Copyright notice: (c) ACM, (2009).
This is the author's version of the work.
It is posted here by permission of ACM
for your personal use. Not for redistribution.
The definitive version is to be published in
ACM Transaction on Graphics, to appear.

Efficient Reconstruction of Non-rigid Shape and Motion from Real-Time 3D Scanner Data

MICHAEL WAND
Saarland University and Max Planck Institut Informatik Saarbrücken

BART ADAMS
Stanford University and Katholieke Universiteit Leuven

MAKSIM OVSJANIKOV
Stanford University

ALEXANDER BERNER, MARTIN BOKELOH, PHILIPP JENKE
University of Tübingen, WSI/GRIS

LEONIDAS GUIBAS
Stanford University

HANS-PETER SEIDEL
Max Planck Institut Informatik Saarbrücken

ANDREAS SCHILLING
University of Tübingen, WSI/GRIS

November 30, 2008

Abstract

We present a new technique for reconstructing a single shape and its non-rigid motion from 3D scanning data. Our algorithm takes a set of time-varying unstructured sample points that show partial views of a deforming object as input and reconstructs a single shape and a deformation field that fit the data. This representation yields dense correspondences for the whole sequence, as well as a completed 3D shape in every frame. In addition, the algorithm automatically removes spatial and temporal noise artifacts and outliers from the raw input data. Unlike previous methods, the algorithm does not require any shape template but computes a fitting shape automatically from the input data. Our reconstruction technique is

based upon a novel topology aware adaptive sub-space deformation technique that allows handling long sequences with high resolution geometry efficiently. The algorithm accesses data in multiple sequential passes, so that long sequences can be streamed from hard disk, not being limited by main memory. We apply the technique to several benchmark data sets, increasing the complexity of the data that can be handled significantly in comparison to previous work, while at the same time improving the reconstruction quality.

Categories and Subject Descriptors: I.3.7 [Computer Graphics] Three-Dimensional Graphics and Realism - *Animation*; I.4.8 [Image Processing and Computer Vision] Scene Analysis - *Surface Fitting*

Keywords: Deformation Modeling, Digital Geometry Processing, Surface Reconstruction, Animation Reconstruction

1 Introduction

Recently, a variety of techniques have been developed to capture geometry from real-world scenes in real-time. This allows for recording moving 3d geometry, which has a large number of very interesting applications in areas such as creating special effects for feature films or modeling animated characters for interactive applications. A number of different techniques have been proposed to perform the acquisition task: High resolution geometry can be acquired at high frame rates using active illumination techniques such as motion compensated structured light (Weise *et al.*, 2007; König & Gumhold, 2007) or active space-time stereo (Davis *et al.*, 2005; Zhang *et al.*, 2003). Other approaches include passive multi-view stereo (Carranza *et al.*, 2003; Würmlin *et al.*, 2002; Zitnick *et al.*, 2004) and hardware based time-of-flight methods (such as (PMD, n.d.; MESA, n.d.)), which offer more flexibility at the expense of significant acquisition noise.

Despite the amazing advances on the acquisition side, all of the available techniques suffer from a number of problems. First, the acquisition is limited by occlusions so that only a partial view of the geometry can be acquired at each time instance. This problem is particularly pronounced when structured active lighting is used, which is the technique that currently produces the highest resolution results. A second, related problem is that the scanning devices only record unrelated sample points of the moving surface with no correspondence information across frames. Therefore, reassembling a complete model is not straightforward and further editing and processing operations on the moving geometry are limited. In addition, the data is typically distorted by noise and outliers of varying degree, which imposes an additional robustness challenge on the reconstruction algorithm.

In this paper, we present a new reconstruction system that retrieves this missing information: The algorithm takes a sequence of point clouds sampled at different time instances as input and automatically assembles them to a common shape

that best fits all of the input data frames. At the same time, a deformation field is computed that deforms this shape to match all the data frames at the different time intervals. This factorization into shape and deformation gives a plausible shape completion for all input frames and provides dense correspondences between all frames. The algorithm is robust to noise and outliers and works on long sequences of real-world data.

Our algorithm does not require an a priori known template model, but builds such a model (and its deformation) on the fly during optimization, similar to (Wand *et al.*, 2007). However, we improve upon this work with following main contributions:

- We parameterize and optimize shape and motion separately resulting in considerable gains in reconstruction quality and performance.
- We propose a novel topology-aware adaptive subspace deformation technique that encodes the motion of the object compactly and allows efficient optimization.
- We propose a memory efficient streaming algorithm that accumulates data-driven constraints in local error quadrics, leading to a main memory working set that is constant with respect to sequence length.
- Our deformation model is adapted automatically to the observed data, which allows us to handle a range from elastic to semi-articulated examples.

We apply the technique to reconstruct several data sets from real-time scanning devices. In comparison to the previous state of the art, we obtain a significantly improved reconstruction quality both in terms of the resulting shape and its deformation and our algorithm is able to handle significantly longer sequences without running into resource limits of standard PC hardware.

2 Related Work

In this section, we give a brief overview of related work in reconstructing correspondences of time variant geometry. In addition, we briefly review related work in the area of deformation modeling and compare it to the technique employed in our system.

2.1 Animation Reconstruction

Most existing approaches for reconstructing moving geometry are based on template surfaces, which means that the user has to provide a rough a priori model of the acquired geometry. Our approach is more general than these techniques

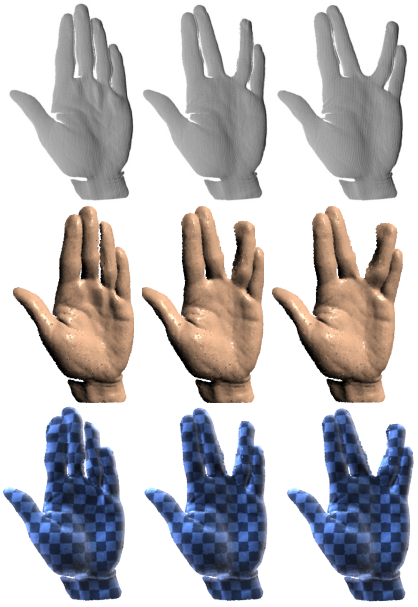

Figure 1: Example reconstruction results obtained with our algorithm from a sequence of input scans of a deforming hand; **left:** original data points (3 example frames out of 32), **middle:** Reconstructed geometry, **right:** reconstructed correspondences indicated by a chessboard texture, with texture coordinates computed from correspondences.

in the sense that it builds the template model automatically as part of the reconstruction process. For example, (Carranza *et al.*, 2003) use template based tracking in the context of a multi-camera passive stereo system. (Anuar & Guskov, 2004) propagate templates through multiple frames using hierarchical optical flow. The "spacetime" faces system of (Zhang *et al.*, 2004) fits a face template mesh to a space-time stereo sequence to enable further editing and inverse-kinematics operations. A sophisticated method for tracking a scanned template mesh according to multi-view stereo and silhouette data has recently been presented in (de Aguiar *et al.*, 2008). The global optimization problem of finding correspondences between shapes is considered in (Anguelov *et al.*, 2005a). The algorithm can match a template to a partial scan without initialization. Some recent papers attack the problem by coupling the reconstruction technique tightly to the acquisition system: (Sand *et al.*, 2003) propose a system that combines multi-view silhouettes and motion capturing. Geometry is accumulated as offset information to a template skeleton. (Park & Hodgins, 2006) generalize motion capturing to a large number of markers. These techniques show impressive results, but depend on the specific acquisition setup and, again, on available a priori model information. Similarly, approaches like SCAPE (Anguelov *et al.*, 2005b), and the automatic registration techniques by (Allen *et al.*, 2002; Allen *et al.*, 2003) use a user supplied template to parameterize the shape matching.

Only very few techniques have been proposed that do not rely on an a priori model but reconstruct it from data: (Pekelny & Gotsman, 2008) propose an algorithm that iteratively registers depth video frames using the ICP algorithm (Besl & McKay, 1992). Their method is fast and simple to implement, however, it relies on a presegmentation by the user and is restricted to objects that undergo articulated rigid motion only. (Mitra *et al.*, 2007) propose a local technique that uses a small window of multiple frames in each step to register long sequences of rigidly moving objects into a common reference shape without need for an additional global error distribution step. The resulting technique is very fast, but restricted to articulated motion. General deformations can only be handled approximately and only in the case of very slowly deforming objects. Non-rigid motion causes artifacts and reconstruction fails in case of partially missing geometry. There has also been work in computer vision, trying to fit a deformable model directly to image data: (Carceroni & Kutulakos, 2002) use a surfel based representation with a general reflectance model and (Starck & Hilton, 2005) use a spherical parameterization of the shape to establish correspondences. However, these approaches do not use a full geometric deformation model. Tracking of meshes based on feature matching and Laplacian diffusion has been considered in (Ahmed *et al.*, 2008; Varanasi *et al.*, 2008). Most closely related to our approach is the method proposed in (Wand *et al.*, 2007), which fits a graph of surfel trajectories to moving geometry. As discussed in the introduction, the method proposed in this paper uses a similar processing pipeline but overcomes the main drawbacks of their method: the simple uniform deformation model in which deformation information is computed for every surfel in

every frame of the geometric shape being reconstructed. This leads to very high computational costs so that neither long sequences (beyond 20-30 frames) nor high resolution geometry (more than a few thousand surfels) can be handled[1]. As shown in the Results Section, our new technique is capable of handling sequences of up to 200 frames and a geometric resolution up to 20,000 surfels per frame, which constitutes a substantial improvement. Our new formulation leads to better conditioned optimization problems, which allows us to improve the reconstruction quality despite of using a more compact representation.

Recently, some extensions have been proposed to the method of (Wand *et al.*, 2007): (Süssmuth *et al.*, 2008) combine a similar approach with fitting an implicit 4D-spacetime surface to the data first, which improves the fitting accuracy to the data by integrating information over multiple time steps (Mitra *et al.*, 2007). This improvement is orthogonal to the contributions of this paper. (Sharf *et al.*, 2008) replace the elastic regularizer by the weaker assumption of volume preservation, which still allows for filling in holes in sequences but not for a reconstruction of global correspondences.

2.2 Deformation Modeling

An important component of our approach is a deformation module that infers plausible deformations of a given shape over multiple frames. Using a set of externally specified correspondence constraints spanning multiple frames, the goal of this module is to infer the best deformation field satisfying these constraints (see Section 5 for details). Although little work has been done to tackle this particular problem (an example of one such method, which only works for meshes, is (Xu *et al.*, 2007)), there exists a rich body of literature aimed at efficiently deforming a given shape with constraints on one, final frame and thus some of the existing methods potentially could be extended to our setting. In the following, we summarize the most closely related work; for an extensive survey in this field, see for example (Nealen *et al.*, 2005; Botsch & Sorkine, 2008). Deformation techniques can be roughly classified into volumetric and thin shell models (Terzopoulos *et al.*, 1987). In our approach, we use a volumetric model forming a thick layer around our surfaces similar to (Botsch *et al.*, 2006). The rationale for this design choice is that a thin shell model cannot be set up reliably in our setting where data points are initially noisy and without reliable normal information. As we seek a deformation field from a common reference domain onto each data frame, and hence have large deformations, we need to employ a nonlinear deformation model (Sheffer & Kraevoy, 2004; Sumner *et al.*, 2005; Botsch *et al.*, 2006; Huang *et al.*, 2006; Au *et al.*, 2006; Sumner *et al.*, 2007; Shi *et al.*,

[1] The product of per frame surfels and the number of frames cannot be substantially larger than about 50,000 - 100,000. At that point, the methods starts needing more than 2GB of main memory for solving the optimization problems, which is problematic at least within a 32bit environment. Handling high resolution geometry with several hundred frames is definitely out of scope of this method as it would require enormous amounts of main memory and computation time.

2007; Botsch *et al.*, 2007; Zhou *et al.*, 2005; Shi *et al.*, 2006). Of these perhaps most immediately pertinent are (Botsch *et al.*, 2006) in which a prism based shell energy is formulated and solved efficiently, and (Botsch *et al.*, 2007) where a similar elastic energy is extended to rigid volumetric cells. Unfortunately the former is based on thin surface shells; an estimation of these is generally difficult in our setting. The latter provides a simplified deformation field but is both topology unaware and employs an interpolation scheme that results in solving a large sparse linear system making it prohibitively slow in our setting. The method we present is also similar in spirit to (Sumner *et al.*, 2007) where the deformation field is discretized, solved for and interopolated using a sparse topology graph. Although we use a similar paradigm, we avoid estimating the rotation and translation components of the deformation field separately, and employ an interpolation scheme which guarantees first-order consistency, which is not true for (Sumner *et al.*, 2007). Thus we reduce the number of graph nodes needed to approximate the deformation field accurately. Further, for the same number of nodes we reduce the number of unknowns in the optimization procedure making it faster. Finally we present a more efficient topology-aware sampling technique that alleviates the sampling problems of (Sumner *et al.*, 2007) such as under-sampling in regions of high geometry complexity.

Our deformation representation is based on classical meshless finite elements (see (Fries & Matthies, 2003) for a good overview), that were recently introduced in computer graphics for physically based animations (e.g., (Müller *et al.*, 2004; Pauly *et al.*, 2005)). We use a similar formulation, but solve the inverse problem of computing the deformation field from given position constraints. Moreover, we propose a novel adaptive sampling algorithm that takes the connectivity of the original geometry into account to ensure an adequate nodal coupling for topologically complex sequences in which different parts of geometry come in close contact. Despite this added topology check, our deformation method maintains the flexibility of traditional meshless algorithms such as fast adaptive sampling and a smooth and consistent deformation field representation.

Recently, we have employed a similar technique in a companion publication (Adams *et al.*, 2008) for controlling animation sequences by user input. There are a few key differences to this work: In our setting, we control a deformation field by clouds of data points with noise and outliers that constrain the deformation field partially rather than user defined key frames at given time intervals. Accordingly, we use a different adaptation scheme: In (Adams *et al.*, 2008), temporal basis functions are used to reduce the overhead for representing smooth trajectories of solids made of homogeneous, elastic materials in between keyframes. In contrast, we use a spatial adaptation scheme that allows for capturing data with varying motion pattern in different parts of the object, such as near-articulated objects. In addition, in this work we address the problem of assembling the domain of the deformation model automatically from data; the undeformed geometry of the object is in our case not given in advance as part of the input. Afterwards, we have to reconstruct the detailed geometry from multi-frame input, which is also not known in advance.

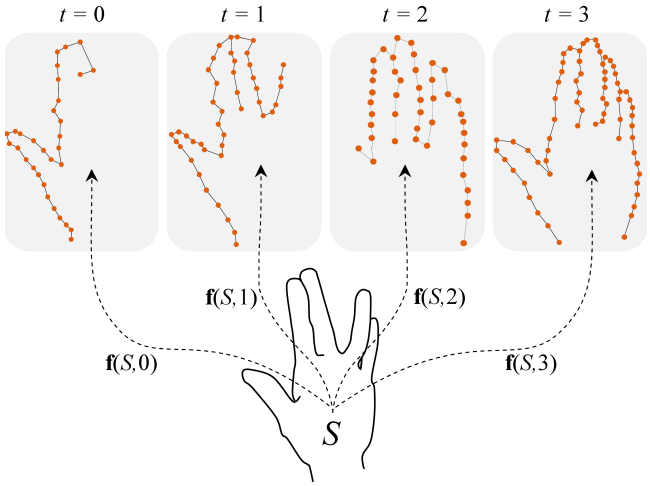

Figure 2: The goal in this paper is to construct one urshape S that represents the geometry of the object and a sparse, but adequate, deformation field $\mathbf{f}$ that properly aligns the urshape to the input data. The urshape S and the deformation field $\mathbf{f}$ are the output of our algorithm and completely define the shape and motion of the acquired object.

3 Overview

Our algorithm expects a sequence of data point clouds as input, denoted by $\mathbf{d}_{i,t} \in \mathbb{R}^3$, where $i \in \{1 \dots n_t\}$ is the index of the data point and $t \in \{1 \dots T\}$ the time at which the measurement was taken. For simplicity, and as this matches all available input data sets, we assume uniform temporal spacing. Our goal is to compute a single two-manifold $S \subset \mathbb{R}^3$, and a time dependent deformation field $\mathbf{f} : S \times \{1 \dots T\} \rightarrow \mathbb{R}^3$, where $\mathbf{f}(\mathbf{x}, t)$ is the deformed position of point $\mathbf{x}$ at time t (see also Figure 2). Following (Wand *et al.*, 2007), we call this common shape an *urshape*. In contrast to that approach however, we solve for a single set of variables representing this shape, not for a separate one in each frame.

3.1 Variational Model

Our goal is now to compute an urshape-deformation pair $(S, \mathbf{f})$ that approximates the data points closely. In the following, we will refer to pairs $(S, \mathbf{f})$ as sequences. To constrain the space of possible solutions, we formulate our goal in a variational framework, where each potential solution $(S, \mathbf{f})$ is scored by an energy function E, which consists of a data fitting and a regularization term

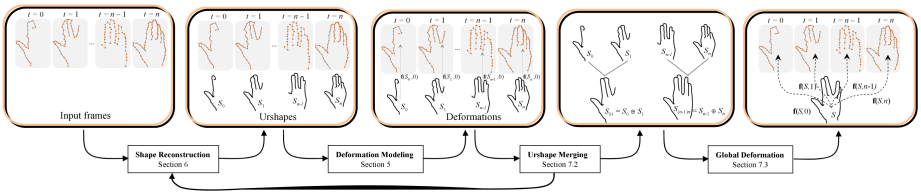

Figure 3: Animation reconstruction pipeline. From the initial input frames, our algorithm first computes urshapes. Next, smooth deformation fields are computed for each sequence. These deformation fields are used to merge adjacent sequences, resulting in updated urshapes that are fed back into the shape reconstruction pipeline until one global urshape and its deformation is found.

that penalizes unreasonable reconstructions (Wand *et al.*, 2007):

$$E(S, \mathbf{f}) = E_{data}(S, \mathbf{f}, \mathbf{d}) + E_{reg}(S, \mathbf{f}). \tag{1}$$

The data term measures the squared distance to the data points:

$$E_{data} = \sum_{t=1}^{T} \sum_{i=1}^{n_t} \text{dist}(\mathbf{f}(S, t), \mathbf{d}_{i,t})^2 \tag{2}$$

We will refine this term later to exclude outlier points that violate a simple Gaussian error model. The regularization energy consists of a weighted sum of five terms:

$$E_{reg} = \underbrace{\omega_r E_{rigid} + \omega_v E_{volume}}_{\text{deformation model}} + \underbrace{\omega_a E_{accel} + \omega_v E_{velocity}}_{\text{general regularization}} + \underbrace{\omega_s E_{smooth}}_{\text{surface rec.}}. \tag{3}$$

The first four terms evaluate the time-dependent deformation field where the first two impose physical constraints of rigidity and volume preservation of the urshape throughout the deformation sequence. The third term penalizes acceleration, to create physically plausible temporal behavior and remove temporal jittering and noise. The fourth term imposes a very small penalty on velocity in order to avoid fluttering artifacts in regions that are not well constraint by data points, such as the shape boundaries. The fifth is concerned with the urshape only; it prefers smooth surfaces to filter out sensor noise and to exclude pathological manifolds S from our reconstruction. As we incrementally add more frames to our reconstruction, this term automatically becomes less dominant and the shape is mostly determined by data points. The weights are user specified constants (see Section 9 for a discussion of parameter setting).

3.2 Optimization Pipeline

In our optimization algorithm, we solve for the two components S and $\mathbf{f}$ separately and alternatingly. When we optimize for $\mathbf{f}$, we assume the urshape S

given, and minimize the data, rigidity, volume, acceleration and velocity energies as discussed above. When we optimize for S on the other hand, we assume that $\mathbf{f}$ is given, and minimize the data and smoothness constraints. The optimization pipeline is depicted in Figure 3. The algorithm first preprocesses all input data, including a preliminary outlier classification. Next, temporally adjacent frames are merged into sequences of the form $(S, \mathbf{f})$ by first computing a common urshape. Given this urshape, we optimize for the deformation field between the two frames. This process is carried on hierarchically, until only one urshape and a global deformation field remains. At every level, both the geometry and the deformation field are optimized globally, to avoid error accumulation.

We now first describe the employed shape and motion discretization (Section 4). Then, we discuss the deformation field optimization (Section 5), the surface optimization (Section 6) and the global matching pipeline in more detail (Section 7).

4 Representation

This section discusses the employed urshape and deformation discretization. These will be used in the following sections for efficient optimization during the reconstruction process.

4.1 Urshape

We represent the urshape S by a set of surface elements $\mathbf{s}_i \in \mathbb{R}^3$, each equipped with a normal estimate $\mathbf{n}_i$. As S is stored only once per sequence, we can afford to employ a simple uniform sampling of S with spacing ϵ_{sampl}, which simplifies the formulation of our algorithm. The spacing ϵ_{sampl} is a user parameter and defines the geometric level of detail at which we reconstruct. The sample points on S are connected by a graph that connects each surfel to its geodesically k-nearest neighbors on S (we typically use $k = 12$ in our examples). We will use $T(\mathbf{s}_i)$ to denote the graph neighbors of surfel $\mathbf{s}_i$ and $|T(\mathbf{s}_i)|$ as the number of those neighbors. To be more robust to outliers, nearest neighbors further away than $3\epsilon_{sample}$ of the surfel set are not connected. One can think of this surfel graph as being a relaxed polygon mesh, in which consistent tessellation is not enforced. This relaxation makes it much easier to update the shape when new data points are examined. We postpone the creation of a consistent mesh to the final stage of the algorithm.

4.2 Deformation Field

The deformation function $\mathbf{f}$ is represented by a graph of non-uniformly sampled points centered at $\mathbf{x}_i \in \mathbb{R}^3$. To distinguish these points from the geometry

points, we will call them nodes in the following. We associate with each such node a deformation vector $\mathbf{u}_{i,t}$ for each of the frames in the animation sequence. These deformation vectors will completely define the deformation field and are the unknowns we will be solving for when optimizing $\mathbf{f}$. Nodes will be created in the vicinity of the given urshape S and their connectivity will be determined by the connectivity graph associated with S. We will discuss the nodal creation, representation and deformation field optimization in more detail in the following section (Section 5).

The sampling of $\mathbf{f}$ and S is strictly independent of each other. Typically S is sampled with more than 10,000 points, while $\mathbf{f}$ is sampled with a few hundred nodes. This discrepancy makes the optimization routine drastically faster and more stable than using the same resolution for both entities. This decoupled representation is one of the main contributions in this work.

5 Deformation Modeling

We represent the deformation on a coarse scale and employ a meshless finite element method to construct appropriate basis functions that are used to interpolate the deformation field, as discussed in Section 5.1. In order to achieve good results even at very sparse deformation sampling, it is crucial to take the connectivity of the original geometry into account when defining the discretization of the deformation field. This topic is addressed in Section 5.2. Finally, in Section 5.3, we describe how this deformation field is optimized using our variational model.

5.1 Deformation Field Representation

We represent the deformation field $\mathbf{f}$ in a discrete manner by assigning per-frame displacement vectors $\mathbf{u}_{i,t}$ to the nodes i. Each node i is defined at a position $\mathbf{x}_i$ with support radius r_i. The displacement vectors $\mathbf{u}_{i,t}$ are the unknowns we will be solving for. The continuous deformation field is then defined using a meshless approximation method (Fries & Matthies, 2003) as:

$$\mathbf{f}(\mathbf{x},t) = \mathbf{x} + \textstyle\sum_j \Phi_j(\mathbf{x})\mathbf{u}_{j,t}, \tag{4}$$

where the scalar shape functions $\Phi_j(\mathbf{x})$ are constructed using moving least squares with a complete linear basis $\mathbf{p}(\mathbf{x}) = [1\ \mathbf{x}]^T$ to guarantee first order consistency (Fries & Matthies, 2003):

$$\Phi_j(\mathbf{x}) = \mathbf{p}(\mathbf{x})^T(\mathbf{M}(\mathbf{x}))^{-1}w(\mathbf{x}-\mathbf{x}_j, r_j)\mathbf{p}(\mathbf{x}_j), \tag{5}$$

where $w(\mathbf{x}-\mathbf{x}_i, r_i) = \max(0, (1-\|\mathbf{x}-\mathbf{x}_i\|^2/r_i^2)^3)$ is a radially symmetric window function with compact support r_i and $\mathbf{M}(\mathbf{x}) = \sum_j w(\mathbf{x}-\mathbf{x}_j, r_j)\mathbf{p}(\mathbf{x}_j)\mathbf{p}(\mathbf{x}_j)^T$ is the 4×4 moment matrix. The moment matrix is non-singular if the evaluation

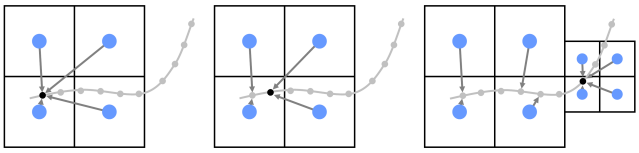

Figure 4: Adaptive grid sampling. The geometry points are processed sequentially (from left to right) and a deformation node is created if necessary in the 8 surrounding grid cells. The black dot indicates the current geometry point. The blue dots are nodes, the arrows point towards the anchor points of the nodes. Anchors are updated for each node to the nearest geometry point. Adaptive sampling creates nodes in smaller grid cells where necessary.

point $\mathbf{x}$ is covered by at least 4 non-planar nodes. In the next section we present a nodal sampling strategy that guarantees this requirement.

In our application, the chosen meshless representation of deformation has a number of advantages. It provides a concise and sound way to define a deformation subspace. Adaptive sampling is easier to implement than with classic finite elements, because no consistent mesh is required. The linear precision allows us to represent rigid body motions exactly with only few basis functions. These occur frequently for real-world dynamics (think for example of a moving hand). The main drawback of meshless methods is the need to invert a small matrix for each evaluation of a shape function. However, because the shape functions are independent of the actual deformation field, they can be precomputed and cached during the optimization.

From the definition of the shape functions, it can be easily seen that nodes influence a spherical region centered at their position. This prohibits correctly resolving topologically complex regions where two geometry patches, albeit close in a Euclidean sense, should be topologically separated. In case of the fingers of a closed hand, for example, we would need an extremely high spatial sampling of the deformation field if we define its influence by distance only, as this is done in previous subspace methods (Müller *et al.*, 2004; Huang *et al.*, 2006; Sumner *et al.*, 2007). In the following section, we propose a novel nodal sampling algorithm that takes the connectivity of the urshape, the domain the deformation function is defined on, into account and correctly prunes nodal influence regions.

5.2 Topology Aware Sampling

Given an urshape and its nearest neighbor surfel graph as defined above, we wish to create a coarse nodal sampling that will represent the deformation field. This sampling has to meet a number of requirements. First, as discussed above, it should be volumetric and within the defined support volume, each surfel should be well constrained, i.e., covered by at least 4 non-planar deformation nodes.

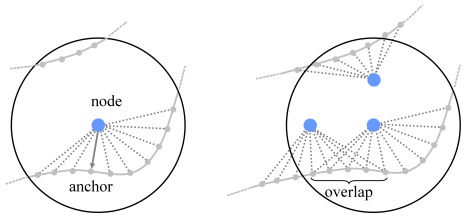

Figure 5: Neighborhood definitions. Left: A node is neighbor to all geometry points that are within its spherical influence region and that can be reached by walking over the topology graph, starting from the node's anchor point without leaving the node's sphere. This correctly discards topologically far geometry patches. Right: A node is neighbor to another node if the former is within the influence region of the latter and if they are both neighbor to a common geometry point. Hence, the top node is not a neighbor of the lower right one, while the lower left one is.

Second, it should be topology aware, creating separate node sets for disconnected regions. Third, the nodal coupling should guarantee continuous shape functions thereby guaranteeing a continuous deformation field. And finally, the sampling algorithm should be efficient as it will be invoked multiple times during animation reconstruction.

Sampling Algorithm: Our sampling strategy works by overlaying the urshape with a set of nodes centered at adaptive grid cells where the cell size depends on the local sampling requirement (we will discuss the adaptation of the sampling density in more detail in Section 7.3). In each grid cell neighboring an urshape surfel, one or more nodes are created depending on how many topologically separate urshape components exist in that region. The actual sampling algorithm works as follows (see also Figure 4): We iterate over all urshape surfels, discretize each point to its nearest grid cell center and create a deformation node if the grid cell is empty. If not, we check whether the node occupying the grid cell is already a neighbor of the current urshape surfel. If this is the case, we can safely move on to the next surfel without creating a node. If the existing node is not a neighbor of the urshape surfel, it must have been sampled from a different geometry patch and thus we have to create a new node. As such, multiple disconnected nodes can be created at the same spatial position. Note that we not only create a node at the quantized geometry position, but also at the 7 other neighboring grid cells surrounding the geometry point surfel. The nodal radius is set to 1.8 times the grid cell size, therefore ensuring that the geometry point is covered by all 8 neighboring nodes. Using this sampling strategy we obtain a volumetric narrow band of N nodes around the geometry.

Neighborhood Definitions: We call the geometry point that creates a node, its anchor point. If multiple points would create the same node, we update the anchor point to the nearest geometry point (Figure 4). The anchor is used to

prune the influence region of each node. Typically, in meshless methods, each node would influence all points within a surrounding sphere centered at the node with radius r_i. However, when different geometry patches are close, such as the fingers of a hand, nodes created from each patch should not influence each other. Hence, we have to take topology into account to define the nodal support regions. Due to the efficiency requirements it is too costly to approximate a smooth geodesic distance on the surfel graph. Moreover, the graph itself provides only C^0-continuous geodesic distance approximations. Therefore, we will use only Euclidean distances to evaluate our window functions. We use the following strategy that guarantees smooth topology-aware shape functions: When a node is created, we only assign it as a nodal neighbor to the geometry points within a Euclidean distance r_i that can be reached from the anchor by walking on the geometry topology graph without leaving a Euclidean sphere of radius r_i (see Figure 5, left). So, all geometry points that are close in Euclidean and topological sense are influenced.

The nodal coupling is also topology aware. A node j is considered a neighbor of node i, if j is within a Euclidean distance r_j to i and if i and j influence a common geometry point (see Figure 5, right). The latter criterion prevents spatially neighboring nodes that are created from topologically distant geometry to incorrectly become neighbors. Also here, we use the Euclidean distance in the weight and shape function computations, guaranteeing their continuity. Note that the only potential situation in which a shape function becomes discontinuous is if a geometry point i is a neighbor of some deformation node n, while another geometry point j in its vicinity is not. But then the edge between i and j must cross the boundary of the Euclidean ball $B(\mathbf{x}_n, r_n)$, thus meaning that n's influence on i is small since $w(\mathbf{x}_i - \mathbf{x}_n, r_n) \approx 0$, implying continuity.

5.3 Deformation Field Optimization

Having obtained a valid nodal sampling, we have to define and discretize the data fitting term and the regularization energies for the deformation function as discussed in Section 3.1. Note again that our goal is to find optimal nodal deformations $\mathbf{u}_{j,t}$ for all frames t.

Data Matching: Following our reconstruction model (Equation 1), we have to minimize the distance between the deformed urshape and the data points. This can be achieved by specifying position constraints that restrict certain points $\mathbf{x}$ on the urshape to deform to a goal position $\mathbf{y}$ in a specified frame t. Hence, the position constraints try to satisfy $\mathbf{f}(\mathbf{x}) = \mathbf{y}$. We will discuss in Section 7.2 how these position constraints are defined. In general, this leads to the following data matching energy (for a single constraint):

$$E_{pos} = (\mathbf{f}(\mathbf{x}, t) - \mathbf{y})^T \mathbf{Q} (\mathbf{f}(\mathbf{x}, t) - \mathbf{y}), \tag{6}$$

where $\mathbf{Q}$ is a symmetric positive definite matrix that forms an error quadric, allowing varying constraints penalties in different directions. In particular, this

includes rank-degenerate constraints acting in one or two directions only. Using Equation 4, this expands to:

$$E_{pos} = (\mathbf{x} + \sum_j \Phi_j(\mathbf{x})\mathbf{u}_{j,t} - \mathbf{y})^T \mathbf{Q} (\mathbf{x} + \sum_j \Phi_j(\mathbf{x})\mathbf{u}_{j,t} - \mathbf{y}), \tag{7}$$

where the summation is over all nodes j influencing the correspondence's base point $\mathbf{x}$. It is easy to see that all correspondence constraints can be added up and the resulting energy can be written as a quadratic form:

$$E_{data} = c + \sum_t \sum_i b_{i,t}\mathbf{u}_{i,t} + \sum_t \sum_i \sum_j \mathbf{u}_{i,t}^T a_{i,j,t} \mathbf{u}_{j,t}, \tag{8}$$

where there is a linear coefficient $b_{i,t}$ for every node that influences a correspondence base point and a quadratic coefficient $a_{i,j,t}$ for every pair of coupled nodes i and j. Accumulating these coefficients for all the correspondences allows efficient evaluation and differentiation of the correspondence energy during optimization: An arbitrary number of constraints can be represented using $O(k^2NT)$ space, where N is the number of deformation nodes and k the out-degree of their interaction graph (which has bounded degree by construction). This is independent of urshape and data resolution, allowing for an efficient streaming computation.

Rigidity: To regularize the deformation field, i.e., to restrict it to plausible deformations, we add an additional energy term that penalizes non-rigid deformations. The deformation field is rigid if and only if at all points and all frames $\nabla\mathbf{f}^T\nabla\mathbf{f} = \mathbf{I}$. Here, we use a point collocation scheme and penalize non-rigid deformations at the nodal positions:

$$E_{rigid} = \sum_t \sum_i \|\nabla\mathbf{f}(\mathbf{x}_i, t)^T \nabla\mathbf{f}(\mathbf{x}_i, t) - \mathbf{I}\|_F^2, \tag{9}$$

where $\nabla\mathbf{f}(\mathbf{x}, t) = \mathbf{I} + \sum_j \nabla\Phi_j(\mathbf{x})\mathbf{u}_{j,t}$. Expanding the rigid energy yields a polynomial of degree 4 in the nodal displacements.

Volume Preservation: Most real-world deformations locally preserve volume. A deformation field is, by definition, locally volume preserving if at every point in space $\det(\nabla\mathbf{f}) = 1$. Again, using a point collocation scheme, this translates to the volume preservation energy:

$$E_{volume} = \sum_t \sum_i (\det(\nabla\mathbf{f}(\mathbf{x}_i, t)) - 1)^2, \tag{10}$$

which expands to a polynomial of degree 6 in the unknowns. In our application, the volume preservation acts mostly as an additional regularization energy that makes the rigidity criterion stably defined, as E_{rigid} alone is invariant under flips of orientation of the deformation gradient (cf. (Müller *et al.*, 2004)).

Temporal Smoothness: All previous constraints are defined for each frame separately. They restrict the way an urshape is allowed to deform into a certain frame at time t. In a typical real-time scanning setup, neighboring frames are acquired from temporally coherent motion. Hence, we want to restrict the reconstructed deformation field to be temporally smooth as well. This can be

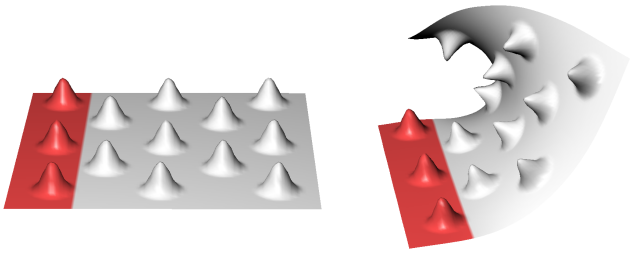

Figure 6: Example obtained using our meshless deformation algorithm. The red points are constrained, while the user deforms the geometry by pulling on one corner of the bumpy plane. As can be seen, the rigidity and volume preservation constraints are faithfully minimized using the non-linear solver and details are preserved during the deformation.

obtained by penalizing acceleration of the deformation nodes. Using simple finite differences this yields the energy function:

$$E_{accel} = \sum_t \sum_i (\mathbf{u}(\mathbf{x}_i, t+1) - 2\mathbf{u}(\mathbf{x}_i, t) + \mathbf{u}(\mathbf{x}_i, t-1))^2, \tag{11}$$

for every node i and (non-boundary) frame t. Here we assume that consecutive frames are acquired at constant time intervals. Similarly, as discussed above, we constrain the velocity of a node over neighboring frames by minimizing the energy:

$$E_{velocity} = \sum_t \sum_i (\mathbf{u}(\mathbf{x}_i, t+1) - \mathbf{u}(\mathbf{x}_i, t))^2. \tag{12}$$

Numerical Optimization: For the optimization, we employ a standard quasi-Newton BFGS solver (Avriel, 2003). We can reduce the memory requirements by employing a non-linear conjugate gradients solver (Shewchuk, 1994), which needs gradient information only, but this turns out to be less efficient in practice for our application. For stiff settings (high rigidity), the run-time disadvantage can be quite substantial (comparing to non-linear CG with diagonal preconditioning). Independent of the solver employed, we can reduce the memory requirements by employing a sliding window strategy in the temporal domain: As easy to see, only acceleration and velocity energies couple multiple frames of our animation. Typically, their influence will be rather local as our problem is mostly constrained by data points. Using this observation, we can solving for only a small window of variables at a time, corresponding to $3-5$ frames at once, and then offset by one frame and iterate. One can think of the local optimization as a local smoothing filter. This strategy requires only a single, sequential scan through all variables so that it is still efficient if the variables do not fit into main memory. We have implemented an explicit paging scheme to support this: It turns out that the quadratic interactions of the variables describing the quadratic data constraints consume a considerable amount of memory, while the state variables themselves usually fit into main memory. Therefore, we employ a LRU-swapping scheme to only fetch the quadrics representing the

constraints (Equation 8) from disk for the frames in consideration. In our whole optimization pipeline, creation of constraints and optimization will be strictly time sequential, so that this scheme comes at a very small additional overhead while extending the size of sequences that can be handled significantly. Using a sliding window does not only improve memory consumption but also the computation time, as the runtime requirements of the solver grow superlinearly with the number of variables in consideration. The sliding-window approach does not compute an optimal solution, but in practice, we did not observe significant artifacts.

Example: Figure 6 illustrates that our deformation framework is capable of faithfully deforming geometry while preserving surface details. As we are using a full non-linear deformation model, rotations within the deformation are handled correctly. The subspace approach makes sure that we still obtain a high performance, despite using an expensive non-linear finite element model.

6 Shape Optimization

In addition to optimizing the deformation field, we also optimize the geometry of the urshape, in a separate step. We follow a simplified version of the strategy of (Huang *et al.*, 2007), but we use an approximate, purely quadratic objective function that can be optimized by just solving a linear system. Basically, we allow urshape surfels to move in normal direction and allow the normal tip to translate in tangent direction. For, this we first need to fix a reference frame for each surfel $\mathbf{s}_i$ with orthonormal coordinate axis $(\mathbf{u}_i, \mathbf{v}_i, \mathbf{n}_i)$ centered at $\mathbf{s}_i$, where $\mathbf{n}_i$ points roughly into the direction of the final surface normal. We compute this initial estimate by a principal component analysis of a larger topological neighborhood of radius $4\epsilon_{sampl}$.

Given these local frames we try to find the updated surfel position $\mathbf{s}'_i = \mathbf{s}'_i + t^n_i \mathbf{n}_i$ and the updated (unnormalized) normal direction $\mathbf{n}'_i = \mathbf{n}_i + t^u_i \mathbf{u}_i + t^v_i \mathbf{v}_i$, where the scalars t^n_i, t^u_i and t^v_i are the unknowns we will be solving for.

6.1 Data Attraction

Similarly to the deformation field optimization, we require the updated urshape to be close to the data. To model data attraction, we allow again for general position constraints that attract a surfel $\mathbf{s}_i$ to a point $\mathbf{y}$. In essence, we try to minimize the point-to-plane distance from the point $\mathbf{y}$ to the plane through $\mathbf{s}'_i$ with normal $\mathbf{n}'_i$. However, to ensure a quadratic penalty function in the unknowns, we take the distance along the undeformed normal $\mathbf{n}_i$, giving:

$$E_{pos} = \frac{(\mathbf{s}'_i - \mathbf{y})^T \mathbf{n}'_i}{\mathbf{n}_i^T \mathbf{n}'_i} \mathbf{n}_i^T \mathbf{P} \frac{(\mathbf{s}'_i - \mathbf{y})^T \mathbf{n}'_i}{\mathbf{n}_i^T \mathbf{n}'_i} \mathbf{n}_i, \tag{13}$$

which can be simplified (note that $\mathbf{n}_i^T \mathbf{n}_i' = 1$) to:

$$E_{pos} = (\mathbf{n}_i^T \mathbf{P} \mathbf{n}_i)((\mathbf{s}_i' - \mathbf{y})^T \mathbf{n}_i')^2, \tag{14}$$

which is quadratic in the unknowns (because $\mathbf{n}_i^T \mathbf{u}_i = 0$ and $\mathbf{n}_i^T \mathbf{v}_i = 0$ by construction).

We specify position constraints by transforming the data points $\mathbf{d}_{i,t}$ into the urshape domain using the inverse deformation field $\mathbf{f}^{-1}$ and associate one such point-to-plane constraint between each data point and its nearest (updated) urshape surfel point, say $\mathbf{s}_j$. The total data fitting energy becomes:

$$E_{pos}(i,t) = (\mathbf{n}_j^T \mathbf{J}_{i,t}^T \mathbf{P}_{i,t} \mathbf{J}_{i,t} \mathbf{n}_j)((\mathbf{s}_j' - \mathbf{f}^{-1}(\mathbf{d}_{i,t}, t))^T \mathbf{n}_j')^2, \tag{15}$$

where $\mathbf{J}_{i,t} = \nabla \mathbf{f}^{-1}(\mathbf{d}_{i,t}, t)$ is the Jacobian of the inverse deformation function. We associate with the data points an error quadric $\mathbf{P}_{i,t}$, which can be computed from the covariance of the data. We assume stationary isotropic noise and set it hence to $\mathbf{P}_{i,t} = \mathbf{P} = \mathbf{I}/\epsilon_{data}^2$, where ϵ_{data} is the (global) average sample spacing input. The total data energy then becomes $E_{data} = \sum_t \sum_i E_{pos}(i,t)$. In order to make surface reconstruction robust to outliers, we truncate the quadratic potentials: Whenever a data point, after being transformed to the urshape domain, ends up farer away than $3\epsilon_{sampl}$ from an urshape point, we ignore the corresponding position constraint, which corresponds to truncating the data potentials to a constant value at a distance of $3\epsilon_{sampl}$.

6.2 Smoothness

The smoothness term tries to make surfels lie in the tangent plane of their neighbors and keep neighboring normals similar: $E_{smooth} = E_{plane} + E_{normals}$, where

$$E_{plane} = \sum_{i=1}^{n} \sum_{j \in T(\mathbf{s}_i)}^{|T(\mathbf{s}_i)|} ((\mathbf{s}_i' - \mathbf{s}_j')^T \mathbf{n}_i)^2, \tag{16}$$

$$E_{normals} = \sum_{i=1}^{n} \sum_{j \in T(\mathbf{s}_i)}^{|T(\mathbf{s}_i)|} (\mathbf{n}_i' - \mathbf{n}_j')^2, \tag{17}$$

Here we make an approximation and assume that the reference frames are constant during optimization, leading to quadratic objective functions.

The main advantage of our formulation of surface fitting is efficiency. Again, we can stream through all data points once, project them back, accumulate quadratic potentials on the unknown coefficients, and solve a rather small linear system defined only on three coefficients per urshape point, independent of the number of frames.

7 Reconstruction Algorithm

Using the building blocks described before, we can now formulate our animation reconstruction pipeline. First, we preprocess our data and store the result in swap files on hard disc, from which we can retrieve the data of each frame in one chunk. Then, we run our surface reconstruction module on each input frame separately and create initial shape estimates (see also Figure 7). To make the formulation of the algorithm simpler, we equip it with an identity deformation field and consider it a single frame sequence. Next, we run a recursive sequence merging algorithm that takes two sequences that are adjacent in time as input and unifies them to a single sequence with one urshape and a global deformation function. We execute this merging in a binary tree, merging first single frame, then two frame, four frame sequences and so on. In order to limit the memory demands, we build and traverse this tree depth first, left to right, starting at the first frame. In this way, we have to handle at most $\log_2 T + 1$ urshapes at the same time. In the following, we will describe the steps in more detail.

7.1 Preprocessing

In the preprocessing step, we apply tensor voting (Medioni *et al.*, 2000) to remove outliers. We closely follow the implementation described in (Wand *et al.*, 2007) for the same purpose. The algorithm detects points that do not lie on any subset of data that forms a smooth 2-manifold. This step helps greatly in initially cleaning up the data, but some outliers will remain undetected, in particular structured artifacts caused by 3D scanners due to technological limitations. This will be handled later during sequence merging using robust data potentials (see next subsection). After tensor voting, we create a resampled copy of each data frame with a small fraction of the complexity of the original data by quantizing the points on a regular grid and keeping only one per grid cell. In our examples, we typically use an effective maximum sample spacing of 1/80th of the maximum bounding box side length of the scene. The resolution has been chosen to still depict well the topology of the original data and the rough geometry, which is sufficient for a coarse alignment. We will use the full resolution data later in a final optimization step. This two-resolution approach significantly speeds up the running time of our reconstruction pipeline.

7.2 Sequence Merging

Merging sequences proceeds in four steps: alignment, urshape merging, deformation refitting and global optimization. The basic strategy is similar to the pipeline in (Wand *et al.*, 2007), however, reformulated to separate deformation from shape. For ease of notation, we will denote the first sequence as A and the second as B in the following. We denote the first and last frame, i.e., the urshape deformed by the corresponding deformation field $\mathbf{f}$ at that time, of a sequence X

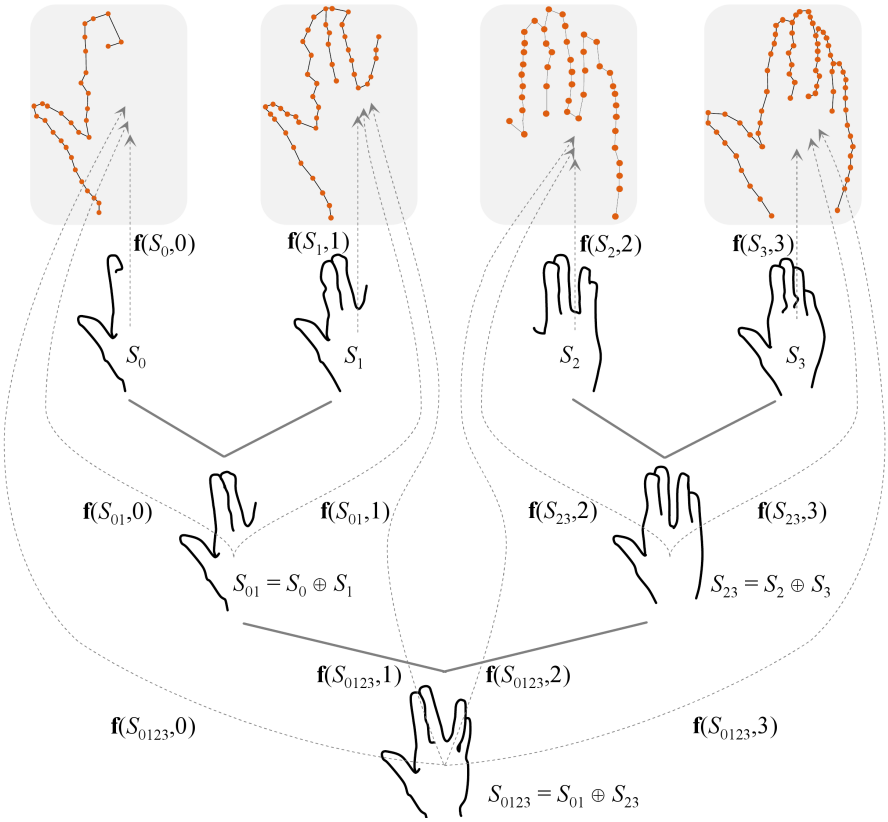

Figure 7: Hierarchical merging of sequences until a final urshape S_{0123} and a deformation field $\mathbf{f}$ is obtained.

by first(X) and last(X), respectively. See also Figure 7 for a schematic overview of the merging pipeline.

Alignment: In the first step, we compute an optimal alignment of last(A) at time t and first(B) at time $t+1$. For this, the second frame creates ICP-like position constraints as input to the deformation optimization as follows. First, we compute the nearest deformed urshape surfel $\mathbf{f}(\mathbf{s}_i, t)$ for each data point in first(B). Then, we setup a position constraint with the data point as target and the urshape point $\mathbf{s}_i$ as origin. We make this a point-to-plane constraint by setting up a rank-1 error matrix in surface normal direction $\mathbf{M} = 1/\epsilon_{data}^2 \mathbf{n}_i \mathbf{n}_i^T$. To be more robust to outliers, we delete points that are too far from the surface. We discard correspondences that point to data points in first(B) further away from the deformed urshape than the 80% percentile of such distances. If the direction of the connection differs by more than 45 degrees from the estimated surface normal, the match is also discarded. To

avoid oscillations, points closer than $3\epsilon_{sampl}$ are never culled. From these ICP constraints we compute an optimal deformation $\mathbf{f}$ as discussed in Section 5, deform the urshape using this deformation and iterate the above procedure until convergence or until a maximum number of 25 iterations has been reached. In order to make the alignment more robust, we adopt the strategy of (Allen *et al.*, 2003): We first perform a rigid merge (setting a weight for the rigidity potential that is $100 - 1000\times$ larger than the final one), and then decrease the rigidity (typically in 2-3 steps in powers of 10). For simple alignment problems, this strategy is not necessary (actually causing a small runtime overhead), but for complex motions with larger temporal spacing, it improves the reliability of the alignment.

Urshape Merging: The surfels of the aligned frames are now concatenated and a k-NN graph restricted to an edge length of $3\epsilon_{sampl}$ is recomputed as introduced earlier. This is a candidate for a new urshape. The new topology will contain a superset of all desired topology edges, because the deformation model guarantees to keep geodesically close points at almost the same distance. However, because we form all connections that are close in a Euclidian sense, naive urshape merging may connect surfels that belong to different connected components (think of a mouth opening and closing in a face scan). We filter false positives by tracing all point pairs from A that have been connected through time. We compute their distance in the whole sequence A and in the new urshape. If the length of an edge changes substantially (over a factor of 1.5) it is removed. The same is done for pairs of points originating from the sequence B.

Deformation Refitting: Up to now, we have computed a new urshape for the compound sequence, but we are still lacking its deformation. To recompute it, we observe that our new urshape consists of points that originate from either sequence A or B. Thus, we create position constraints that move points from A to their original location for all frames involved. Accordingly, we create similar constraints for points from B for the later frames. Then, we use the deformation optimization module to recompute a deformation sequence. This step fills in an interpolated or extrapolated motion for points of the common urshape that have not been modeled previously in the newly introduced part. Interpolation happens in regions where the urshapes from A and B overlap. In disjoint areas, we obtain an extrapolation with minimal deformation.

Global Optimization: We now have stitched together the two sequences, but small errors may have occurred during this process. If we repeatedly perform merges, these errors might accumulate to larger mismatches. Therefore, we reoptimize shape and deformation again: First, we perform a resampling step that deletes points on the new urshape closer to other urshape points than ϵ_{sampl} and still covered by neighboring points to keep the sampling density constant over the optimization. Then, we run the shape optimization module to update our shape estimate, using data points from all frames involved. Next, we recompute the deformation field $\mathbf{f}$ using only ICP-like constraints based on

our previous initialization. To make the process robust to structured outliers, we also delete urshape points that are only observed (close to data points) in a few frames (typically, less than 3 or below 10%) if our merged sequence is already long enough (typically at least 16 frames) to make this decision. This whole process is reiterated at every recursion level of the merging; each iteration requires a sequential scan through our input data.

Urshape Propagation: A serious problem of the divide-and-conquer sequence merging algorithm is that it starts from scratch at every merging operation. In particular at the lowest level of the tree, where single frames are merged, it might easily happen that the frames involved contain only a few data points with many disconnected components. Pairwise matching with deformable ICP is then severely ill posed. To counteract this, we propagate our current urshape estimate through the merge tree: When we traverse the tree from left to right, we always finish all possible subtrees to the left before picking the next unseen frame to the right. Whenever we pick a new frame that has been previously not observed we align the rightmost frame from the highest available hierarchy level on the left with the new frame on the right. Only after this, we start the initial merge of this frame with its immediately following frame to the right, as dictated by the binary merge tree. The additional points are carried through the whole optimization pipeline as shape guess that is not (yet) supported by data points. After a few frames of urshape propagation, we have usually accumulated a well behaved shape with large connected components. This strategy does not help if we start merging at a time step at which the geometry is very sparsely acquired and fragmented into many parts. We do not require a complete starting shape, but such fragmented situations still need to be avoided at the starting frames to prevent convergence problems in the alignment. An automatic strategy for picking good starting frames and merging in different time directions is currently still subject to future work.

7.3 Final Global Optimization

After completing the merging pipeline, we perform an additional step of global optimization. In this step, we employ the full original data resolution (i.e., all original data points not classified as outliers). In order to be able to represent the higher resolution data adequately, we upsample the urshape obtained from hierarchical sequence merging. For upsampling, we performing repeated splits of the topology edges until each point has a neighboring point within the new sample spacing ϵ_{sampl}; in practice, we typically choose half the original sample spacing (quadrupling the resolution). Then, we run the global optimization step (for shape and deformation) again. At this point, we set higher penalties on deformation and acceleration to obtain stable results. However, larger deformation penalties lead to artifacts in strongly deforming areas, such as clothing or joints.

Adaptive sampling: To counteract this problem, we can use the information

gathered so far in the urshape. For this, we determine the area where the motion cannot be well represented by the uniformly sampled deformation model: We scan through all (non-outlier) data points again and compute the residual vectors between data points and closest urshape surfels. We accumulate signed point-to-plane errors for all surfels. We use signed distances in order to distinguish between noise (which is evenly likely to deviate in both directions) and structured areas of undersampled deformation (where deviations are correlated over time and space and thus typically point into the same direction). Subsequently, we extract the information for the adaptive sampling from this representation: First, we apply a median filter to remove noise and outliers from the residuals. We use a very large filter radius: The radius is a fixed fraction of the initial deformation node support radius (typically $0.2\times$), as we are aiming at extracting information at that level of resolution. Next, we use the filtered residuals to refine our deformation model. This is done by quantizing the filtered residual values. We use a simple heuristic to control the quantization process automatically (Figure 8 shows examples the outcome of the quantization): We determine the 75% percentile of the residuals and map these to a refinement level of zero, corresponding to the original resolution. For the remaining values, we compute a linear map that maps the 95% percentile to a level of two and interpolate in between, clamping at level two, which corresponds to 1/4 nodal spacing, or 16 times the original resolution (with respect to a surface sheet). The rationell for this choice is the assumption that most of the data is already adequately sampled (75%) and that there might be a few outliers (top 5%). In between, the resolution levels are interpolated. In the nodal sampling algorithm (Section 5.2), we use this information to locally reduce the grid cell size. Doing so will have two effects. First, the surface becomes more bendable, as the offset distance is reduced. Second, the sampling rate is increased so that the higher frequency details that now become visible can be represented as well. The proposed parameters have been determined by experimentation and usually work well for typical data sets. They can be fine tuned by the user, if necessary.

8 Implementation & Results

We have implemented the system described in this paper in C++ and ran a number of experiments with real-time 3D scanner data sets. The experiments were conducted on a pool of several standard PCs, using a single system and a single core only per computation. The systems were all roughly comparable (Intel Core2 Duo/Quad systems with 2.4-2.5Ghz, at least 2GB RAM). We perform all our benchmarks in a 32bit environment (Windows XP 32bit) in order to examine the memory efficiency. The employed runtime allows a process to allocated up to 2GB of main memory[2]. The primary input data (taking several

[2]The results can be directly compared to (Wand *et al.*, 2007), who are subject to the same restrictions; we will demonstrate that our algorithm allows for significantly more complex reconstructions using the same maximum memory footprint.

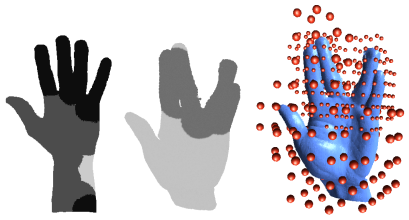

Figure 8: Weights computed by our adaptive sampling strategy for data sets hand-1 and hand-2, and the corresponding sampling. The color corresponds to the factor with respect to the base sample spacing. White means 1.0 (no increase in sampling), light gray 0.5 (two-fold oversampling) and dark gray 0.25 (four-fold oversampling).

gigabyte in some data sets) was stored out of core on the system hard drive; time for data access was negligible in comparison to the optimization costs. Statistics about the results are summarized in Table 1, and screen shots of reconstruction results are shown in Figure 9. All renderings have been produced directly from the point based model, using a GPU implementation of EWA surface splatting (Zwicker *et al.*, 2001), which yields slightly better results than meshing using marching cubes, in particular in boundary regions. For animated results, please refer to the video accompanying this paper.

8.1 Results on Example Data Sets

In the following, we describe the results obtained with our implementation and compare them to previous work. We have applied our algorithm to 6 different data sets, three of which have also been used in (Wand *et al.*, 2007).

"Face": The first is a scan of a facial expression, obtained with the motion compensated real-time structured light scanner of (König & Gumhold, 2007). Our input sequence consists of 79 frames with an overall amount of 23.9 million data points. We have resampled the data set to about 2.4 million overall points (quantized to 0.25% bounding box size) and compute the hierarchical alignment in 304 minutes. Afterwards, we ran an additional optimization step, using the original data in its full resolution (116 min). In the final optimization, we the acceleration weights (for initial merging, acceleration is always set to zero). For the subsequent examples, we use a similar processing pipeline.

"hand-1": Our second example is a hand data set acquired with the system of (Weise *et al.*, 2007). As this is an articulated motion, we use adaptive sampling in the final optimization step. The refinement regions are determined automatically from our hierarchical alignment results are shown in Figure 8: The algorithm automatically concentrates a higher node density in the regions of the fingers that are moving. The same example has also been used in [Mitra

et al. 2007]; in comparison to their approach, our global-optimization-based technique is significantly more expensive. However, we can correctly capture the deforming geometry and fill all acquisition holes reliably.

"hand-2": The second sequence of a hand shows a gesture acquired with a commercial real-time structured light scanner system (ABW-3D). In this example, we use the described topology change detection routine to automatically separate the finger movement. In the video of this sequence, a minor artifact in form of a slightly slurry movement of the fingers is still visible. On closer examination, it turns out that this artifact is not caused by the topology change (as one might expect) but by the limited ability of deformable point-to-plane ICP to align the fingers with high precision. This is due to the fact that the actual geometry is rather flat (only the very front facing part of the fingers is acquired).

"hand-3": The third moving hand sequence is particularly challenging, as only rather small parts of the geometry are visible while the model is strongly deforming. With our locally convergent reconstruction technique, we are able to track the first 120 frames of the sequence reliably (as shown in the video). This is about the limit of what can be dealt with using local deformable ICP as motion tracking algorithm.

"hand-4": The forth sequence shows a hand moving with less deformation but significant rotation. Assembling this sequence is still challenging, as the fingers are only acquired partially and appear rather flat in the input data. Our algorithm recovers the shape correctly up to some artifacts at the thumb, which are due to a few incorrect matches that cause the geometry to appear flatter than expected. This sequence is our longest example, covering over 200 frames. In terms of computation times and memory, this sequence is handled without problems (using roughly 5h of computation time and 0.5GB of main memory; see below for details).

"popcorn tin": The final example shows results for a data set of a person waving a popcorn tin, acquired with a color coded structured light scanner [Fong and Buron 2005]. The particular challenge of this data set is that the original data contains structured noise that forms waves on the surface as well and large outlier surfaces that pop up occasionally. Our algorithm is still able to track the sequence reliably and reconstruct a smooth surface. We can remove the large, structured outliers by the coverage test described in Section 7 (removing urshape parts backed by data with sporadic appearance over time).

Effect of adaptive sampling: For the "popcorn tin" sequence, we have prepared three different versions: using uniform deformation sampling (grid spacing = 1/12 of the bounding box length), adaptive sampling (grid spacing = 1/6 of the bounding box size, refinement up to = 1/24), and an adaptively sampled version with user guidance. The uniformly sampled sequence yields reasonable results but misses out some minute details. For the adaptively sampled version, we only alter the final optimization step (merging is still computed at grid res-

olution 1/12) and reduce the base resolution by a factor of two, but allowing to refine by two subdivision levels. Hence, more rigid regions are represented with less flexibility, reducing noise artifacts, while strongly deforming regions are sampled with higher resolution. Within the limits of 2GB of main memory, it was not possible to handle the full data set at the highest level of resolution, because of the memory requirements of the BFGS solver applied to the deformation nodes within the window size (Table 3 shows the memory consumption). The adaptive computation is more efficient by using a higher resolution only where necessary. A drawback of the fully automatic adaptive solution is that it also allocates a high resolution representation to areas with structured noise: The adaptively sampled version is able to capture the waving pattern on the popcorn tin. From the point of view of the algorithm, it is impossible to see that this is not actual motion going on but artifacts of the acquisition system, so it correctly adapts in such regions. In practice, this is not desirable but easy to fix by user input: In our system, we can "paint" the refinement levels directly on the urshape surface. By specifying no refinement for the rigid tin, that shows the acquisition artifacts, we can locally remove these noise patterns while retaining the automatically computed refined resolution in other parts of the object (see the video for a side-by-side comparison of the three versions; please note the differences on the popcorn tin and the person's shirt. Screenshots are shown in Figure 11).

Comparison to related work: In comparison to (Pekelny & Gotsman, 2008) and (Mitra *et al.*, 2007), our approach is much slower, but applicable to more general scenes, not being restricted to piecewise rigid or very slow motion. Our deformable reconstruction approach builds on the conceptual framework of (Wand *et al.*, 2007), which can solve the same problem with comparable generality. The improvement over the previous algorithm is in the complexity of scenes that can be handled (see Table 1 and Table 2): Our results work on sequences up to 200 frames and with up to about 20,000 surfels, improving upon (Wand *et al.*, 2007) by roughly an order of magnitude each. The hard limit for their technique is that the product of time and surfels could not exceed a limit of about 50-100,000K discretization points[3] without exceeding the 2GB limitation imposed by the runtime system. Our largest example (hand-4) encodes the positions of 2.47million surfels. In principle, our system can handle higher resolution urshapes and longer sequences. However, at this point the computation time of the algorithm becomes a limiting factor. The improvement in geometric fidelity in comparison to the previous results is examined in Figure 10: The reconstructed geometry is of significantly better quality than the previous results. Please note that some topological problems (such as in the eye regions in the face data set) are avoided with the new technique as this is resolved more clearly by the higher resolution geometry. The third improvement with our new technique is a better quality in motion reconstruction, which can be seen best in the accompanying video: Although we represent the motion in a sparse subspace,

[3]The exact limit depends various factors such as the neighborhood sizes in the topology graph.

we obtain results with less jittering and a higher effective spatial resolution of motion details.

Performance: Our method is in most cases faster than the previous technique; on a per-frame basis, our overall running times are $4-10\times$ faster (see Table 2). The only exception is the hand-2 data set, where we used both a particularly high resolution geometry and deformation to resolve the splitting fingers. This is a shortcomming of the rather simplistic topology detection scheme[4]. We have also measured the memory requirements of our new technique. Because we stream the high volume original data using a cached out-of-core approach, it is not reasonalbe to directly measure the overall process memory consumption; instead, we measure the working set assigned to the process by the operating system in order to estimate the active memory used during the computation. Table 3 shows the results. The working set estimates are rather noisy, but clearly show that the memory requirements depend on the resolution of the deformation model, but do not strongly depend on the sequence length, as desired. The amount of memory needed to solve for the deformation variables within one time window is currently the main limiting factor. This could be improved by employing a more memory efficient numerical algorithm than BFGS (which we prefer for its speed), such as non-linear conjugate gradients with an improved preconditioner.

8.2 Discussion & Limitations

Our method is able to reconstruct a template model from data sequences with acquisition holes and yields stable global correspondences over time. Our deformation model is able to depict the deformation more accurately than the previous model, although significantly fewer nodes are used to model the deformation. The technique has still a number of limitations: First, it relies on a locally convergent deformable ICP approach to determine frame-to-frame correspondences. For sequences with large temporal spacing, this does not converge. This problem also occurs if objects disappear in an acquisition hole and come out in a very different pose. Our result of the hand-3 sequence shows only part of the original data; in further frames, the hand closes fully, several fingers disappear from acquisition, and reappear in a different pose several frames later. In that case, our algorithm creates results with wrong alignments (hand with 6 fingers). Without the improved pipeline (using the urshape propagation), merging of the hand-3 data set was not possible to the shown extend. Another limitation is the computation time: Although the technique is not limited by memory demands, and it is significantly more efficient than the existing methods of comparable generality, the computational costs are still high. The main bottleneck is still in nearest neighbor queries necessary to setup ICP constraints and to construct topologically correctly connected deformation nodes. The costs

[4] In addition, some of the code used in this example did not use the same level of optimization as the other examples, which accounts for a additional runtime penalty.

for the latter are the main reason for not using too highly detailed urshapes. These problems could be addressed by more extensive precomputations, such as setting up volumetric signed distance fields for the data (Süssmuth *et al.*, 2008) and automatically employing a reduced resolution in setting up the deformation node topology. The last issue is choosing parameters: Our different energy terms have to be weighted by appropriate coefficients to balance their effect. The parameters need to be chosen carefully. We have been able to use one and the same base set of parameters for the merging step of the algorithm. For the hand-2 data set, the initial grid size was doubled (1/25 instead of 1/12 of the bounding box side length) in order to resolve the topological separation more accurately. For the popcorn tin data set, the culling threshold for surface points visible in only few data frames has been increased to reduce artifacts caused by structured outliers. Finally, reducing the settings for culling distant data points (usually 80% percentile, now 3%) in the hand-4 data set improves the alignment quality. For the final optimization step, we start from the same settings as in the sequence merging and manually determine acceleration penalty weights and typically also increase the rigidty slightly in order to get a more stable final result. Overall, our experience is that parameter setting is non-trivial, but we can find a set of standard parameters that give reasonable "one click" reconstruction results for most data sets. Fine tuning is required to obtain the best reconstruction quality.

9 Conclusions & Future Work

We have presented a novel system for reconstructing a single shape and its deformation from a temporal sequence of point clouds from real-time 3D scanner data. We handle geometry and deformation separately, using a novel adaptive subspace deformation technique, which leads to a more efficient algorithm and produces significantly better results than the previous approach of [Wand et al. 2007]: We obtain high resolution geometry and obtain visually better results for the deformation. The algorithm uses only a constant working set in memory for each optimization step that is solved, streaming through the data in small windows of a few frames size, so that arbitrary long sequences could be handled in principle. In future work, we would like to improve the optimization code using a parallel implementation on multi-core processors or possibly graphics hardware to further reduce the running time requirements, which currently discourage reconstructions of very large input sequences. While our system handles the local alignment problem robustly and in good quality, it is limited to situations where the inter-frame alignment can reliably and unambiguously recovered by a local deformable-ICP-like approach. We think that finding a strategy for globally assembling a full deformation sequence of partial scans robustly is a major open challenge in this line of work. For pairwise alignment, a number of robust global techniques have been proposed recently (Chang & Zwicker, 2008; Huang *et al.*, 2008; Li *et al.*, 2008), but for sequences, this problem is not yet

data set	data points (in all frames)	frames	nodes (merge/final opt.)	urshape surfels (merge/final opt.)	rec. time (merge/final opt.)
face[1]	23,880,623	79	315/315	20,977/20,920	304 min / 116 min
hand-1[2]	3,767,068	100	226/225	5,996/17,332	76 min / 44 min
hand-1 adapt.[2]	3,767,068	100	226/552	5,996/17,332	76 min / 80 min
hand-2[3]	4,106,467	34	932/414	23,357/23,357	326 min / 31 min
hand-3[1]	30,658,761	120	335/1,939	2,725/16,840	128 min / 445 min
hand-4[1]	45,325,272	201	403/424	4216/12,274	154 min / 147 min
popcorn tin[4]	5,179,918	98	429/423	6,428/6,428	175 min / 43 min
pop.t. adapt.[4]	5,179,918	98	429/1,151	6,428/6,428	175 min / 435 min
pop. t. user[4]	5,179,918	98	429/ 686	6,428/6,428	175 min / 224 min

Table 1: Example data sets: [1] from (König & Gumhold, 2007), [2] from (Weise *et al.*, 2007), [3] from aim@shape courtesy of O. Schall, [4] from (Fong & Buron, 2005).

	face	hand-2	popcorn tin
#frames	20	26	15
#surfels/frame	1637	819	3799
merging	328 min	67 min	327 min
final opt	122 min	23 min	22 min

Table 2: In comparison: running times of [Wand et al. 2007] (on a Pentium 4 3.4Ghz).

solved. An approach of automatically tracking geometric features over time with the ability to take topological changes into account could be a first step in addressing this problem.

Acknowledgments

The authors whish to thank Stefan Gumhold, Sören König, Phil Phong, Oliver Schall, and Thibaut Weise for providing the real-time 3D scanning data sets to us. This work was supported by the Max-Planck Center for Visual Computing and Communication (MPC-VCC), the Cluster of Excellence for Multi-Modal Computing and Interaction at Saarland University, NSF grants ITR 0205671 and FRG 0354543, NIH grant GM-072970, DARPA grant HR0011-05-1- 0007, DFG grant "Perceptual Graphics", Bart Adams has been funded as a post-doctoral researcher by the Fund for Scientific Research, Flanders (F.W.O.-Vlaanderen).

References

Adams, Bart, Ovsjanikov, Maks, Wand, Michael, Seidel, Hans-Peter, & Guibas, Leonidas J. 2008. Meshless Modeling of Deformable Shapes and their Mo-

data set	step	peak working set
face	merge	762 MB
face	global opt (uniform)	508 MB
hand-1	merge	452 MB
hand-1	final opt. (uniform)	602 MB
hand-4	merge	414 MB
hand-4	final opt. (uniform)	563 MB
popcorn	merge	1,3 GB
popcorn	final opt. (uniform)	700 MB
popcorn	final opt. (adaptiv)	1,6 GB

Table 3: Memory requirements during the optimization (working sets). The memory footprint depends mostly on the resolution of the deformation model, not on the length of the sequence.

Figure 9: The left 3 images correspond to the input data. The right 6 images to the reconstruction (sample of deformed urshapes). The texture in the leftmost images has been created by orthogonal projection onto the urshape and was then propagated to all other frames. The images are sorted from top to bottom according to Table 1 (*face*, *hand-1*, *hand-2*, *hand3*. *hand-4*, *popcorn tin*).

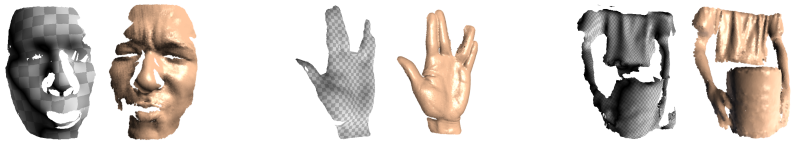

Figure 10: Comparison to [Wand et al. 2007] (left their reconstruction result / right ours): By decoupling shape and deformation, we obtain a significantly improved reconstruction quality. As shown in the side-by-side comparison above, the geometry is reproduced in more detail and with less topological noise artifacts in the solution, due to the higher resolution geometry. As shown in the accompanying video, the quality of the deformation fields is also improved.

Figure 11: Comparing three different reconstructions of the popcorn tin data set. Left: uniformly sampled deformations (grid size 1/12 bounding box side length). Middle: adaptive sampling (grid size $1/6 - 1/24$, computed fully automatically). Rightmost: same adaptive sampling, but with user augmented rigidity function.

tion. *In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation.*

Ahmed, Naveed, Theobalt, Christian, Roessl, Christian, Thrun, Sebastian, & Seidel, Hans-Peter. 2008. Dense Correspondence Finding for Parametrization-free Animation Reconstruction from Video. *In: Proc. Conf. Computer Vision and Pattern Recognition (CVPR).*

Allen, Brett, Curless, Brian, & Popović, Zoran. 2002. Articulated body deformation from range scan data. **21**(3), 612–619.

Allen, Brett, Curless, Brian, & Popović, Zoran. 2003. The space of human body shapes: reconstruction and parameterization from range scans. *Pages 587–594 of: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers.* New York, NY, USA: ACM.

Anguelov, D., Koller, D., Srinivasan, P., Thrun, S., Pang, H.-C., & Davis, J. 2005a. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. *In: Advances in Neural Information Processing Systems (NIPS 2004).*

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., & Davis, J. 2005b (August). SCAPE: Shape Completion and Animation of People. *In: Proceedings of the 32nd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH).*

Anuar, Nizam, & Guskov, Igor. 2004. Extracting Animated Meshes with Adaptive Motion Estimation. *Pages 63–71 of: VMV*.

Au, Oscar Kin-Chung, Tai, Chiew-Lan, Liu, Ligang, & Fu, Hongbo. 2006. Dual Laplacian Editing for Meshes. *IEEE Transactions on Visualization and Computer Graphics*, **12**(3), 386–395.

Avriel, Mordecai. 2003. *Nonlinear Programming: Analysis and Methods*. Dover Publishing.

Besl, Paul J., & McKay, Neil D. 1992. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, **14**(2), 239–256.

Botsch, Mario, & Sorkine, Olga. 2008. On Linear Variational Surface Deformation Methods. *IEEE Transactions on Visualization and Computer Graphics*, **14**(1), 213–230.

Botsch, Mario, Pauly, Mark, Gross, Markus, & Kobbelt, Leif. 2006. PriMo: coupled prisms for intuitive surface modeling. *Pages 11–20 of: SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Botsch, Mario, Pauly, Mark, Wicke, Martin, & Gross, Markus. 2007. Adaptive Space Deformations Based on Rigid Cells. *Computer Graphics Forum*, **26**(3), 339–347.

Carceroni, Rodrigo L., & Kutulakos, Kiriakos N. 2002. Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid 3D Motion, Shape and Reflectance. *Int. J. Comput. Vision*, **49**(2-3), 175–214.

Carranza, Joel, Theobalt, Christian, Magnor, Marcus A., & Seidel, Hans-Peter. 2003. Free-viewpoint video of human actors. *Pages 569–577 of: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*. New York, NY, USA: ACM.

Chang, Will, & Zwicker, Matthias. 2008. Automatic Registration for Articulated Shapes. *Computer Graphics Forum (Proceedings of SGP 2008)*, **27**(5).

Davis, James, Nehab, Diego, Ramamoorthi, Ravi, & Rusinkiewicz, Szymon. 2005. Spacetime Stereo: A Unifying Framework for Depth from Triangulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **27**(2), 296–302.

de Aguiar, Edilson, Stoll, Carsten, Theobalt, Christian, Ahmed, Naveed, Seidel, Hans-Peter, & Thrun, Sebastian. 2008. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, **27**(3), 1–10.

Fong, P., & Buron, F. 2005. High-resolution three-dimensional sensing of fast deforming objects. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, 1606–1611.

Fries, Thomas-Peter, & Matthies, Hermann G. 2003. *Classification and*

Overview of Meshfree Methods. Tech. rept. TU Brunswick, Germany Nr. 2003-03.

Huang, Jin, Shi, Xiaohan, Liu, Xinguo, Zhou, Kun, Wei, Li-Yi, Teng, Shang-Hua, Bao, Hujun, Guo, Baining, & Shum, Heung-Yeung. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph.*, **25**(3), 1126–1134.

Huang, Qi-Xing, Adams, Bart, & Wand, Michael. 2007. Bayesian surface reconstruction via iterative scan alignment to an optimized prototype. *Pages 213–223 of: SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Huang, Qi-Xing, Adams, Bart, Wicke, Martin, & Guibas, Leonidas J. 2008. Non-Rigid Registration Under Isometric Deformations. *Computer Graphics Forum (Proc. SGP 2008)*, **27**(5).

König, Sören, & Gumhold, Stefan. 2007. Image-based Motion Compensation for Structured Light Scanning of Dynamic Scenes. *In: EG Workshop on Dynamic 3D Imaging (Dyn3D '07)*.

Li, Hao, Sumner, Robert W., & Pauly, Mark. 2008. Global Correspondence Optimization for Non-Rigid Registration of Depth Scans. *Computer Graphics Forum (Proc. SGP 2008)*, **27**(5).

Medioni, G., Lee, M.S., & Tang, C.K. 2000. *A Computational Framework for Segmentation and Grouping*. Elsevier.

MESA. http://www.mesa-imaging.ch.

Mitra, Niloy J., Flöry, Simon, Ovsjanikov, Maks, Gelfand, Natasha, Guibas, Leonidas, & Pottmann, Helmut. 2007. Dynamic geometry registration. *Pages 173–182 of: SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., & Alexa, M. 2004. Point based animation of elastic, plastic and melting objects. *Pages 141–151 of: SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Nealen, Andrew, MÃ$\frac{1}{4}$ller, Matthias, Keiser, Richard, Boxerman, Eddy, & Carlson, Mark. 2005. Physically Based Deformable Models in Computer Graphics.

Park, Sang Il, & Hodgins, Jessica K. 2006. Capturing and animating skin deformation in human motion. *ACM Trans. Graph.*, **25**(3), 881–889.

Pauly, Mark, Keiser, Richard, Adams, Bart, Dutré, Philip, Gross, Markus, & Guibas, Leonidas J. 2005. Meshless animation of fracturing solids. *ACM Trans. Graph.*, **24**(3), 957–964.

Pekelny, Y., & Gotsman, C. 2008 (4). Articulated Object Reconstruction and Markerless Motion Capture from Depth Video. vol. 27.

PMD. http://www.pmdtec.com.

Sand, Peter, McMillan, Leonard, & Popović, Jovan. 2003. Continuous capture of skin deformation. *ACM Trans. Graph.*, **22**(3), 578–586.

Sharf, Andrei, Alcantara, Dan A., Lewiner, Thomas, Greif, Chen, Sheffer, Alla, Amenta, Nina, & Cohen-Or, Daniel. 2008. Space-time Surface Reconstruction using Incompressible Flow. *ACM Trans. Graph. (Proc. Siggraph Asia)*.

Sheffer, Alla, & Kraevoy, Vladislav. 2004. Pyramid Coordinates for Morphing and Deformation. *Pages 68–75 of: 3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*. Washington, DC, USA: IEEE Computer Society.

Shewchuk, Jonathan Richard. 1994. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rept. Carnegie Mellon University, School of Computer Science.

Shi, Lin, Yu, Yizhou, Bell, Nathan, & Feng, Wei-Wen. 2006. A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph.*, **25**(3), 1108–1117.

Shi, Xiaohan, Zhou, Kun, Tong, Yiying, Desbrun, Mathieu, Bao, Hujun, & Guo, Baining. 2007. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *Page 81 of: SIGGRAPH '07: ACM SIGGRAPH 2007 papers*. New York, NY, USA: ACM.

Starck, Jonathan, & Hilton, Adrian. 2005. Spherical Matching for Temporal Correspondence of Non-Rigid Surfaces. *Pages 1387–1394 of: ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society.

Sumner, Robert W., Zwicker, Matthias, Gotsman, Craig, & Popović, Jovan. 2005. Mesh-based inverse kinematics. *ACM Trans. Graph.*, **24**(3), 488–495.

Sumner, Robert W., Schmid, Johannes, & Pauly, Mark. 2007. Embedded deformation for shape manipulation. *Page 80 of: SIGGRAPH '07: ACM SIGGRAPH 2007 papers*. New York, NY, USA: ACM.

Süssmuth, Jochen, Winter, Marco, & Greiner, Günther. 2008. Reconstructing Animated Meshes from Time-Varying Point Clouds. *Computer Graphics Forum (Proceedings of SGP 2008)*, **27**(5), 1469–1476.

Terzopoulos, Demetri, Platt, John, Barr, Alan, & Fleischer, Kurt. 1987. Elastically deformable models. *Pages 205–214 of: SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM.

Varanasi, Kiran, Zaharescu, Andrei, Boyer, Edmond, & Horaud, Radu P. 2008. Temporal Surface Tracking Using Mesh Evolution. *Pages 30–43 of: Proceedings of the Tenth European Conference on Computer Vision*. LNCS, vol. Part II. Marseille, France: Springer-Verlag.

Wand, Michael, Jenke, Philipp, Huang, Qixing, Bokeloh, Martin, Guibas, Leonidas, & Schilling, Andreas. 2007. Reconstruction of deforming geometry from time-varying point clouds. *Pages 49–58 of: SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Weise, T., Leibe, B., & Gool, L. Van. 2007 (June). Fast 3D Scanning with Automatic Motion Compensation. *In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*.

Würmlin, Stephan, Lamboray, Edouard, Staadt, Oliver G., & Gross, Markus H. 2002. 3D Video Recorder. *Page 325 of: PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*. Washington, DC, USA: IEEE Computer Society.

Xu, Weiwei, Zhou, Kun, Yu, Yizhou, Tan, Qifeng, Peng, Qunsheng, & Guo, Baining. 2007. Gradient domain editing of deforming mesh sequences. *Page 84 of: SIGGRAPH '07: ACM SIGGRAPH 2007 papers*. New York, NY, USA: ACM.

Zhang, Li, Curless, Brian, & Seitz, Steven M. 2003 (June). Spacetime Stereo: Shape Recovery for Dynamic Scenes. *Pages 367–374 of: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Zhang, Li, Snavely, Noah, Curless, Brian, & Seitz, Steven M. 2004 (August). Spacetime Faces: High-Resolution Capture for Modeling and Animation. *Pages 548–558 of: ACM Annual Conference on Computer Graphics*.

Zhou, Kun, Huang, Jin, Snyder, John, Liu, Xinguo, Bao, Hujun, Guo, Baining, & Shum, Heung-Yeung. 2005. Large mesh deformation using the volumetric graph Laplacian. *ACM Trans. Graph.*, **24**(3), 496–503.

Zitnick, C. Lawrence, Kang, Sing Bing, Uyttendaele, Matthew, Winder, Simon, & Szeliski, Richard. 2004. High-quality video view interpolation using a layered representation. *Pages 600–608 of: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*. New York, NY, USA: ACM.

Zwicker, Matthias, Pfister, Hanspeter, van Baar, Jeroen, & Gross, Markus. 2001. Surface splatting. *Pages 371–378 of: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM.