rendering techniques we investigated earlier [6]. In this paper, however, we focus on the aspect of spatio-temporal reconstruction, a computer vision problem that is a primary focus of research interest. It has been approached from several points of view.

For one single frame of multi-camera data, voxel models of the scene have been quite popular for some time. In particular, the visual hull can be easily and quickly be computed as an approximate conservative model of scene geometry [9], and has been widely used as a geometry proxy for image-based rendering. Much refined voxel models can be obtained using space-carving [8], where one starts with a completely solid voxel model, and carves away regions of bad photo-consistency. Vedula et al. [13] achieve temporal coherence by introducing a 6D model which includes motion information about each voxel, and additionally carving away voxels inconsistent with the estimated scene flow.

Level set models are a nice alternative to voxels. The boolean function defined on the voxel grid marking a voxel as occupied or empty may be viewed as a $\{0, 1\}$-valued function whose 0.5 level set is the surface enclosing the occupied voxels. In that sense, level sets are a true generalization of voxel models. Techniques which naturally employ level set models are those based on *weighted minimal surfaces*, which minimize an energy functional given as a surface integral of a scalar valued weight function. The variational formulation of these kind of problems leads to a surface evolution PDE which can be implemented using level set techniques. Faugeras and Keriven [4] analyzed how minimal surfaces can be employed for 3D reconstruction of static scenes from multiple views. This technique was recently extended to simultaneously estimate the radiance of surfaces, and demonstrated to give good results in practice [7]. Another well-known technique which utilizes minimal surfaces is *Geodesic Active Contours* [1]. While originally designed for segmentation in 2D, it quickly became clear that it could be generalized to 3D [2], and also applied to other tasks. It is particularly attractive for modeling surfaces from point clouds [14]. Geodesic contours have also been employed for 2D detection and tracking of moving objects [11].

In [5], we gave a mathematical analysis of weighted minimal hypersurfaces in arbitrary dimension and for a general class of weight functions. We derived the Euler-Lagrange equation yielding a necessary minimality condition. Our analysis covers all of the the variational methods mentioned above. In this paper, we present a variational method of a new kind, applying the freedom in dimensionality allowed by the theorem we proved in [5]. A fourth dimension is introduced which represents the flow of time in the

video sequence. Our goal is to reconstruct a smooth three-dimensional hypersurface embedded in space-time. The intersections of this hypersurface with planes of constant time are two-dimensional surfaces, which represent the geometry of the scene in a single time instant. Our approach defines an energy functional for the hypersurface. The minimum of the functional is the geometry which optimizes photo-consistency as well as temporal smoothness.

In Sect. 2, we will introduce the mathematical foundations of the algorithm and give a rigorous definition of our method in terms of an energy minimization problem. We also review how the minimization can be performed as a surface evolution implemented with a level set method. Implementation details are discussed in Sect. 3, where we describe our parallel scheme which computes the evolution equation using a narrow band level set method. We also propose algorithms necessary to evaluate the more involved terms of the equation. Results obtained with real-world video data are presented in Sect. 4.

2. Space-time 3D Reconstruction

In this section, we present the mathematical foundations of our 3D reconstruction algorithm. We assume that we have a set of fully calibrated, fixed cameras. The input to our algorithm are the projection matrices for the set of cameras, as well as a video stream for each camera. We want to obtain a smooth surface Σ_t for each time instant t, representing the geometry of the scene at that point in time. The surfaces shall be as consistent as possible with the given video data. Furthermore, as in reality, all resulting surfaces should change smoothly over time.

2.1. Mathematical Foundations

To achieve these desirable properties, we do not consider each frame of the sequences individually. Instead, we regard all two-dimensional surfaces Σ_t to be subsets of one smooth three-dimensional hypersurface $\mathfrak{H}$ embedded in four-dimensional space-time. From this viewpoint, the reconstructed surfaces

$$\Sigma_t \;=\; \mathfrak{H} \cap \left(\mathbb{R}^3, t\right) \subset \mathbb{R}^3$$

are the intersections of $\mathfrak{H}$ with planes of constant time. Because we reconstruct only one single surface for all frames, the temporal smoothness is intrinsic to our method.

However, we have to take care of photo-consistency of the reconstructed geometry with the given image sequences. We minimize an energy functional

$$\mathcal{A}\left(\mathfrak{H}\right) \;:=\; \int_{\mathfrak{H}} \Phi \; dA. \tag{1}$$

defined as an integral of the scalar valued weight function Φ over the whole hypersurface. $\Phi = \Phi(s, \mathbf{n})$ measures the photo-consistency error density, and may depend on the surface point s and the normal $\mathbf{n}$ at this point. In [5], we employed a mathematical tool known as the *method of the moving frame* in order to prove the following theorem which is valid in arbitrary dimension.

Theorem. A k-dimensional surface $\mathfrak{H} \subset \mathbb{R}^{k+1}$ which minimizes the functional $\mathcal{A}\left(\mathfrak{H}\right) := \int_\Sigma \Phi\left(s, \mathbf{n}(s)\right)\, dA(s)$ satisfies the Euler-Lagrange equation

$$\langle \Phi_s, \mathbf{n} \rangle \;-\; \mathrm{Tr}\left(\mathbf{S}\right)\Phi \;+\; \mathrm{div}_{\mathfrak{H}}(\Phi_{\mathbf{n}}) \;=\; 0, \tag{2}$$

where $\mathbf{S}$ is the shape operator of the surface, also known as the Weingarten map or second fundamental tensor. We now present suitable choices for the error measure Φ.

2.2. Continuous Space-time Carving

We need some additional notation for color and visibility of points in space-time first. Let t denote a time instant, then a time-dependent image I_k^t is associated to each camera k. The camera projects the scene onto the image plane via a fixed projection $\pi_k : \mathbb{R}^3 \to \mathbb{R}^2$. We can then compute the color c_k^t of every point (s, t) on the hypersurface:

$$c_k^t(s) = I_k^t \circ \pi_k(s).$$

Here, the image I_k^t is regarded as a mapping assigning color values to points in the image plane.

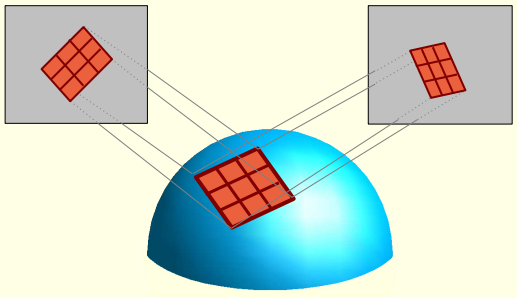
In the presence of the surface Σ_t, let $\nu_k^t(s)$ denote whether or not s is visible in camera k at time t. $\nu_k^t(s)$ is defined to be one if s is visible, and zero otherwise.

The most basic error measure can now be defined as

$$\Phi^S(s,t) \;:=\; \frac{1}{V_{s,t}} \sum_{i,j=1}^{l} \nu_i^t(s)\nu_j^t(s) \cdot \left\| c_i^t(s) - c_j^t(s) \right\|.$$

The number $V_{s,t}$ of pairs of cameras able to see the point s at time t is used to normalize the function.

If the error function Φ^S is used as the functional, the resulting algorithm is similar to a space carving scheme in each single time step. In that method, as introduced by Kutulakos and Seitz [8], voxels in a discrete voxel grid are carved away if Φ^S lies above a certain threshold value when averaged over the voxel. In our scheme, the discrete voxels are replaced by a continous surface. In the surface evolution introduced later, this surface will move inwards until photo-consistency is achieved. This process is analogous to the carving process [8]. The same functional for regular surfaces in $\mathbb{R}^3$ was introduced by Faugeras and Keriven [4] for static scene reconstruction. As an additional constraint, we enforce temporal coherence in the form of temporal smoothness of the resulting hypersurface, which makes our method ideal for video sequences.



A small grid in the tangent plane is projected in both camera images. This leads to two columns of data containing the pixels colors of the grid corners in the respective video image. Between these two columns of data, the normalized cross correlation is computed.

Figure 1. *Cross-correlation error term* Φ^G.

2.3. Normal Optimization

Because the theorem also allows for error functions which may depend on the normal, we can take the scheme one step further to include an optimization for the surface normals as well. A similar idea was also presented in [4], however, we give a slightly modified version and still work in space-time to enforce temporal smoothness.

In order to set up an error function, we have to analyze how well a small surface patch at position s with a given normal $\mathbf{n}$ fits the images at time t. To this end, we assign to each of these values a small patch $\square_{s,\mathbf{n}}$ within the plane orthogonal to $\mathbf{n}$, Fig. 1. How exactly this patch is chosen does not matter, however, the choice should be consistent over time and space and satisfy a few conditions which will become evident soon. In our implementation, we always choose rectangular patches rotated into the target plane by a well-defined rotation.

We will now define a measure how well the patch $\square_{s,\mathbf{n}}$ is in accordance with the images at time t. For that, we employ the normalized cross-correlation of corresponding pixels in the images, a well-established matching criterion in computer vision. Mathematically, the resulting functional for a point $x = (s,t) \in \mathbb{R}^4$ with normal direction $\mathbf{n}$ is defined as follows:

$$\Phi^G(x,\mathbf{n}) \;:=\; -\frac{1}{V_{s,t}} \sum_{i,j=1}^{l} \nu_i^t(s)\nu_j^t(s) \cdot \frac{\chi_{i,j}^t(s,\mathbf{n})}{A\left(\square_{s,\mathbf{n}}\right)}$$

with the zero-mean cross-correlation

$$\chi_{i,j}^t(s,\mathbf{n}^t) \;:=\; \int_{\square_{s,\mathbf{n}^t}} \left(c_i^t - \overline{I}_i^{x,\mathbf{n}}\right)\left(c_j^t - \overline{I}_j^{x,\mathbf{n}}\right)\, dA,$$

and the mean color value of the projected patch in the images computed according to

$$\overline{I}_i^{x,\mathbf{n}} \;:=\; \frac{1}{A\left(\square_{s,\mathbf{n}}\right)} \int c_i^t \, dA.$$

Some things have to be clarified. First, the correlation measure $\chi_{i,j}^t$ for a pair of cameras is normalized using the area $A\left(\square_{s,\mathbf{n}}\right)$ of the patch. Second, it is now clear that we have to choose $\square_{s,\mathbf{n}}$ sufficiently large so that it is projected onto several pixels. On the other hand, it should not be so large that only parts of it are visible in the images. As a compromise, we choose its diameter to be equal to the cell diameter of the underlying computation grid, as defined in Sect. 3. Third, the integration of Φ^G in the energy functional involves the normals of $\mathfrak{H}$ in 4D space, while $\mathbf{n}$ is supposed to lie in $\mathbb{R}^3$. For that reason, we project normals of $\mathfrak{H}$ into the tangent space of Σ_t in order to get $\mathbf{n}$.

When this functional is minimized, two constraints are optimized simultaneously. Each surface Σ_t together with its normals is selected to best match the images at that time instant. Furthermore, a smooth change of the surfaces Σ_t with time is encouraged because of the curvature term in the Euler-Lagrange equation. The error functional can be minimized using a surface evolution implemented via a level set scheme, as derived in the next subsection.

2.4. Level Set Evolution Equation

In order to find the minimum of the energy functional, we have to find a solution to the Euler-Lagrange equation (2) according to our theorem. An efficient way to do this is to rewrite it as a surface evolution which can be implemented using level sets [10, 3]. This technique is well-established for a wide area of applications [12].

In the level set framework, we consider a familiy of surfaces $\mathfrak{H}_\tau$ represented as the zero level sets of a regular function

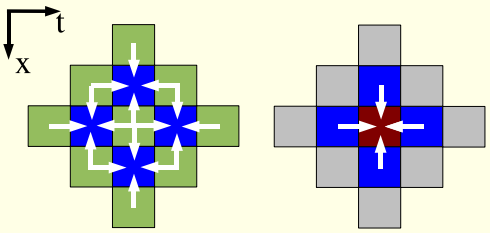
$$u : \mathbb{R}^4 \times \mathbb{R}^{\geq 0} \to \mathbb{R}, \quad u(x,\tau) = 0 \;\Leftrightarrow\; x \in \mathfrak{H}_\tau. \qquad (3)$$

$u(\cdot,\tau)$ is required to be positive inside the volume enclosed by $\mathfrak{H}_\tau$, and negative on the outside. We derived in detail in [5] that when exposed to the evolution equation

$$\begin{aligned} \frac{\partial}{\partial\tau} u \;&=\; \Psi\, |\nabla u| \\ \text{with} \qquad \Psi \;&:=\; \left[-\operatorname{div}\left(\Phi \cdot \frac{\nabla u}{|\nabla u|} \right) \;+\; \operatorname{div}_\Sigma(\Phi_{\mathbf{n}}) \right], \end{aligned} \qquad (4)$$

the surfaces $\mathfrak{H}_\tau$ converge towards a local minimum of the Euler-Lagrange equation if we start with a suitable initial surface $\mathfrak{H}_0$.

In the next section, we will analyze how this evolution can be implemented efficiently in a multi-processor environment.



Step 1 *Step 2*

- In the first step, the values of u_i in the green cells are used to compute the level set normal $\mathbf{n} \in \mathbb{R}^4$ in the blue cells using central differences. Having computed $\mathbf{n}$, we can also compute Φ for the blue cells. Note that for the purpose of the above 2D illustration, the three spatial dimensions are represented as one.
- For the second step, we compute the values for the central red cell, also using finite differences. The discrete formula for $\mathrm{div}(\Phi\mathbf{n})$ at position $p = (x, y, z, t)$ is

$$\sum_{i=1}^{4} \frac{\Phi^{p+e_i} n_i^{p+e_i} - \Phi^{p-e_i} n_i^{p-e_i}}{2}.$$

  We can also compute the curvature $\mathrm{Tr}\,(\mathbf{S})$ directly by omitting Φ in the above formula.
- The difficult part is to compute $\mathrm{div}_\Sigma(\Phi_\mathbf{n})$ for the red cell. It is equal to the trace of $\Phi_{\mathbf{n}s}$ restricted to the tangent plane Π orthogonal to the normal at p. So we first compute $\Phi_\mathbf{n}$ for the blue cells using finite differences, taking the known normal $\mathbf{n}$ of the cell as the center point. With these values, we can set up the 4×4 matrix $U := \Phi_{\mathbf{n}s}$ for the red cell. Choose an arbitrary orthonormal base $\{t_0, t_1, t_2\}$ of the plane Π. The entries for the 3×3 matrix V of the mapping $\Phi_{\mathbf{n}s}|_\Pi$ can then be computed as

$$v_{ij} \;=\; t_i^T U t_j, 1 \le i, j \le 3.$$

Figure 2. ***Evaluation of the differential operator.***

3. Parallel Implementation

In order to implement the level set evolution equation, the volume surrounding the hypersurface $\mathfrak{H}$ has to be discretized. We use a regular four-dimensional grid of evenly distributed cells with variable spatial resolution of usually 64^3 or 128^3 cells. The temporal resolution is naturally equal to the number of frames in the input video sequences. It is currently not yet possible to store the full data for each grid cell together with all images of all video sequences within the main memory of a standard PC. A parallel implementation where the workload and data is distributed over several computers is therefore mandatory.

For that reason, we choose the narrow band level set method [12] to implement the evolution equation because it is straightforward to parallelize. We start with an initial surface $\mathfrak{H}_0$ and the values u_0^{xyzt} of the corresponding level set function u_0 in the centers of the grid cells. A suitable initial surface for our case will be defined at the end of this section. The values of the level set function are updated iteratively using the upwind scheme. At iteration step $i + 1$, the new values u_{i+1}^{xyzt} are obtained from the values u_i^{xyzt} of the previous iteration step by a discrete version of equation (4) using an explicit time step:

$$u_{i+1}^{xyzt} \;=\; u_i^{xyzt} \;+\; \Psi\left(u_i^{xyzt}\right) |\nabla u_i| \cdot \Delta\tau. \qquad (5)$$

Here, $\Psi\left(u_i^{xyzt}\right)$ is the value of the discretized version of the differential operator Ψ acting on u_i, evaluated in the cell (x, y, z, t). Central differences on the four-dimensional grid are used to compute the derivatives involved in Eqn. 4. The norm of the discretized gradient $|\nabla u_i|$ is calculated according to the upwind scheme [12]. To ensure stability, the step size $\Delta\tau$ must be chosen such that the level sets of u_i cannot cross more than one cell at a time, i.e. satisfy the CFL-condition

$$\Delta\tau \;\le\; \max_{(x,y,z,t)\in\Gamma} \left(\frac{\text{diam cell}(x, y, z, t)}{\left|\Psi\left(u_i^{xyzt}\right) \cdot \nabla u\right|} \right). \qquad (6)$$

The computation of individual cells can easily be distributed over several processes. In our implementation, each process is responsible for the computation of one single slice of the grid of constant time t_i. This slice corresponds to the geometry of the ith frame of the video sequence. Fig. 2 shows in more detail how the value $\Psi\left(u_i^{xyzt}\right)$ is numerically evaluated from the values of u_i in the grid cells. According to this figure, we need the values of grid cells up to two cells apart from (x, y, z, t) in order to evaluate the operator. As a consequence, each process P_i also has to know the slices of the four other processes $P_{i\pm1}, P_{i\pm2}$. These have to be communicated over the network. In addition, each process needs to store the image data of its own video frame and the two adjacent frames according to Fig. 2.

To summarize, one full iteration consists of the following four steps:

- Each process transmits its own slice S_i to the adjacent processes and receives the other necessary slices from its four neighbours.
- Afterwards, each process computes $\Psi\left(u_i^{xyzt}\right)$ for all cells in its slice near the zero level set of u_i, using the scheme presented in Fig. 2.
- The maximum value of the operator for each process is transmitted to a special server process. From these maxima, the server calculates the optimal step size $\Delta\tau$ allowed by the inequality (6).
- The server broadcasts the maximum to all processes, which afterwards compute the evolution on their slice using equation (5).

Grid res.	# procs.	Time per iteration [s]		Memory
		without n.o.	with n.o.	per proc.
32^3	60	0.9	25	80 MB
	40	1.4	38	
	20	2.5	60	
64^3	60	7	140	180 MB
	40	11	210	
	20	17	360	
128^3	60	30	510	535 MB
	40	55	840	
	20	102	1200	

Table 1. ***Time and memory requirements for one single iteration, with and without normal optimization, depending on resolution and number of processes.***

We finally have to define a suitable initial surface $\mathfrak{H}_0$ to start the iteration process. For this purpose, we employ the visual hull, which by definition is always a superset of the correct scene geometry. In order to compute a level set representation, we have to choose suitable values of u_0 for each grid cell. For this purpose, we fix a grid cell c and select a number of evenly distributed sample points $x_0, \ldots, x_k$ inside it. These points are projected into each source image, and we compute the percentage $p \in [0, 1]$ of the projections which fall into the silhouettes of the object to be reconstructed. To the initial level set function u_0 is then assigned the value $2p - 1$ at cell c. Since we only have to compute an approximate starting surface, this straightforward method gives sufficiently good results in practice. In particular, the projection of the zero level set of u_0 into the source images very closely resembles the silhoettes of the object if k is sufficiently high.

4. Results

In order to test our algorithm, we run it on real-world 320×240 RGB video sequences of a ballet dancer. All input images are segmented into foreground and background using a thresholding technique. Consequently, we can compute the refined visual hull to get a starting volume for the PDE evolution, Fig. 3. For our test runs, we choose a 20 frame long part of the sequence with the depicted frame in the middle. As becomes apparent in Fig. 4, this frame is particularly difficult to reconstruct, because we do not have a camera capturing the scene from above. For that reason, most of the area in between the arms of the dancer is not carved away in the initial surface.

When we run a standard space-carving algorithm for this single frame alone, the situation improves. The shirt of the dancer contains not much texture information, however, so only part of the critical region is carved away as it should be. Only when we employ the full algorithm which takes into account temporal coherence between the geometry of the frames do we get the satisfactory result in Fig. 4 on the right.

Table 1 informs about the time and memory required by each of the slave processes for a single iteration. Our program ran on a Sun Fire 15K with 75 UltraSPARC III+ processors at 900 MHz, featuring 176 GBytes of main memory. It can be observed that the normal optimization requires a lot of computation time when compared to the standard version of our algorithm. For that reason, we first let the geometry evolve towards a surface which is very close to an optimal result, as assessed by the operator of the program. Afterwards, we switch on the normal optimization in order to improve the reconstruction of small surface details. In average, we need around one hundred iterations of the initial evolution and twenty more of the normal optimization until the surface has converged to the final result.

5. Summary and Conclusions

We have presented a novel 3D reconstruction algorithm which takes into account all frames of a multi-video sequence simultaneously. The idea is to optimize photo-consistency with all given data as well as temporal smoothness. Our method is formulated as a weighted minimal surface problem posed for a 3D hypersurface in space-time. Intersecting this hypersurface with planes of constant time gives the 2D surface geometry in each single time instant. The energy functional defining the minimization problem enforces photo-consistency, while temporal smoothness is intrinsic to our method. The functional can be minimized by implementing a surface evolution PDE using the narrow band level set method. Significant improments compared to space carving approaches which do not take temporal coherence into account can be observed in the results. In the future, we plan to include a global optimization of surface reflectance properties into the same unifying framework.

References

[1] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *Proc. International Conference on Computer Vision*, pages 694–699, 1995.

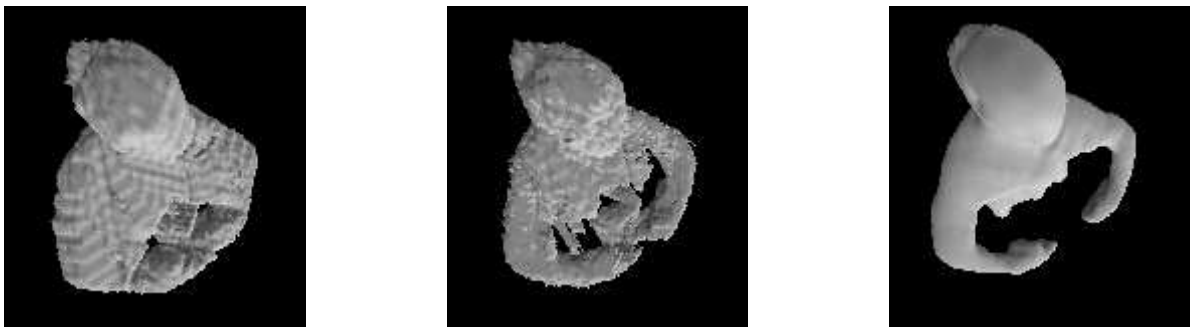
[2] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert. Three dimensional object modeling via minimal surfaces. In *Proc. European Conference on Computer Vision*, volume 1, pages 97–106. Springer, Apr. 1996.

[3] D. Chop. Computing minimal surfaces via level set curvature fbw. *Journal of Computational Physics*, 106:77–91, 1993.

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

(a) Source silhouettes (b) Visual hull (c) Final result

Figure 3. ***(a) Source silhouettes from the cameras, single frame. (b) Refined visual hull, representing the initial surface. (c) Final result after running the complete algorithm.***



(a) Visual hull (b) Space carving result (c) Our result

Figure 4. ***Comparison of different reconstruction schemes.*** **(a) Visual hull. (b) The result obtained from static space carving. The difficult geometry between the arms is slightly improved. (c) An almost optimal reconstruction is achieved by our algorithm.**

[5] B. Goldluecke and M. Magnor. Weighted minimal hypersurfaces and their applications in computer vision. To appear in *Proceedings of ECCV 2004*.

[6] B. Goldluecke and M. Magnor. Real-time microfacet billboarding for free-viewpoint video rendering. In *International Conference on Image Processing*, pages ?–?, Barcelona, Spain, September 2003.

[7] H. Jin, S. Soatto, and A. J. Yezzi. Multi-view stereo beyond Lambert. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 171–178, Madison, Wisconsin, USA, June 2003.

[8] K. N. Kutukalos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):197–216, July 2000.

[9] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Recognition*, 16(2):150–162, Feb. 1994.

[10] S. Osher and J. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on the Hamilton-Jacobi formulation. *Journal of Computational Physics*, 79:12–49, 1988.

[11] N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):266–280, 2000.

[12] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Monographs on Applied and Computational Mathematics. Cambridge University Press, 2nd edition, 1999.

[13] S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6D. In *Proc. Computer Vision and Pattern Recognition*, volume 2, pages 592–598, 2000.

[14] H. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. *1st IEEE Workshop on Variational and Level Set Methods, 8th ICCV*, 80(3):194–202, 2001.