Space-time surface reconstruction using incompressible flow

ANDREI SHARF[1], DAN A. ALCANTARA[1], CHEN GREIF[2], THOMAS LEWINER[3], ALLA SHEFFER[2], NINA AMENTA[1] AND DANIEL COHEN-OR[4]

[1] Department of Computer Science — University of California — Davis — United States
[2] Department of Computer Science — University of British Columbia — Vancouver — Canada
[3] Department of Matematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[4] School of Computer Science — Tel Aviv University — Israel
www.math.tau.ac.il/~asharf. graphics.idav.ucdavis.edu/people/profile?pid=888.
www.cs.ubc.ca/~greif. www.mat.puc--rio.br/~tomlew. www.cs.ubc.ca/~sheffa.
www.cs.ucdavis.edu/~amenta. www.math.tau.ac.il/~dcor.

Abstract. We introduce a volumetric space-time technique for the reconstruction of moving and deforming objects from point data. The output of our method is a four-dimensional space-time solid, made up of spatial slices, each of which is a three-dimensional solid bounded by a watertight manifold. The motion of the object is described as an incompressible flow of material through time. We optimize the flow so that the distance material moves from one time frame to the next is bounded, the density of material remains constant, and the object remains compact. This formulation overcomes deficiencies in the acquired data, such as persistent occlusions, errors, and missing frames. We demonstrate the performance of our flow-based technique by reconstructing coherent sequences of watertight models from incomplete scanner data.
Keywords: ***Reconstruction. Space-time. Volumetric techniques.***

Figure 1: *Reconstruction of a space-time surface.* Left*: A sequence of point clouds of a running man containing holes due to self occlusions.* Middle*: Renderings of two iterations of our flow solver. Cells confidently identified as being inside the surface are colored dark blue. As the solver progresses, mass accumulates inside the surface.* Right*: The result of an implicit function reconstruction from the flow solution.*

1 Introduction

Recent advanced scanning technologies together with increasing computational power allow the space-time capture of three-dimensional objects as they move and deform. Systems such as [34, 9, 19] produce dense point samples over large parts of the surface of a moving object, at rates from ten to thirty frames per second. As these technologies mature, they make it possible to capture and reconstruct both the deforming model and its motion. Organizing the data captured by these scanners into a coherent model of a deforming object is a challenging computational problem that is just beginning to be addressed [23, 32, 25].

As pointed out in one of the first applications of real-time scanning [26], the main challenge posed by the time component is to fill in missing data by accumulating information over time. The input scans are typically collected by a small set of synchronized cameras. Because of the small number of fixed views, large parts of the surface are occluded in each frame, leading to gaping holes that often persist across many

Preprint MAT. 03/08, communicated on January 26^{th}, 2008 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in the proceedings of Siggraph Asia 2008: Transaction on Graphics, volume 27, number 5, paper 110, ACM 2008.

frames. Current real-time systems also suffer from low resolution, insufficient frame rate and noise. While it is reasonable to assume advances in real-time scanning technology will improve the data quality, persistent occlusions will always be an issue.

To complete the missing data in a principled way we take advantage of space-time coherence and adopt a global approach, which considers all frames simultaneously. Furthermore, to guide the reconstruction, we include as many reasonable physical assumptions as we can into our computation. First, we directly reconstruct a deforming solid, that is, a four-dimensional space-time surface. By explicitly modeling the mass field of the object, we leverage the knowledge that the boundary of the reconstructed object is a watertight manifold. Most importantly, we explicitly model the flow of this material through time. We constrain all of the material in one time-frame to move to some nearby position in the next, ensuring mass conservation, while preventing mass from concentrating or dispersing, producing what we call an *incompressible flow*. We also introduce a momentum term which ensures that material moves smoothly through time. We use this flow terminology somewhat loosely; as will be seen, our method does not produce a physically accurate fluid flow; nor does it need to.

This incompressible flow prior puts a very strong constraint on the shape of the four-dimensional solid and on how its 3D spatial slices evolve over time. Using all the frames simultaneously to compute the flow, data from arbitrarily distant frames is used to plausibly complete missing data in occluded regions. We can also successfully reconstruct entirely missing frames by extrapolating information from sampled ones, overcoming sparse frame rates (Figure 6).

Our formulation requires the fairly mild assumptions that the object moves smoothly through time and that the speed of movement between consecutive time frames is bounded. We do not need to assume that the motion is globally or locally rigid, nor do we have any assumptions on the topology or geometry of the object. In contrast to many previous methods [22, 16, 33] we do not require additional space-time coherence information such as marker correspondences.

Specifically, each three-dimensional frame is represented by its characteristic function on a regular grid. We construct the mass field, i.e. the characteristic function, and its flow simultaneously for *all* frames. To do so, we constrain all the material at time t to move to some adjacent grid cell at time $t + 1$. We set the material in any cell identified as inside the object to be one and in any outside cell to be zero, avoiding compression or dilatation inside each cell. The estimation of both distribution of material in space and its flow is computed by iteratively updating and solving a linear system, applying at each iteration a simple re-weighting step to enforce incompressibility and enhance the object boundaries.

The obvious difficulty with our approach is that the entire volume and its flow are explicitly represented, producing a very large linear system. We handle this by employing a preconditioned Krylov subspace iterative solver that exploits the sparsity and the specific spectral properties of the system.

2 Background

While numerous methods exist for reconstructing static 3D models, none of them is directly applicable to the dynamic seting. The main challenge faced by dynamic reconstruction is to properly define the relationship between the temporal and spatial dimensions in order to accumulate information over time to correctly complete locally missing data [14, 26]. Much of the attention in 4D reconstruction has been paid to processing specific shapes such as humans or garments where additional information is available to guide the reconstruction, e.g., marker correspondences between scans [22, 16, 33] or between the scans and a template [2, 3, 35, 4]. Unmarked data is much simpler to acquire making a markerless approach like ours more general.

Arbitrary shape reconstruction from point cloud sequences is usually treated as a surface deformation over time. Specifically, Shinya [28] extracts an initial surface from the first point cloud, and evolves it towards the subsequent point clouds while minimizing a deformation energy computed over the surface triangulation. Wang et al. [31] further refine this energy using harmonic maps. Anuar and Guskov [5] use optical flow to estimate the deformation, switching from surface representation to volumetric flow integration. Recently, Pekelny and Gotsman [25] introduce another method to accumulate information for the specific case of articulations of rigid parts. The input point cloud is manually segmented and the parts are tracked over time assuming that each frame is a good initial guess for an iterative closest point rigid motion registration with the next frame. These approaches are sensitive to the quality of the initial surface since they accumulate information only forward in time.

To integrate time information for arbitrary shapes, Wandet al. [32] and Mitra et al. [23] formulate the reconstruction problem directly in 4D, focusing on inter-frame registration. Wand et al. [32] optimize a 4D shape represented by a set of surfels. They define the surface from statistical densities, imposing spatial smoothness and rigid motion priors. Mitra et al. [23] register point clouds directly on a 4D hypersurface, assuming rigid interframe motion. With dense temporal sampling, this allows estimating local deformations of the model. We approach the problem in 4D as well, but by operating on a volume, rather than a surface like Wand et al. [32] or Mitra et al. [23]. In particular, we generate a sequence of watertight manifolds as output, and we model the flow of mass in order to compensate for missing data. Unlike these techniques our method does not require dense spatial sampling to complete missing data.

Other approaches address the related problem of reconstructing moving objects from video sequences [21, 11, 12]. They carve a 4D hypersurface and then optimize it by enforcing photo-consistency with a 2D video sequence.

The corresponding work was published in the proceedings of Siggraph Asia 2008: Transaction on Graphics, volume 27, number 5, paper 110, ACM 2008.

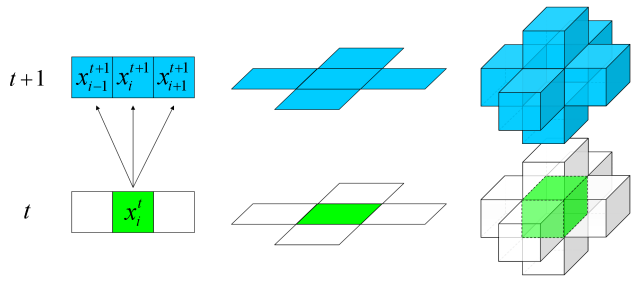

Figure 2: *Spatial and time adjacency in 1D, 2D and 3D space: white cells are adjacent to the green cell in space; blue cells are adjacent to the green cell in time. Only blue cells can have non-zero incoming flow from the green cell.*

The ability to reconstruct a watertight object using the strong incompressible flow prior distinguishes our method from previous techniques. This prior leads to more reliable reconstruction enabling us to obtain accurate, watertight reconstructions from poorly sampled data.

As a last note, our solution to the 4D problem uses the concept of material flow. Similar techniques have been devised for 2D video in the Computer Vision literature [29, 15, 30, 6], to track motions through minimization of specific image similarity measures.

3 Problem Formulation

We consider the problem of reconstructing a moving and possibly deforming object of arbitrary topology given a sequence of three-dimensional frames. Each frame consists of a cloud of points in 3D sampled over the object's surface. Our goal is to reconstruct a watertight surface in space-time, that is, a three-dimensional surface embedded in 4D. We represent this 4D solid using its characteristic function on a 4D grid, from which we extract the actual object surface at a final step (Section 4(c)) .

For the sake of simplicity we describe our formulation for an object in one-dimensional space deforming through time. The domain is then an $(n+1) \times (m+1)$ space-time grid, and we use i, j for space indices and t for time ones. The three- and four-dimensional formulations we use later are straightforward generalizations obtained by replacing the spatial adjacency relationships used in one dimension with those for higher dimensions (Figure 2). The characteristic function values x_i^t at each cell can be seen as representing the amount of material in the cell. Using this representation we describe the motion and deformation of the object over time as flow of material through space-time. The flow is represented by variables $v_{i,j}^t$ representing the amount of material moving from cell x_i^t at time t to another cell x_j^{t+1} at time $t+1$. Note that, in contrast to the the simulation of fluid flows, material in our model moves through time at a constant rate, so that higher flow values on an edge mean more material moving through the edge. We formulate the reconstruction as a solution to a constrained minimization problem, based on the following set of assumptions.

Flow incompressibility: We interpret flow incompressibility as constraining the amount of material in any cell to be equal to the amount of material flowing into the cell from the previous time frame, and also to the amount of material flowing out of the cell into the next time frame. Using the previous notation, this is:

$$x_i^t = \sum_j v_{j,i}^{t-1}, \qquad i = 0 \ldots n, \;\; t = 1 \ldots m \tag{1}$$

$$x_i^t = \sum_j v_{i,j}^t, \qquad i = 0 \ldots n, \;\; t = 0 \ldots m-1 \tag{2}$$

Bounded speed: We expect the grid resolution in time to be sufficiently dense with respect to the deformation speed. Specifically, we assume that at each time step material can only move to temporally adjacent cells (shown in Figure 2). In $1D \times time$ this means that $v_{i,j}^t$ is non-zero only if $i - 1 \leq j \leq i + 1$. This constraint is handled implicitly by including in the formulation only the variables for $v_{i,j}^t$ that can potentially be non-zero. Note that the actual sampling of the data in time could be at a lower resolution than that of the solution grid, or in other words we can have time-frames with no samples.

Spatial continuity: In the characteristic function, we expect values x_i^t in spatially adjacent cells to be identical everywhere, except across the object boundaries. Because the scans give an incomplete representation of the surface, however, we do not know where some of the boundaries are and in addition we cannot be sure of the in/out orientation at all known boundaries. We therefore require the values of x_i^t to be constant everywhere except at a small number of sharp discontinuities. Minimizing a functional such as

$$F_c(x_i^t) = \sum_t \sum_{i=1\ldots n} \left(x_i^t - x_{i-1}^t\right)^{0.8} \tag{3}$$

gives this effect. The sub-linear exponent 0.8 in the norm penalizes many small discontinuities more than a few large ones [20]. It would be easier, computationally, to use a least-squares error functional, since the derivative would be linear. But that would have exactly the wrong effect; with an exponent of 2 instead of 0.8, the minimum would be achieved when the differences between cell values x_i^t are evenly distributed across all cells. We address this computational issue, and also some refinements of this functional, in Section 4(b).

Note that F_c only links cells adjacent to one another in the spatial dimension. We use the flow to control the variation of the function in the time dimension.

Flow momentum: We expect the object deformation, and hence the flow, to be smooth in time. This smoothness is captured by a momentum term that penalizes the flow for changing direction from one time frame to the next:

$$F_m(v_{i,j}) = \sum_{i,j,t=0\ldots m-1} \left(v_{i,j}^t - v_{j,j+(j-i)}^{t+1}\right)^2 \tag{4}$$

This term uses the discretized flow directions available on the grid, which is not correct at the object boundaries; there we rely on the spatial continuity term to enforce continuity of

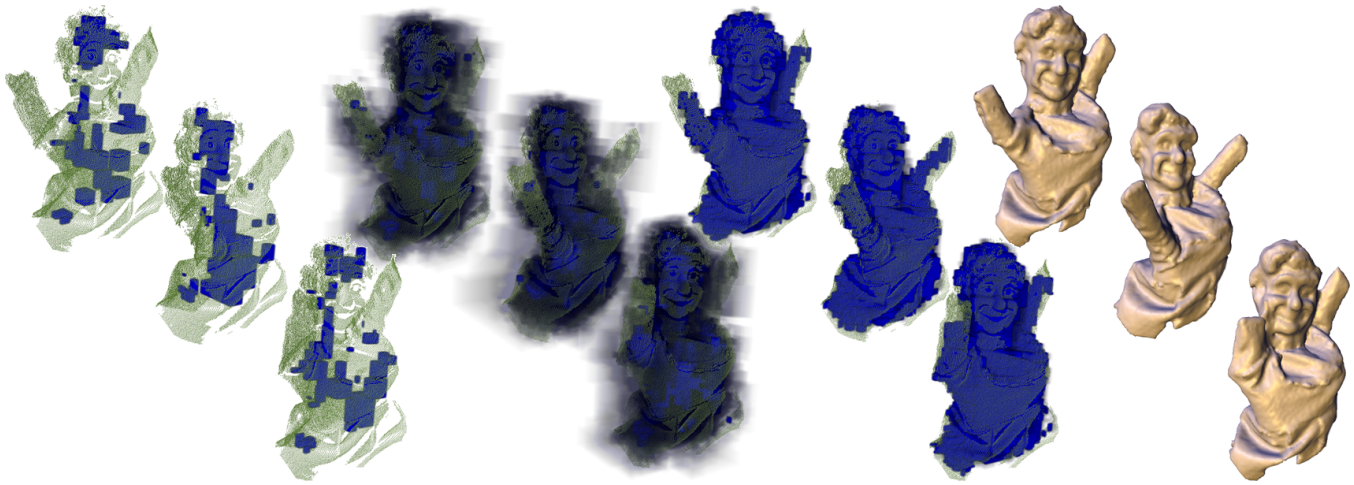

Figure 3: *Reconstruction of a scanned moving hand puppet.* Left*: Input scan points to the flow solver (green points), with the initial inside cell labeling (dark blue cells).* Center*: Two flow solver iterations: mass cells is represented by a grayscale map. Cells determined to be inside by the solver are colored dark blue.* Right*: Our final reconstruction.*

momentum as well. We tried including a similar boundary-enforcement mechanism in the momentum term, but we found that the slight improvement in performance did not justify the additional cost.

Data fidelity: The values of the characteristic function x for some of the grid cells are known even before the optimization starts. Specifically, we assign $x_i^t = 1$ to cells containing scan points, and we mark those as boundary cells. Values of zero and one are assigned to additional cells, both at initialization and during the computation, as discussed in Section 4. These values are treated as hard constraints and handled implicitly by using back-substitution, removing the corresponding variables from the system.

Problem formulation: The solution to our space-time reconstruction problem is computed by optimizing a weighted combination of spatial continuity F_c and flow momentum F_m, $F = \alpha F_c + (1-\alpha)F_m$, subject to the incompressibility constraints. We used $\alpha = 2/3$ for the 2D$\times$time examples, and $\alpha = 1/3$ for the 3D$\times$time ones, giving higher weight to flow smoothness. This difference of weighting was necessary to compensate for the increased number of spatially adjacent cells in each time frame. We observed that when there was a lot of missing data, putting more weight on flow momentum filled in missing areas faster, accelerating convergence.

4 Solution Mechanism

(a) Preprocessing and Initialization

The characteristic function values for some cells can be reliably computed without the flow computation. Initializing as many cells as possible drastically reduces the number of variables in the functional to be optimized, significantly increasing solution speed. It also improves the solution accuracy. Therefore we do as much preprocessing as we can before attempting the flow computation.

We begin by computing a low-resolution visual hull for each frame, based on the scan planes and the scanner positions and orientations. This allows us to loosely label many outside cells. Certainly in the presence of noise and holes, estimation of the visual hull is error-prone, nevertheless, at this pre-processing step high precision is not required.

We select a small number of cells to loosely label as inside using the following heuristic. We compute for each frame a low-resolution unsigned distance field measuring the distance from scan data, and select cells that are both local maxima of the distance function and are inside the visual hull. The idea is that such cells simultaneously lie near the medial axis of the surface and inside the visual hull, and as such are very likely to be inside the object itself.

This is followed by a low-resolution 4D surface reconstruction, using the FEM reconstruction method of Sharf et al. [27]. This algorithm takes as input both the data points and the above described inside/outside labels. It computes a smooth function that is negative inside the object and positive outside. At such low resolution, this computation is very efficient. We utilize the FEM solution in two ways. First, we use it to improve our inside/outside labeling; we label maxima of the FEM function as outside the object and its minima as inside. Second, we use it to assign normals to the data points. We use these normals to modify the spatial continuity functional F_c, as described below in Section 4(b). Examples of initial cell labeling are shown in Figure 3.

(b) Optimization

We now describe our optimization of the flow functional. We use an Iteratively Reweighted Least Squares [18] approach to handle the sub-linear exponent in F_c. We introduce a weight $w_{i,j}^t$ for every pair of spatially adjacent cells x_i^t and x_j^t, which is used to modify the spatial continuity functional F_c. The overall computation requires iteratively solving a large linear system, modifying the $w_{i,j}^t$ at each it-

The corresponding work was published in the proceedings of Siggraph Asia 2008: Transaction on Graphics, volume 27, number 5, paper 110, ACM 2008.

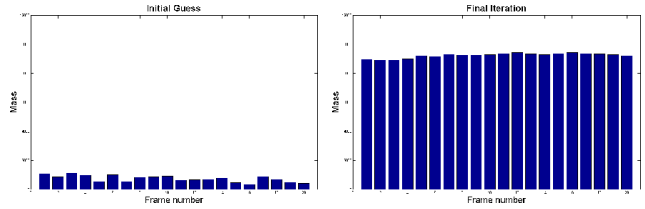

Figure 4: *Graphs demonstrating mass incompressibility and spatial continuity principles on the Elephant data set(video).* Left*: Our initial guess labels only few cells as inside mass per frame.* Right*: Spatial continuity term propagates the initial mass throughout the solid, while incompressibility term keeps total mass nearly level across frames.*

eration.

Initialization: In our initial setting of weights we observe that the discontinuities are expected to be at the boundaries of the object. We use the data-point normals determined during the FEM pre-processing to loosely estimate which faces of the boundary cells point outwards, by comparing the face and data-point normals. Then in the first iteration we optimize the spatial continuity functional:

$$F_c'(x_i^t) = \sum_t \sum_{i=1\ldots n} w_{i,i-1}^t (x_i^t - x_{i-1}^t)^2 \qquad (5)$$

where the initial value of $w_{i,j}^t$ is zero if the face (i,j) is an outward boundary face, and one everywhere else. Due to the subsequent global optimization, local normal estimation errors have no significant impact on the final result. In figure 4 we show that mass propagates from the small initial guess (left) to the actual mass (right) due to this continuity setting.

Iterations: After the first step the resulting distribution of x_i^t values provides some indication of where discontinuities are likely to arise; the greater the discontinuity between two cells in the current solution, the more likely a boundary falls there in the final result. Iteratively Reweighted Least Squares [18] suggests a principled way to use this information: we set the weight w_i^t in Equation (5) to be $w_i^t = (x_{i+1}^t - x_i^t)^{0.8-2}$, using the values of x from the previous iteration. This effectively guarantees that if the scheme converges, the solution we have is the minimum of the original functional F_c. To normalize the weights, we bound them from above by $0.001^{0.8-2}$ and scale into the range (0, 10). As expected, as the iteration proceeds the values of x_i^t become concentrated at zero and one as desired (Figure 6).

Regularization: The matrix in the linear system we solve at each iteration is of the form of a discrete Laplacian, and as such can be ill-conditioned, causing numerical instability. We use a standard regularization technique [17] adding the following term to our functional:

$$F_r = \sum (x_i^t)^2 + \sum (v_{i,j}^t)^2.$$

F_r is assigned a fairly small weight of 0.0025, since while we want the regularization to stabilize the system, it should have little effect on the final result.

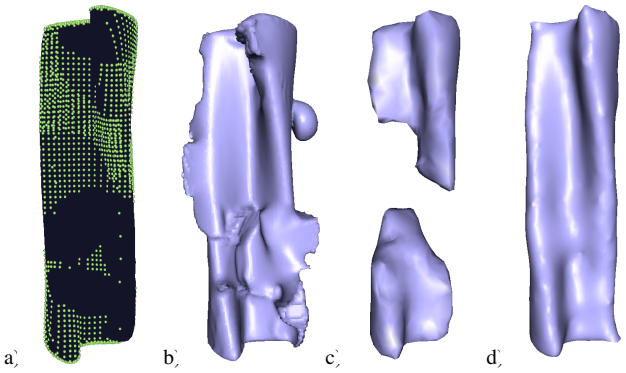

Figure 5: *A rotating 2D boomerang sweeps out the space-time surface (a); green points show where the surface was sampled. A horizontal slice represents the state of the boomerang at one time frame. Comparing 3D RBF reconstruction (b), direct FEM reconstruction (c), our flow reconstruction (d), we note how our flow overcomes the unsampled frames and preserves the boomerang's concavity.*

Bootstrapping and clamping: As the flow solution is computed iteratively, many of the values obtained for x_i^t approach the extrema of zero and one; in fact, the linear system may also produce values of x_i^t which are negative or greater than one. In our experiments only slight deviations occurred from the $[0-1]$ range. We use these extreme values of x_i^t to classify the corresponding cells as inside or outside. In subsequent iterations we clamp the values at these cells to be exactly one or zero, and use them as additional data fidelity constraints, using back-substitution.

Clamping introduces some rounding error to our mass conservation, but the error is essentially random and does not lead to a serious bias (around 2%, as seen in Figure 4). It also enforces incompressibility; an x_i^t greater than one would represent a concentration of material in a cell, which is not allowed in our flow formulation.

Our formulation also allows us to enforce some flow constraints via back-substitution, further reducing the number of variables in the system. Specifically, given a cell with value zero, we can trivially constrain all flow incoming into the cell or outgoing from it to zero. Similarly, if the incoming or outgoing flow for a cell sums to zero, the cell value is set to zero.

The algorithm effectively converges when the values for all the cells are set by the bootstrapping. In practice, to speed up the process, we typically stop once 90% to 95% of the cell values are set, with the final step of the algorithm (Section 4(c)) resolving the classification for those based on smoothness considerations.

(c) Postprocessing

In the postprocessing step, we refine the optimal flow solution to generate the final high resolution output surfaces. This step serves two purposes; first, it improves the flow resolution if necessary, and second, it replaces the characteristic function with an estimated signed distance function, which is

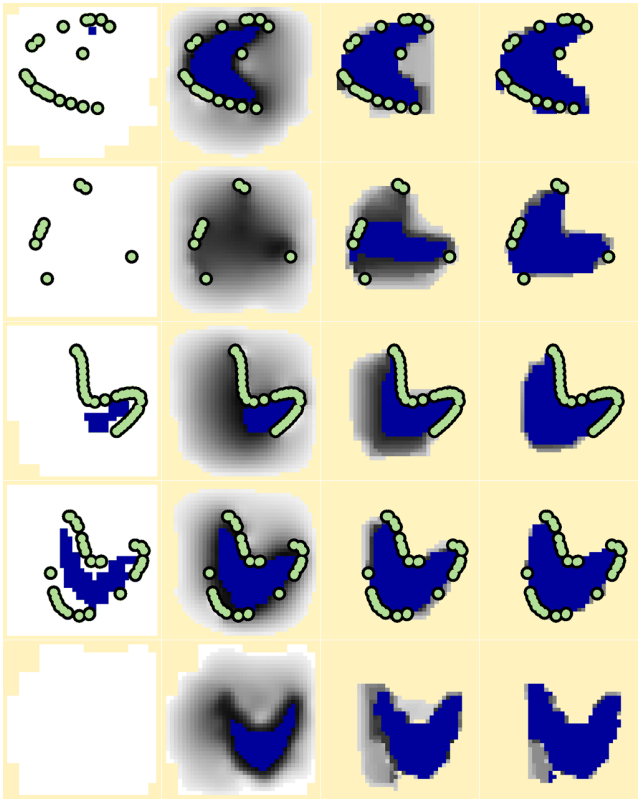

Figure 6: *Flow solution for several frames of the boomerang rotation (Figure 5). Points on the 2D outline are represented by green circles, while mass is represented by a grayscale map. Cells determined to be inside (resp. outside) by the solver are colored dark blue (resp. cream). Mass concentrates on the boomerang's shape as the iterations progress (from left to right) even for frames with few or no data-points (bottom row).*

easier to contour. The refinement is done on a *hexadeca-tree* ($4D$ octree) so that we can increase resolution near the surface to account for fine details in the input data.

We use a standard least squares solver for a Laplace operator on the hexadeca-tree, which maximizes local space-time smoothness. The in/out voxel labels of the flow solution are used as boundary conditions in the Laplace system. We assign signed distance values from the surface to cells in this tree based on both the flow solution and distance to the input data points. Cells deep within the object and cells containing data points are weighted more heavily than cells near the surface but with no data. Thus, the space close to the surface, is initialized as unknown.

The Laplace solution defines a 4D implicit function whose zero set represents the desired 3D surface across time. To extract polygonal surfaces, a standard marching cubes surface extraction is applied at each temporal cross-section of the implicit field independently.

5 Solving the Linear System

To utilize the presented formulation we must efficiently handle the large linear systems that arise during the computation. In this section we show how we exploit the matrix block structure to compute a solution quickly and with a minimal overhead in terms of memory. To accomplish this, we use an augmentation approach [10].

At each iteration our formulation yields linear systems of the form

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

where A arises from the optimized functional and B reflects the constraints (Section 3). The matrix is large, sparse and indefinite, and standard direct techniques based on forming the LDL^T decomposition with symmetric pivoting result in significant loss of sparsity. This in turn would lead to unreasonably large memory requirements and prohibitive computational cost. Therefore, we use the MINRES iterative method with a symmetric positive-definite block diagonal preconditioner which is well suited to the problem and requires a minimal overhead.

We set the preconditioner to be

$$M = \begin{pmatrix} A + \gamma^{-1} B^T B & 0 \\ 0 & \gamma I \end{pmatrix},$$

where γ is a scalar and I is the identity matrix. Denote the matrix of the system by K. The matrix $A + \gamma^{-1} B^T B$, which is a stabilized primal Schur complement of K, is sparse and symmetric positive-definite. It can be shown that the preconditioned matrix $M^{-1}K$ has an eigenvalue 1 of algebraic multiplicity equal to the dimension of A. Furthermore, the negative eigenvalues of the preconditioned matrix are all strictly between -1 and 0, and in our setting most of them are located near -1, since nearly zero eigenvalues of A are mapped into eigenvalues of the preconditioned matrix that are nearly -1 [13].

This high algebraic multiplicity is crucial, since the speed of convergence of MINRES (and in fact any preconditioned Krylov subspace method) primarily depends on how well the eigenvalues of the preconditioned matrix are clustered; in this case we will be able to obtain convergence within a number of iterations which is significantly smaller than the dimensions of the linear system, and in fact also smaller than the number of constraints.

Our numerical solution is based on inner/outer iterations. In solving for $A + \gamma^{-1} B^T B$ (the inner iteration) we apply a conjugate gradient solver, preconditioned with an incomplete Cholesky factorization with a drop tolerance of 0.01. Since the discrete operators A and $B^T B$ have spectral norms of approximately the same order, we set $\gamma = 1$ throughout the computation. We solve the inner iterations with a convergence tolerance of 10^{-6}. The outer MINRES iterations are solved with a convergence tolerance of 0.001 to 0.00001.

The use of a high drop tolerance for the incomplete factorization and a loose outer convergence tolerance make the solver computationally inexpensive, with small iteration counts and modest memory requirements. Even in four dimensions the solver effectively converges within fewer than

The corresponding work was published in the proceedings of Siggraph Asia 2008: Transaction on Graphics, volume 27, number 5, paper 110, ACM 2008.

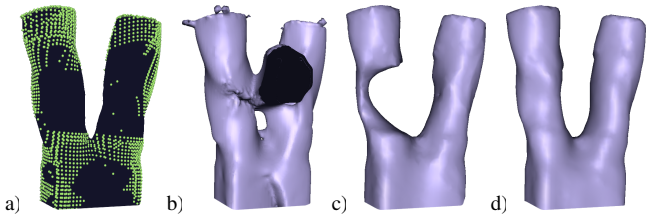

Figure 7: *The 2D space-time surface of a single rounded box splitting into two distinct round objects (see Figure 8). Sampled space-time surface (a). Implicit surface reconstruction using RBF (b) and using FEM without using flow (c). Results using our incompressible flow (d).*

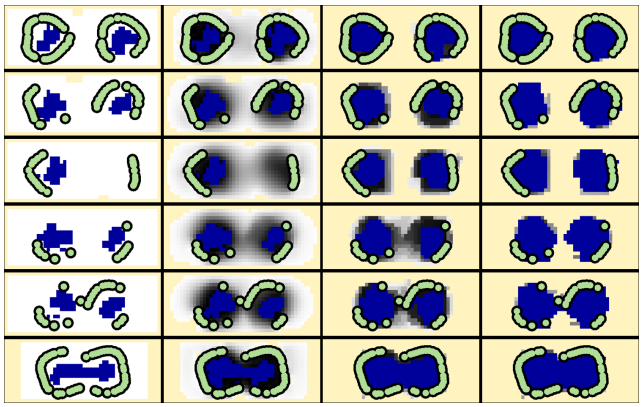

Figure 8: *Flow iterations for the splitting object of Figure 7. Despite the poor sampling of the round areas, the estimation of the mass is able to separate and accumulate where the round objects are expected to be for each time step.*

a hundred outer iterations. Our solver implementation combines the TAUCS library [36] with our own specialized MINRES solver. We can solve the system either on a standard workstation (2GB RAM 2.4 GHz CPU), or very quickly on a small 40-cores cluster, as described in Section 6. As a comparison, a direct parallel LDL^T solver for the same problem takes more than a day to compute the solution on the same cluster.

6 Results

We have tested our method on several real and synthetic examples, in both 2D×time and 3D×time. In all cases our method produced watertight surfaces with good quality completion of missing data.

2D×time results. To illustrate the merits of the flow solution, we use some 2D×time examples. In Figures 5 and 6, we show a two-dimensional "boomerang shaped" object moving through time. We leave large portions of the moving object un-sampled in many frames, mimicking the effects of occlusion. Furthermore, in order to explore our method's limits and robustness, we leave some frames completely empty (Figure 6 bottom row). Observe at the center of Figure 5 that the reconstructions generated by usual surface reconstruction, such as RBF [24] or direct FEM [27], introduce large cavities where significant data is missing. We note that while these might be valid reconstructions of a 3D solid, they are inconsistent under our mass incompressibility assumption. Figure 6 illustrates the flow solution across several frames for the boomerang example. The areas of the final 2D cross sections are roughly equal across all frames; they are not perfectly identical since the clamping of material values to either zero or one at most voxels leads to rounding errors.

A similar behavior can be observed in Figures 7 and 8, which show a two-dimensional rounded box that splits from one connected component into two. Although there are very large chucks of missing data, our method faithfully reconstructs the geometry. In Figure 9, we again compare the results of our method on three other moving and deforming 2D objects to a straightforward 3D reconstruction. Without the mass conservation effects of the flow solution, the direct reconstruction is incorrect. The input data in these examples visually demonstrate the type of problems that also occur in 3D×time setting.

3D×time setting. We tested our method on several synthetic and real 3D space-time data-sets, whose statistics are reported in Table 1. For the flow computation, the spatial resolution of all of our frames is 64^3. For the surface extraction, the final resolution of the hexadeca-tree varies from 64^4 to 256^4 depending on the model smoothness. While this resolution is not particularly fine, it was sufficient for the purpose of identifying inside/outside cells in our tests.

In our computations we are staying within the limit of 20 frames at constant resolution for the flow solver which converges in about 100 iterations. Using this framework we achieve an average computational time of one minute per-frame, roughly split into 20sec per flow solver and 40sec post-processing on a workstation with 16GB RAM 3.73 GHz CPU. We note that these times are an order of magnitude faster than those reported by Wand et al. [32] and about twice as fast as those of the recent marker-based method of White et. al. [33]. The memory requirements of our method are up to 4.5GB for the flow-solver and up to 2GB for the post-processing.

To generate the synthetic inputs in our experiments, we use a virtual multi-camera that simulates a real-time laser scanner. For each camera it shoots rays, registering the closest visible surface points from the virtual camera position. We record only position information for each scan point and not surface orientation.

3D×time results. Figure 1 shows the virtual scan of a running man, using four cameras. The large holes due to occlusions are completed in a natural manner with the space-time model.

Figure 11 shows a scan of a wooden doll, this time using only two virtual cameras. The larger occluded regions are again completed in a manner consistent with other frames, although this time the smooth completions in occluded re-

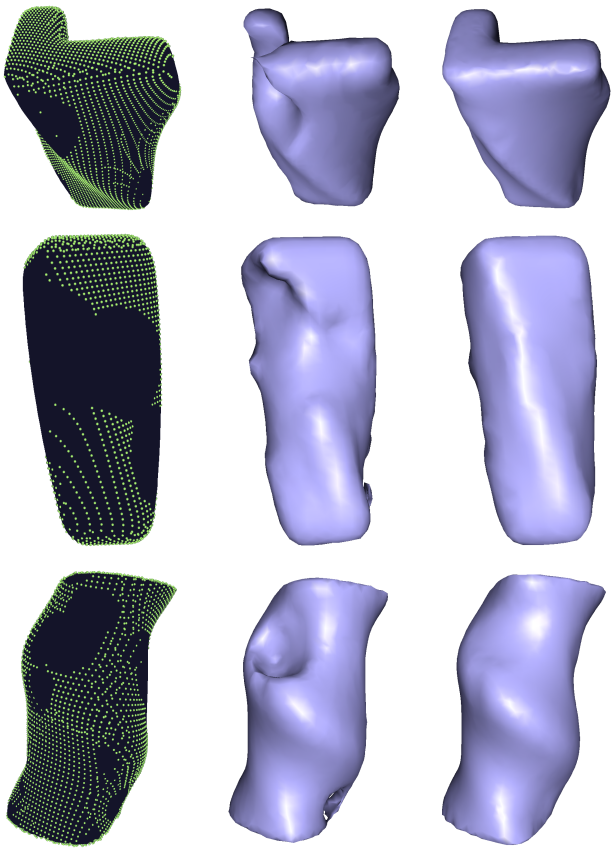

Figure 9: *Various examples of 2D shapes moving over time, sampled so that large holes exist near the objects borders.* Left*: The original space-time surfaces with point samples.* Middle*: direct FEM surface reconstructions.* Right*: Reconstructions using our incompressible flow method.*

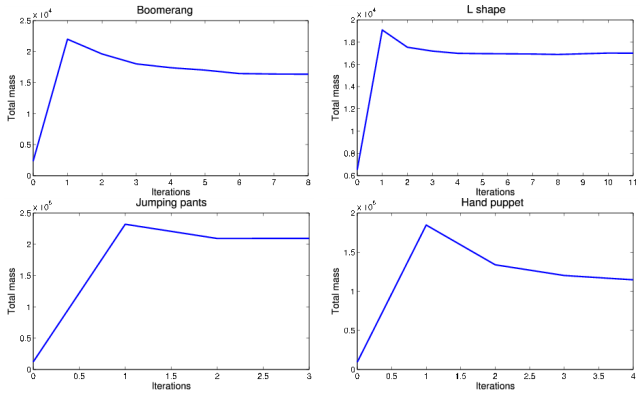

Figure 10: *Graphs showing the total mass variation across iterations of the solution procedure.* Top*: 2D+time data.* Bottom*: 3D+time data.*

gions are somewhat noticeable. As demonstrated, using an implicit FEM reconstruction alone (at 128^3 resolution) produces significantly inferior results.

Figure 3 shows a reconstruction of a moving hand puppet, captured by a low resolution structured light scanner from only two view points. The point data is noisy and has a persistent wide strip of missing data on the sides of the head. The front-facing arm sometimes occludes the face, and the back arm is sometimes occluded. The topology of the head and both arms is correctly reconstructed with the flow solution, and the geometry is fairly consistent. The thin palms of the puppet are too small to be recovered well given the flow grid resolution.

We also tested our method on inputs for which our incompressibility assumption is only approximately satisfied, such as the garment scans in Figures 12 and 13. Leveraging our iterative solver's ability to solve the system inexactly, even in the case of singularities, our method succeeds in reconstructing these inputs and filling holes due to occlusion. In Figure 12, we use an animation of a jumping pair of pants [33] to create a simulated scan using three cameras. The pants rotate, so that almost every part of their surface is visible at some time. The flow solution accurately integrates the data from different times.

Figure 13 shows the results of applying our method to real scans of moving garments. The data was captured using a set of sixteen high-definition video cameras combined with a multiview stereo registration step [7]. Although the scans contain holes due to occlusions, irregular sampling and noise, our flow system successfully generates a coherent 4D volume that smoothly varies across time and fills the missing data. Furthermore, due to the volumetric nature of our solution, the reconstructed shapes are always watertight manifold polygonal surfaces.

Finally, in Figure 10 we graph the mass convergence over the course of the flow iterations. The large jump in the first iteration arises from our initialization procedure, which marks cells known to be either inside or lying on the boundary as having a mass of 1. After the first iteration, we have a better guess of the amount of mass actually contained in the system. Therefore, as the iterations progress, the total amount of mass levels out. Similarly, Figure 4 shows the mass propagation from an initial guess to the actual mass in the last iteration. The measurements were carried on the Elephant data set [32] shown in the accompanying video. We note that while on this example both methods correctly reconstruct the watertight surface, an important feature of our method is that it allways guarantees a watertight output which is not the case for Wand et al. [32].

Discussion and Limitations We observed above that volume is not perfectly preserved by our reconstruction. There are several reasons for this. The bounded speed assumption (Section 3), which might appear to be strictly enforced by our formulation, is not perfectly satisfied by several of the inputs we generated, with data moving more than one-cell distance between frames. Also, for computational reasons and to guarantee a valid characteristic function, we clamp material values to zero and one, potentially violating the incompressibility constraints in the process. Finally, as described in Section 5, the convergence tolerance, chosen to

The corresponding work was published in the proceedings of Siggraph Asia 2008: Transaction on Graphics, volume 27, number 5, paper 110, ACM 2008.

Data set	Frames	Avg. ppf	initialized
Running man	30	21600	24.8%
Wooden doll	30	8000	20.7%
Pants	280	10100	29.8%
Puppet	200	34000	28.5%
Dress	130	31000	30.1%
T-shirt	160	35000	27.1%
Sweater	70	48000	27.3%
Elephant	20	40000	22.4%

Table 1: *Number of frames, average number of boundary points per frame (ppf), and number of cells initialized over the whole volume for the 3D×time data sets presented.*

provide a sensible balance between accuracy and computational cost, is significantly larger than the machine roundoff error.

While these considerations do mean that our solution does not precisely satisfy the assumptions of the model, this flexibility in fact proves to be extremely useful. We can handle inputs that do not strictly meet the speed assumption or for which the volume does vary. Thus, when the constraints cannot be satisfied exactly, they are satisfied in a least squares sense. Nevertheless, in all our tests the iterative solver reached a tolerance that maintains several decimal digits of accuracy. Indeed, in all our experiments we have found that the mass conservation error, measured as the square root of mass variance across the frames, was always less than 7% for the 2D examples and less than 2% for the 3D examples. The worst example in 2D is shown in Figure 7, in which the object does indeed experience a small change in mass when it splits.

While our post-processing step produces a generally smooth 4D solid, the final 3D meshes representing the time-slices of the solid are computed independently. Thus the connectivity can change between consecutive frames. In low resolution, this can cause some visual flickering. However, as the volume is refined, this flickering becomes less noticeable, and it eventually disappears in high resolution. Algorithms for temporally coherent meshing are an interesting research direction.

7 Conclusions

We have presented a novel volumetric approach for space-time surface reconstruction that leverages the knowledge of object behavior across time, to plausibly reconstruct the deforming surface despite persistent occlusions. Based on reasonable physical assumptions we formulate the reconstruction problem as a solution to a space-time formulation linking together mass and flow across the entire data sequence. We provide an efficient mechanism to solve the resulting optimization problem and demonstrate the viability of our approach on a set of complex examples. We compare our reconstruction results to those generated by extending two 3D reconstruction techniques to 2D×time (RBF and FEM) and 3D×time (FEM). A key idea in our work is to handle the time dimension very differently than the spatial dimensions using several flow priors. Hence, our method correctly completes gaping holes in the data which persist across multiple frames.

Despite the use of a specifically tailored solver, our method is still not suitable for tackling very large scale problems. One avenue for future work which will make it possible to solve problems with much higher resolution would be to consider a hierarchical or domain decomposition approach for optimizing the flow formulation. Specific applications such as modeling moving humans [8, 1] could be enhanced by incorporating additional assumptions, such as skeletal structure or rigidity, within the framework of material flow.

Acknowledgments

We would like to thank the following people for sharing their data with us. Puppet data was scanned with the help of Alexander Bogomjakov, Technion Israel. Scans of moving garments were obtained from Derek Bradeley, UBC. Synthetic scans of wooden doll and running man were taken from the Utah 3D Animation Repository.

References

[1] E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor and H.-P. Seidel. Video-driven animation of human body scans. In *3DTV*, pages 1–4. IEEE, 2007.

[2] B. Allen, B. Curless and Z. Popovic. Articulated body deformation from range scan data. In *Siggraph*, pages 612–619. ACM, 2002.

[3] D. Anguelov, D. Koller, H.-C. Pang, P. Srinivasan and S. Thrun. Recovering Articulated Object Models from 3D Range Data. In *Uncertainty in Artificial Intelligence*, pages 18–26. Morgan Kaufmann, 2004.

[4] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers and J. Davis. SCAPE: shape completion and animation of people. *Transactions on Graphics*, 24(3):408–416, 2005.

[5] N. Anuar and I. Guskov. Extracting Animated Meshes with Adaptive Motion Estimation. In *Vision, Modeling, and Visualization*, pages 63–71. IOS, 2004.

[6] J. L. Barron and N. A. Thacker. Tutorial: Computing 2D and 3D Optical Flow. Technical Report 012, Tina Memo, 2004.

[7] D. Bradley, T. Popa, A. Sheffer, W. Heidrich and T. Boubekeur. Markerless garment capture. *Transactions on Graphics*, 27(3):1–9, 2008.

[8] J. Carranza, C. Theobalt, M. Magnor and H.-P. Seidel. Free-viewpoint video of human actors. In *Siggraph*, pages 569–577. ACM, 2003.

[9] P. Fong and F. Buron. Sensing Deforming and Moving Objects with Commercial Off the Shelf Hardware. In *Computer Vision and Pattern Recognition*, page 101. IEEE, 2005.

[10] G. Golub and C. Greif. On solving block structured indefinite linear systems. *Journal on Scientific Computing*, 24(6):2076–2092, 2003.

[11] B. Goldlücke and M. Magnor. Spacetime-continuous geometry meshes from multi-view video sequences. In *International Conference on Image Processing*, pages 625–628. IEEE, 2005.

[12] B. Goldlücke, I. Ihrke, C. Linz and M. Magnor. Weighted Minimal Hypersurface Reconstruction. *Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1194–1208, 2007.

[13] C. Greif and D. Schötzau. Preconditioners for Saddle Point Linear Systems with Highly Singular (1,1) Blocks. *Electronic Transactions on Numerical Analysis*, 22:114–121, 2006.

[14] E. Grosso, G. Sandini and M. Tistarelli. 3D object reconstruction using stereo and motion. *Systems, Man and Cybernetics*, 19(6):1465–1476, 1989.

[15] N. C. Gupta and L. N. Kanal. 3D Motion Estimation from Motion Field. *Artificial Intelligence*, 78(1-2):45–86, 1995.

[16] I. Guskov, S. Klibanov and B. Bryant. Trackable surfaces. In *Symposium on Computer Animation*, pages 251–257. ACM/Eurographics, 2003.

[17] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, 1998.

[18] P. Holland and R. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813 – 827, 1977.

[19] T. P. Koninckx and L. J. van Gool. Real-Time Range Acquisition by Adaptive Structured Light. *Transactions on Pattern Analysis and Machine Intelligence*, 28(3):432–445, 2006.

[20] A. Levin, R. Fergus, F. Durand and W. Freeman. Image and depth from a conventional camera with a coded aperture. In *Siggraph*, page 70. ACM, 2007.

[21] M. Magnor and B. Goldlücke. Spacetime-Coherent Geometry Reconstruction from Multiple Video Streams. In *3D Data Processing, Visualization and Transmission*, pages 365–372. IEEE, 2004.

[22] S. R. Marschner, B. K. Guenter and S. Raghupathy. Modeling and Rendering for Realistic Facial Animation. In *Rendering Techniques*, pages 231–242. Eurographics, 2000.

[23] N. J. Mitra, S. Flüry, M. Ovsjanikov, N. Gelfand, L. J. Guibas and H. Pottmann. Dynamic geometry registration. In *Symposium on Geometry Processing*, pages 173–182. ACM/Eurographics, 2007.

[24] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H.-P. Seidel. Multi-level partition of unity implicits. In *Siggraph*, pages 463–470. ACM, 2003.

[25] Y. Pekelny and C. Gotsman. Articulated object reconstruction and motion capture from depth video. In *Eurographics*, page *to appear*. Eurographics, 2008.

[26] S. Rusinkiewicz, O. A. Hall-Holt and M. Levoy. Real-time 3D model acquisition. In *Siggraph*, pages 438–446. ACM, 2002.

[27] A. Sharf, T. Lewiner, G. Shklarski, S. Toledo and D. Cohen-Or. Interactive topology-aware surface reconstruction. In *Siggraph*, page 43. ACM, 2007.

[28] M. Shinya. Unifying Measured Point Sequences of Deforming Objects. In *3D Data Processing, Visualization and Transmission*, pages 904–911. IEEE, 2004.

[29] S. Ullman. The interpretation of structure from motion. *Biological Sciences B*, 203(1153):405–426, 1979.

[30] S. Vedula, S. Baker, P. Rander, R. T. Collins and T. Kanade. Three-Dimensional Scene Flow. In *International Conference on Computer Vision*, pages 722–729. IEEE, 1999.

[31] Y. Wang, M. Gupta, S. Z. 0002, S. Wang, X. Gu, D. Samaras and P. Huang. High Resolution Tracking of Non-Rigid 3D Motion of Densely Sampled Data Using Harmonic Maps. In *International Conference on Computer Vision*, pages 388–395. IEEE, 2005.

[32] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. J. Guibas and A. Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Symposium on Geometry Processing*, pages 49–58. ACM/Eurographics, 2007.

[33] R. White, K. Crane and D. A. Forsyth. Capturing and animating occluded cloth. In *Siggraph*, page 34. ACM, 2007.

[34] L. Zhang, B. Curless and S. M. Seitz. Spacetime Stereo: Shape Recovery for Dynamic Scenes. In *Computer Vision and Pattern Recognition*, pages 367–374. IEEE, 2003.

[35] L. Zhang, N. Snavely, B. Curless and S. M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *Transactions on Graphics*, 23(3):548–558, 2004.

[36] S. Toledo. TAUCS: A library of sparse linear solvers, 2003. version 2.2.

The corresponding work was published in the proceedings of Siggraph Asia 2008: Transaction on Graphics, volume 27, number 5, paper 110, ACM 2008.

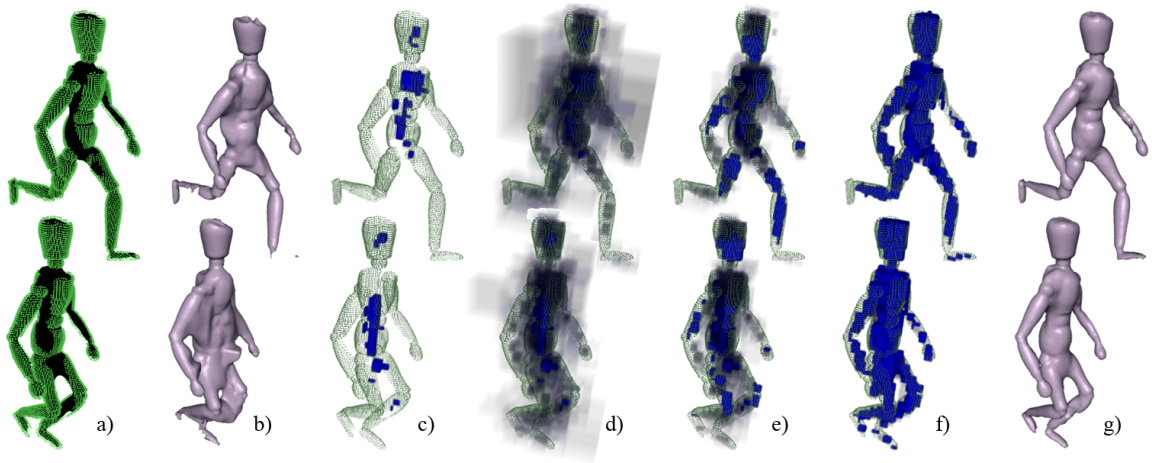

Figure 11: *Space-time surface of a 3D wooden doll's walking cycle. Each row represents a particular frame from the cycle. (a)Points captured by the scanner. (b) The result of a basic FEM reconstruction. (c) Initial "inside" cell labeling. (d-f) Three solver iterations showcasing mass concentration within the surface as the solver progresses. (g) Final reconstruction.*

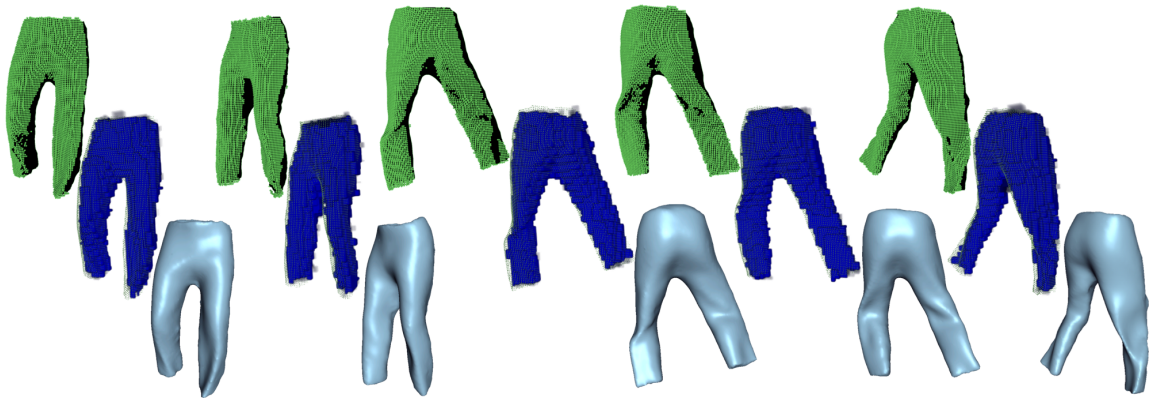

Figure 12: *Reconstruction of jumping pants from a motion capture sequence.* Top*: The points used for the reconstruction.* Center*: The mass field obtained at the final iteration of our flow solver.* Bottom*: our reconstruction.*

Figure 13: *Excerpts from reconstruction sequences of various moving garments captured from multiple video cameras.* Left*: the captured points. We show zoom-ins of irregular sampling, holes and outliers on the leftmost column.* Right*: our reconstruction.*