Spacetime-coherent Geometry Reconstruction from Multiple Video Streams

Marcus Magnor and Bastian Goldlücke
MPI Informatik
Saarbrücken, Germany
magnor@mpi-sb.mpg.de

Abstract

By reconstructing time-varying geometry one frame at a time, one ignores the continuity of natural motion, wasting useful information about the underlying video-image formation process and taking into account temporally discontinuous reconstruction results. In 4D spacetime, the surface of a dynamic object describes a continuous 3D hyper-surface. This hyper-surface can be implicitly defined as the minimum of an energy functional designed to optimize photo-consistency. Based on an Euler-Lagrange reformulation of the problem, we find this hyper-surface from a handful of synchronized video recordings. The resulting object geometry varies smoothly over time, and intermittently invisible object regions are correctly interpolated from previously and/or future frames.

1. Introduction

How can the time-varying geometry of a dynamic natural scene be captured ? Currently, multi-video footage is the only financially and logistically feasible way of recording dynamic events by which shape information is (at least implicitly) recorded. To recover the time-varying shape from multi-video recordings, however, is a challenging task. The traditional approach is to regard the video frames sequentially, reconstructing one 3D geometry model per time step. However, this procedure potentially introduces temporal discontinuities in the evolving geometry which may show up annoyingly during playback.

Real-world objects vary continuously over time. By incorporating this inherent trait into the reconstruction algorithm, object geometry is recovered more robustly, and temporarily occluded regions can be correctly interpolated. To do so, the video image sequences must be processed globally, regarding 3D scene evolution in 4D spacetime. In this paper, we describe a mathematically sound approach to reconstruct such temporally continuous object shape from only a handful of video cameras distributed around the scene.

In the following Section, we review previous work on dynamic 3D reconstruction. In Sect. 3, we introduce the mathematical foundations of the algorithm and give a rigorous definition of our method in terms of an energy minimization problem. Sect. 4 demonstrates how the minimal hyper-surface can be found by surface evolution. Implementation details are discussed in Sect. 5 where we describe a parallelized scheme which computes the surface evolution equation using a narrow-band level-set method. Results obtained from real-world video data are shown in Sect. 6 before we wrap up the our presentation in Sect. 7.

2. Related Work

For static scenes, a number of approaches have been devised for reconstructing 3D geometry from photographs [17]. The visual hull has been widely used as a geometry proxy because it can be computed efficiently and represents an approximate, conservative model of object geometry [12, 13]. More refined voxel models can be obtained using space-carving [11] and space coloring [18] which take local photo-consistency into account. Vedula et al. [21] extend the space-carving approach by additionally estimating the scene flow to determine local motion for each voxel.

Level set models are a useful alternative to voxel representations. 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 this 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 variation-al formulation of these kinds of problems leads to a surface evolution-PDE which can be implemented using level set techniques. Faugeras and Keriven [6] analyzed how minimal surfaces can be employed for 3D reconstruction of static scenes from multiple views. This technique was recently extended to simultaneously estimate geometry and surface reflectance [9]. Another well-known technique which utilizes minimal surfaces is known as *Geodesic Active Contours* [3]. Originally designed for segmentation in 2D [10], it can be generalized to three dimensions [4] and is particularly attractive for modeling surfaces from point clouds [25, 24]. Geodesic contours have also been employed for 2D detection and tracking of moving objects [16].

Our reconstruction approach is based on finding a 3D hyper-surface in 4D spacetime that minimizes an energy functional. In [8], we give a mathematical analysis of weighted minimal hyper-surfaces in arbitrary dimension and for a general class of weight functions. We derive the Euler-Lagrange equation yielding a necessary minimality condition. This analysis also covers all of the the variational methods mentioned above. Here, we present a specific application of this theoretical work [7]. To recover smoothly varying geometry, we introduce a fourth dimension which represents the flow of time of the 3D scene. Our goal is to reconstruct a smooth three-dimensional hyper-surface embedded in spacetime. The intersections of this hyper-surface with planes of constant time are two-dimensional surfaces, which represent the geometry of the scene in a single time instant. The hyper-surface is found by minimizing the integral of an energy functional which weights photo-consistency as well as temporal smoothness over the hyper-surface.

3. 3D Reconstruction in Spacetime

In this section, we present the mathematical foundations of our 3D reconstruction algorithm. We assume that we have a set of fully calibrated, fixed, and synchronized video 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 wish 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 vary continuously over time.

3.1 Mathematical Foundations

To achieve these properties, we cannot consider each time step individually. Instead, we regard the two-dimensional object surface Σ_t at each time step as subsets of one smooth, three-dimensional hyper-surface $\mathfrak{H}$ embedded in four-dimensional spacetime. From this viewpoint, the reconstructed object 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 one hyper-surface for all frames, temporal smoothness is implicitly guaranteed.

To take photo-consistency of the reconstructed geometry into account, we set up 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 hyper-surface. $\Phi = \Phi(s, \mathbf{n})$ measures the photo-consistency error density and depends on the surface point s and the normal $\mathbf{n}$ at any point. The larger the value of Φ, the higher the photo-consistency error. The surface which matches the given input data best is a minimum of this energy functional. In [8], we make use of 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. In Sect. 4, we review how this Euler-Lagrange equation (2) can be solved in practice using a surface evolution equation implemented using level sets. In the remainder of this section, we present suitable choices for the error measure Φ.

3.2 Spacetime Continuity

We need some additional notation for color and visibility of points in spacetime. Let t denote a time instant. A video image I_k^t is then associated with 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 compute the color c_k^t of every point (s, t) on the hyper-surface by

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

For the 2D object surface Σ_t, let $\nu_k^t(s)$ denote whether point s is visible in camera k at time t. $\nu_k^t(s)$ is defined to be unity if s is visible, and zero otherwise. A photo-consistency error measure can then be defined as

$$\Phi^S(s,t) \;:=\; \frac{1}{V_{s,t}} \sum_{i,j=1}^{l} \nu_i^t(s)\nu_j^t(s) \cdot \left\| c_i^t(s) - c_j^t(s) \right\|. \tag{3}$$

$V_{s,t}$ is used to normalize the function and denotes the number of camera pairs which are able to see the point s at time t. This functional for regular surfaces in $\mathbb{R}^3$ was introduced by Faugeras and Keriven [6] for static scene reconstruction. If the error function Φ^S is used for reconstructing a 2D surface in 3D space, the resulting algorithm is essentially equivalent to space carving, where each time step is regarded separately: In [11], voxels in a discrete voxel grid are carved away if Φ^S lies above a certain threshold value when averaged over the voxel. By using a surface evolution scheme for reconstruction, a continuous surface is obtained, instead.

To reconstruct smoothly varying geometry over time, we increase the dimensionality of the problem by one and regard a 3D hyper-surface in 4D spacetime, while the photo-consistency error measure (3) remains the same.

3.3 Normal Optimization

Because theorem (2) also encompasses error functions that depend on surface normal orientation, we are able to additionally optimize surface normal orientation. A similar idea was presented in [6]. However, we give a slightly modified version and work in spacetime to enforce temporal smoothness.

In order to set up an appropriate error functional, we have to analyze how well a surface point at position s with a given normal $\mathbf{n}$ corresponds to the images at time t. To this end, we assign to each hyper-surface point a small patch $\square_{s,\mathbf{n}}$ within the plane orthogonal to $\mathbf{n}$, Fig. 1. In our implementation, we choose rectangular patches rotated into the target plane by a well-defined rotation. How exactly this patch is chosen does not matter. However, the choice should be consistent over time and space.

We define a measure how well the patch $\square_{s,\mathbf{n}}$ matches the images at time t by employing 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

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

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. It is clear that

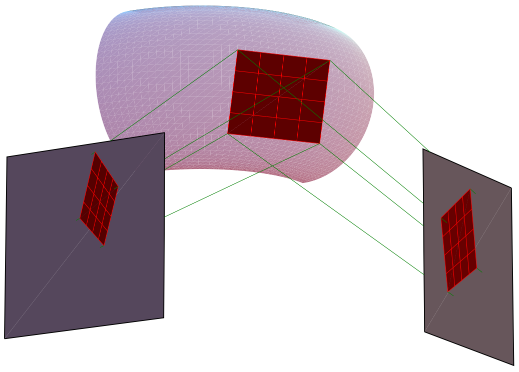


Figure 1. *Practical computation of the cross-correlation error term Φ^G: A small region in the tangent plane is projected into both camera images, and the normalized cross-correlation is computed.*

we need to choose $\square_{s,\mathbf{n}}$ sufficiently large so that it is projected onto several pixels. On the other hand, it should not be too large, otherwise only parts of it are visible in the images. As a compromise, we set its diameter to be equal to the cell diameter of the underlying computation grid, as defined in Sect. 5. 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 $\mathbb{R}^3$. Therefore, we project normals of $\mathfrak{H}$ into the tangent space of Σ_t in order to obtain $\mathbf{n}$.

By minimizing this functional, 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 (2). To numerically find the minimum of the error functional, a surface evolution approach is used, implemented via level sets.

4. Level Set Evolution Equation

In order to minimize the energy functional, we need to find a solution to (2). An efficient way to do so is to rephrase the problem in form of a surface evolution which can be implemented using level sets [15, 5]. This technique is well-established for a wide range of applications [20]. For the remainder of the text, let

$$\Psi := \langle \Phi_s, \mathbf{n} \rangle \;-\; \mathrm{Tr}\,(\mathbf{S})\,\Phi \;+\; \mathrm{div}_\Sigma(\Phi_\mathbf{n}).$$

A surface $\mathfrak{H}$, which is a solution to the Euler-Lagrange equation $\Psi = 0$, is likewise a stationary solution to a surface evolution equation, where Ψ describes a force in the normal direction:

$$\frac{\partial}{\partial \tau}\mathfrak{H}_\tau = \Psi \mathbf{n}. \qquad (4)$$

If we start with an initial surface $\mathfrak{H}_0$ and let the surface evolve using this equation, it will eventually converge to a local minimum of $\mathcal{A}$. Instead of implementing the surface evolution directly, we make use of the notion of level sets. We express the surfaces $\mathfrak{H}_\tau$ for each parameter value $\tau \geq 0$ as the zero level sets of a regular function

$$u : \mathbb{R}^4 \times \mathbb{R}^{\geq 0} \to \mathbb{R}, \quad u(x,\tau) = 0 \;\Leftrightarrow\; x \in \mathfrak{H}_\tau. \qquad (5)$$

We require $u(\cdot,\tau)$ to be positive inside the volume enclosed by $\mathfrak{H}_\tau$, and negative on the outside. An immediate consequence is this

Lemma. Let ∇ be the gradient operator for the spatial coordinates of u. Then we can compute the outer normal and the trace of the shape operator for $\mathfrak{H}_\tau$ using

$$\mathbf{n} = -\frac{\nabla u}{|\nabla u|} \quad \text{and} \quad \mathrm{Tr}\,(\mathbf{S}) = \mathrm{div}\left(\frac{\nabla u}{|\nabla u|}\right).$$

Proof. The relationship for the normal is obvious. By definition, the shape operator is given by $\mathbf{S} := -D\mathbf{n}$ and maps the tangential space $T\mathfrak{H}$ into itself. It follows that

$$\mathrm{Tr}\,(\mathbf{S}) = \mathrm{Tr}\,(-D\mathbf{n}) = \mathrm{div}(-\mathbf{n}) = \mathrm{div}\left(\frac{\nabla u}{|\nabla u|}\right).$$

Note that we consider the normal to be defined on all level sets of u. ◇

Taking the derivative of (5) with respect to τ and inserting the result in the above relationships, we arrive at the final reformulation of (4) in terms of a level-set evolution:

$$\frac{\partial}{\partial \tau}u = \left[-\,\mathrm{div}\left(\Phi \cdot \frac{\nabla u}{|\nabla u|}\right) + \mathrm{div}_\Sigma(\Phi_\mathbf{n})\right] |\nabla u|\,. \qquad (6)$$

In the next section, we analyze in detail how this evolution equation can be implemented efficiently in a multiprocessor environment.

5. Parallel Implementation

In order to implement the level-set evolution equation, the volume surrounding the hyper-surface $\mathfrak{H}$ has to be discretized. We use a regular four-dimensional grid of evenly distributed cells with variable spatial resolution of either 64^3 or 128^3 cells. The temporal resolution is equal to the number of frames in the input video sequences. If the sequence is of any reasonable length, this signifies a massive amount of data and computation time. In fact, it is not possible to store the full data set for each grid cell together with all images of all video sequences within the main memory of a standard PC. A parallel implementation is inevitable, distributing the workload and data over several computers.

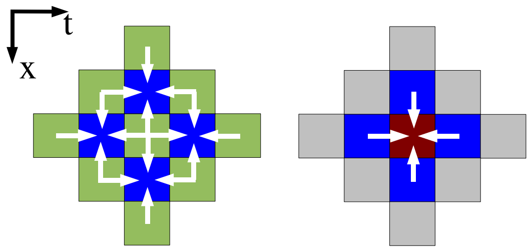


Figure 2. *Evaluation of the differential operator. In the first step, the values of u_i in the light cells are used to compute the level set normal $\mathbf{n} \in \mathbb{R}^4$ in the dark cells using central differences. For the second step, we compute the values for the central cell, also using finite differences.*

We choose to use the narrow-band level-set method [20] for implementing the evolution equation because it is straight-forward 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 is suggested 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 (6) using an explicit time step:

$$u_{i+1}^{xyzt} = u_i^{xyzt} + \Psi\left(u_i^{xyzt}\right) |\nabla u_i| \cdot \Delta\tau. \qquad (7)$$

$\Psi\left(u_i^{xyzt}\right)$ 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 (6), Fig. 2. The norm of the discretized gradient $|\nabla u_i|$ is calculated according to the upwind scheme [20]. 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

$$\Delta\tau \leq \max_{(x,y,z,t)\in\Gamma}\left(\frac{\mathrm{diam\ cell}(x,y,z,t)}{\left|\Psi\left(u_i^{xyzt}\right)\cdot\nabla u\right|}\right). \qquad (8)$$

The differential operator must be evaluated for each grid cell near the zero level set, so the computations necessary for each cell depend only on a local neighborhood, and the computation of individual cells can easily be distributed

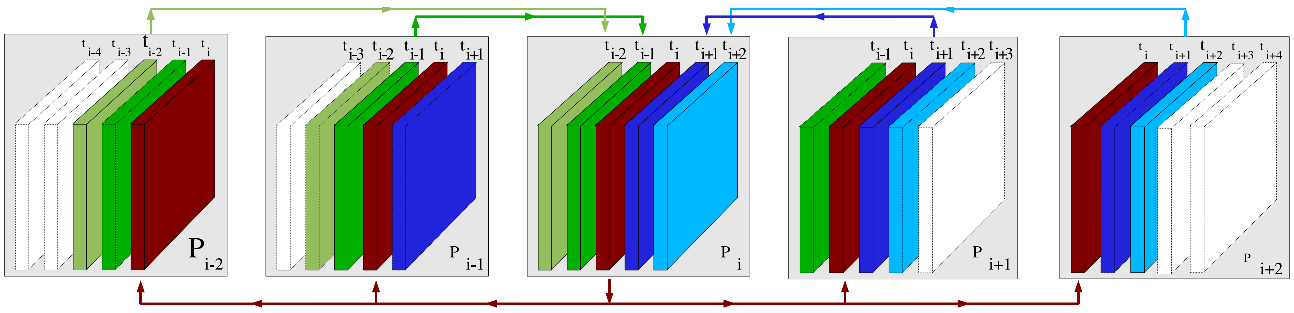


Figure 3. *Data transmission of process P_i before an iteration. Each process stores five slices of constant time and is responsible for the computation of the center slice. P_i computed its slice in the last iteration and now transmits it over the network. On the other hand, it receives the other slices from its neighbors for the next iteration. In the figure, slices of the same color contain the same information after the communication.*

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 illustrates how the value $\Psi\left(u_i^{xyzt}\right)$ is numerically evaluated from the values of u_i in the grid cells. 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 access 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.

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 neighbors according to Fig. 3.
- Then, 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 the maxima, the server calculates the optimal step size $\Delta\tau$ according to (8).
- The server broadcasts the step size to all processes which compute the evolution on their slice according to (7).

After each iteration, the server process may poll the current geometry from any of the other processes in order to give the user feedback about the current state of the iteration. The iteration stops when the flow field is zero, or may be terminated by the user.

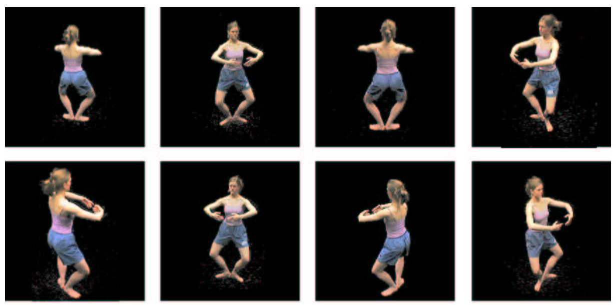


Figure 4. *Input image data: eight segmented video frames per time step represent the input to our algorithm.*

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

6. Results

In order to test our algorithm, we use real-world 320×240 RGB video sequences of a ballet dancer recorded at

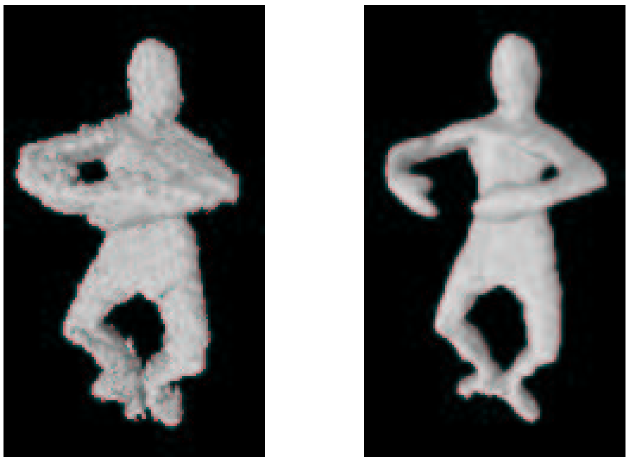

(a) Visual hull (b) Final result

Figure 5. *Reconstruction: The object's visual hull is used as the initial surface to start the PDE evolution. On the right, the final result is depicted after running the complete algorithm including normal optimization.*

15 fps, Fig. 4. All input images are automatically segmented into foreground and background [1]. A level-set representation of the visual hull is computed to initialize the volume for the PDE evolution, Fig. 5.

For our test runs, we selected a sequence with intermittently completely occluded object regions, Fig. 6. The recording cameras are arranged along the perimeter of the scene , so no camera can capture the area between arms and chest of the dancer in the depicted frame. When we run the space-carving algorithm on this frame, little improvement can be observed. Only when we employ spacetime-coherent reconstruction do we obtain a satisfactory result, Fig. 6: The invisible region is interpolated from previous and future frames automatically by the algorithm's implicit temporal smoothness constraint.

In Fig. 7, we illustrate more geometry results for different times and from different viewpoints. When textured with an image-based rendering algorithm using the source images, it can be observed that photo-consistency is indeed excellent.

Tab. 1 states the computation times and memory requirements of each of the slave processes for a single iteration. Our tests were performed on a Sun Fire 15K with 75 UltraSPARC III+ processors at 900 MHz, featuring 176 GBytes of main memory. It can be observed that normal optimization requires a lot of computation time when compared to the standard version of our algorithm. For this reason, we

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 required for a single iteration, depending on the resolution of the computation grid and the number of processors (procs.). Both timings with and without the normal optimization (n.o.) are stated.*

first reconstruct geometry by itself before we turn on normal optimization which is chiefly improves the reconstruction of small surface details. On average, we need around one hundred iterations of the initial evolution and twenty more for normal optimization until the surface converges to the final result.

In order to speed up surface evolution, one further term can be included in (6), as suggested in [6]. We subtract a multiple ϵ of the curvature $\mathrm{Tr}\,(\mathbf{S})$, with ϵ being a small, user-defined constant factor. This forces the resulting hypersurface to become smoother, so larger steps $\Delta\tau$ can be taken to evolve the PDE.

7. Summary and Conclusions

We have presented a novel reconstruction algorithm for dynamic objects which takes into account all frames of a multi-video sequence. The underlying idea is to optimize photo-consistency as well as temporal smoothness simultaneously. Our method is formulated as a weighted minimal surface problem to find a 3D hyper-surface in 4D spacetime. Intersecting this hyper-surface with planes of constant time gives the 2D surface geometry at any time instant. The energy functional is minimized by implementing a surface evolution PDE using the narrow-band level-set method. Significant quality improvements can be observed when compared to photo-consistency approaches which do not take temporal coherence into account.

More results, publications and demo videos on spacetime-coherent reconstruction can be found on our web site *www.grovis.de*.

References

[1] M. Bichsel. Segmenting simply connected moving objects in a static scene. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(11):1138–1142, 1994.

[2] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. on Computer Graphics*, 22(3):569–577, July 2003.

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

[4] 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.

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

[6] 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.

[7] B. Goldlücke and M. Magnor. Space-time isosurface evolution for temporally coherent 3D reconstruction. In *IEEE Proc.Computer Vision and Pattern Recognition (CVPR'04)*, Washington, USA, 2004. accepted.

[8] B. Goldlücke and M. Magnor. Weighted minimal hypersurfaces and their applications in computer vision. *Proc. European Conference on Computer Vision (ECCV'04)*, Prague, Czech Republic, May 2004. accepted.

[9] 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.

[10] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[11] 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.

[12] 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.

[13] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. Image-based 3D photography using opacity hulls. In *Proceedings of ACM SIGGRAPH*, pages 427–436, 2002.

[14] P. J. Olver, G. Sapiro, and A. Tannenbaum. Invariant geometric evolutions of surfaces and volumetric smoothing. *SIAM Journal on Applied Mathematics*, 57(1):176–194, 1997.

[15] 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.

[16] 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.

[17] M. Pollefeys and L. V. Gool. From images to 3D models. *Communications of the ACM*, 45(7):50–55, 2002.

[18] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[19] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):1–23, Nov. 1999.

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

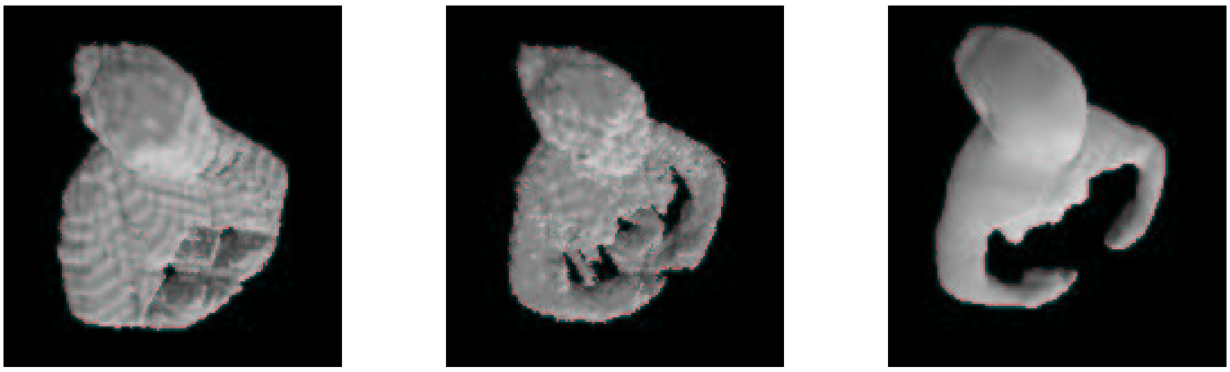
[21] S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6D. In *Proc. Computer Vision and Pattern Recognition (CVPR'00)*, volume 2, pages 592–598, 2000.

[22] S. Würmlin, E. Lamboray, O. Staadt, and M. Gross. 3D video recorder. In *Proceedings of Pacific Graphics*, pages 10–22, 2002.

[23] J. Xu and H. Zhao. An Eulerian formulation for solving partial differential equations along a moving interface. *Journal of Scientific Computing*, 19(1-3):573–594, 2003.

[24] 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.

[25] H. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. In *Computer Vision and Image Understanding*, pages 295–319, 2000.



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

Figure 6. *Comparison of different reconstruction algorithms (grid resolution* 128^3*). (a) The visual hull, as seen from above. Since there is no camera to capture the scene from above, most voxels in the area between the arms remain occupied. (b) The result obtained by static space carving. The invisible regions between the arms can still not be resolved. (c) Our algorithm uses temporal information to interpolate the currently invisible region between the arms.*

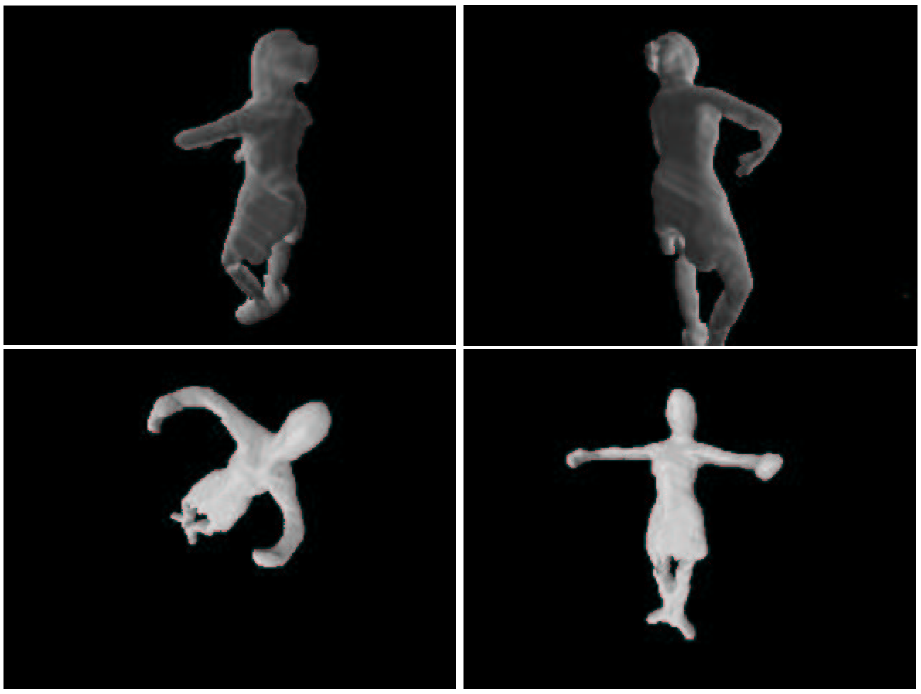


Figure 7. *Reconstructed geometry for time instants of the sequence. All depicted viewpoints are far away from the viewpoints of the cameras.*