Eurographics Symposium on Geometry Processing 2009
Marc Alexa and Michael Kazhdan
(Guest Editors)

Volume 28 (*2009*), *Number 5*

Constructing Deformable Shapes from Range Scans

paper1063

Abstract
We present a method to reconstruct a deformable shape from a set of range scans. The input to our algorithm is a sequence of range scans where the object takes a different pose in each scan. Then, we solve for the skinning weights and transformation matrices of the linear blend skinning (LBS) model that best align all input scans to a common reference pose. To handle multiple frames effectively, we avoid an accumulation of the alignment error by optimizing the alignment of all frames simultaneously. In addition, we develop a method to maintain a single set of skinning weights rather than having to maintain the weights separately in each frame. Our method does not rely on markers, an initial segmentation, or template information to construct the model. Also, unlike previous work that focus just on registration, the advantage of our method is that we can use the skinning weights to animate and re-pose the reconstructed object.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—

1. Introduction

Capturing deformable geometry is an important problem in computer graphics. A system that enables fast and accurate capture of moving shapes has many potential applications in games, movie production, and virtual reality. This topic has recently received considerable attention specifically for applications in performance capture [BPS*08, dAST*08, VBMP08].

We can now use high-speed depth cameras to capture range scans of dynamic objects at video frame rates [WLVG07]. Unlike static range scans, dynamic range scans contain both the information about the surface and how the underlying object moves. This provides an exciting opportunity to construct both the surface of the object and a model of its motion. However, a key processing step with range data is registration: the process of aligning frames to reconstruct the complete surface. This problem has been solved for static objects [BM92, BR07], but it remains challenging for moving objects that can take different poses in each scan.

In this paper, we present an algorithm to reconstruct both a surface and a motion model from a set of range scans. To solve for the object surface, we must compensate for the motion of the object to align all the scans to a common "reference" pose. We accomplish this by prescribing a motion model for the shape and solving for the model parameters to best predict the observed range scan data. This reconstructed model can then be directly used for re-posing and animating the shape.

For describing the movement of the object, we use a simple motion model called linear blend skinning (LBS). In this model, the surface is divided into "soft regions," where each region is associated with a transformation matrix. These regions are indicated by assigning a vector of "skinning weights" at each point on the surface, where the ith component of the vector correspondences to the weight or influence of the ith transformation. Using this model allows easy editing of the movement because a user only needs to specify a few transformations to move the entire shape. Our goal is to automatically solve for these parameters—the transformations and weights of LBS—to align a set of range scans to a common pose. Our method does not rely on markers, an initial segmentation, or template information to construct the model. Also, unlike previous work that focus just on registration, the advantage of our method is that it gives both the completed shape and associated skinning weights.

The main contribution of this paper is to optimize these quantities for multiple scans. The basic idea is to solve for the alignment of all frames to a common "reference" frame. A challenge here is that scans can align on top of each other, causing the surface to gradually "thicken." Also, small errors in the alignment can gradually grow, leading to a "drift" of the object's surface. We formulate the optimization so that it solves for the global alignment of multiple scans simultaneously. This is effective for reducing these types of errors.

Also, we must define a weight for all points (in all frames) to solve for the transformations. This is because the transformations and weights depend on each other. We describe a way to maintain a single, consistent set of weights across all input frames.

Overcoming these challenges allows us to accurately reconstruct complete shape models from range scans of moving geometry. Specifically, our contributions are:

- An algorithm to reconstruct a deformable shape from partial 3D data, without markers or a template, using a simultaneous alignment of all input data,
- An algorithm for global optimization of weights and transformations on multiple frames,
- A technique to maintain a single, consistent weight function among multiple input frames.

2. Related Work

Shape registration is an active and well-studied area in computer graphics and vision. There has been considerable interest recently in registering multiple scans of deformable shapes. A class of methods model the moving object as a surface in 4D space-time [MFO*07,SWG08,SAL*08]. In addition, Wand and colleagues [WJH*07,WAO*09] fit a graph of surface point trajectories to the input data. The main difference in our work is that we learn both the shape and skinning weights, allowing a user to animate the reconstructed object using forward or inverse kinematics.

A closely related work is by Pekelny and Gotsman [PG08], who reconstruct both the surface and articulated skeleton from depth scans with a user-given segmentation. In our work, we do not assume that a segmentation is given, but instead we learn this automatically based on the motion of the input data. Chang and Zwicker [CZ09] present a method to automatically fit the weights and transformations of LBS to align a pair of range scans. We build on this method and extend it handle multiple range scans, by formulating the optimization of weights and transformations over multiple frames.

We also draw on the work of rigid registration [Neu97] and extend it to the articulated case. However, we extend these approaches to handle multiple transformations in each frame and also to incorporate joints between the transformations in each frame.

Our work is also related to methods that focus on learning models of surface motion from geometric data. In particular, there is much work in capturing, representing, and manipulating human body shapes [ACP02,ACP03,ASK*05, ACPH06, HSS*09]. These works rely on markers and/or templates to build the initial database of body shapes from range scans, but are able to learn more expressive surface motion models than our approach. It is also possible to construct rigged models from mesh animations [JT05, SY07, dATTS08], but these techniques rely on input triangle meshes that have the same connectivity and topology. In contrast, our method attempts to construct rigged models using partial geometric data that is not in correspondence, without using markers or a template.

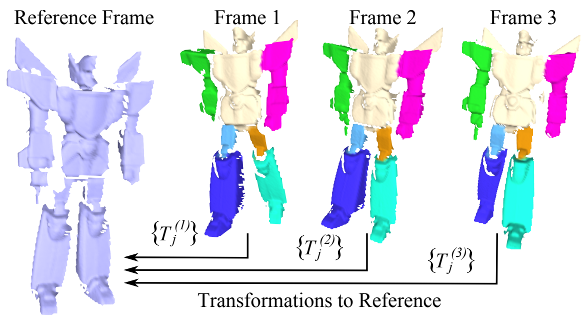

Figure 1: *For each input frame, we solve for a set of transformations that align the frame to the reference.*

3. Algorithm Overview

Our goal is to reconstruct the complete surface and associated skinning weights of an object that fits the observed range scan data. Concretely, we must find the transformation matrices and their associated skinning weights (regions) that aligns all input frames to a reference frame, as shown in Figure 1.

For this purpose, we extend a pairwise registration method [CZ09] to handle multiple scans. If this method is applied directly to multiple scans, it causes a "drifting" of the surface (if frames are registered sequentially), or it needs to keep a separate set of skinning weights for each registered pair (if frames are registered hierarchically in groups of two frames).

We address these limitations and reformulate the optimization to simultaneously align multiple input frames, while maintaining a single set of skinning weights for the entire dataset. To improve the performance of the optimization, we optimize the alignment of a reduced set of sampled locations on each frame. These sample points serve as locations to measure the alignment distance of each frame to the reference. We summarize our entire algorithm in the following procedure and in Figure 2.

1. **Initialize Reference:** Sample points and initialize weights on the reference frame (Section 4.1).
2. **For each new frame:**
   a. **Initialize Frame:** Initialize weights on the new frame, by aligning the reference frame to the new frame (Section 4.2).
   b. **Global Alignment:** Optimize the global alignment of all initialized frames (Section 4.3).
   c. **Update Reference:** Update the reference with the surface information of all frames (Section 4.4).
3. **Post-process:** Refine weights, accumulate all scanned points, and reconstruct the surface geometry (Section 4.4).

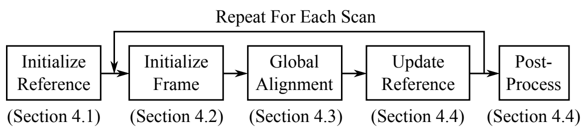

Figure 2: *Overview of our reconstruction algorithm.*

In the subsequent sections we discuss each step of our algorithm in more detail. But first, we provide some technical background for our discussion.

3.1. Background

The linear blend skinning (LBS) model describes the motion of a surface using weights and transformations. Each vertex of the surface is assigned a vector of weights (one weight per transformation) in some reference pose. Then, the transformations move each vertex according to its assigned weights by computing a weighted sum of the transformed points. This is summarized by the function $D(\mathbf{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, where

$$D(\mathbf{x}) = \sum_{j=1}^{B} w_j(\mathbf{x}) T_j(\mathbf{x}) = \sum_{j=1}^{B} w_j(\mathbf{x}) \left(R_j \mathbf{x} + \mathbf{t}_j \right). \quad (1)$$

Here, B is the total number of transformations, and j is the index of the transformation. $T_j(\mathbf{x})$ denotes applying the transformation (rotation $R_j \in SO(3)$ and translation $\mathbf{t}_j \in \mathbb{R}^3$) to the position $\mathbf{x}$. Also, let us denote $\mathbf{w}(\mathbf{x})$ as the vector of weights at $\mathbf{x}$, with components $w_j(\mathbf{x})$ for each transformation j. We assume that the weights are non-negative and sum to 1 for all $\mathbf{x} \in \mathbb{R}^3$.

For our algorithm, we will use a single weight function defined in the reference frame. However, since the pose of each frame is different, we will have a different set of transformations for each frame. We use the notation $T_j^{(f \rightarrow \text{Ref})}$ to denote the jth transformation for frame f. Note that the direction of transformation $T_j^{(f \rightarrow \text{Ref})}$ is from frame f to the reference frame.

The range scan data is given as a set of point clouds. We also estimate the normals at each point by using a simple mesh constructed from the rectilinear structure of the scanned points. We can also use a more robust alternative, such as estimating the least-squares fitting plane at each point.

4. Registration Algorithm

In this section, we describe in detail our optimization for the global alignment of all frames. We first describe how to initialize each input frame, followed by details of the global optimization.

4.1. Initializing the Reference Frame

In our algorithm, the reference frame serves two important purposes: as a common location to (1) align all the input frames, and (2) define the skinning weights. Because we need to solve for the weights, we define them on a regu-

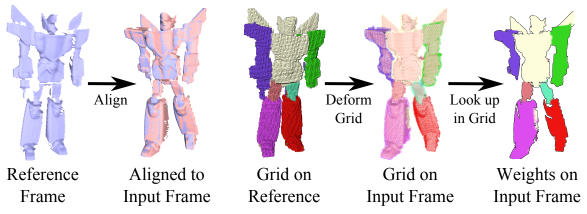

Figure 3: *To look up the weights on any input frame, we first align the reference to the frame and deform the grid accordingly. We can then directly look up the weights in the deformed grid.*

lar volumetric grid enclosing the reference frame geometry [CZ09]. Thus, in this step we start the registration by (1) sampling points on the reference to measure the alignment of future frames, and (2) creating the grid and initializing the weights on this grid.

We pick samples by subsampling the input points using best-candidate sampling [Mit91]. We initialize the weights by assigning binary weights (labels) on each grid point. This is done by first picking B random locations on the reference frame (where B is the total number of transformations) and assigning each grid point the label of the closest location. This provides a rough segmentation of the input geometry into multiple parts, and will serve as a starting point for refining the weights further in the optimization.

4.2. Initializing a New Frame

For each new frame, our goal is to align this new frame to the reference. Although we could find the transformations from the new frame to the reference frame, we also need weights defined on this new frame to specify where each transformation applies. However, we only define the weights on the grid in the reference frame. We need some way to determine the weights in the new frame using the grid of the reference frame.

This is the purpose of the initialization step. The idea is to first perform an alignment of the *reference frame* to the *new frame*. This is done by solving for the transformations that minimize the alignment distance of the reference frame to the new frame. Then we deform the grid itself according to these transformations, which overlays the grid with the points in the new frame. Now we can directly look up the weights in the new frame by finding the grid cell containing each point and interpolating the weights of the cell's vertices. This procedure defines weights in the new frame, and now we can include the new frame in the global registration.

Suppose that we have processed up to frame l. To perform the alignment to the new frame $l+1$, we take the transformations to the last frame l as a starting point to optimize the transformations to the new frame $l+1$. The optimization itself is the same as the global alignment (Section 4.3), except that we solve for transformations $T^{(l \rightarrow l+1)}$ from frame l to frame $l+1$, while the weights are still optimized for all

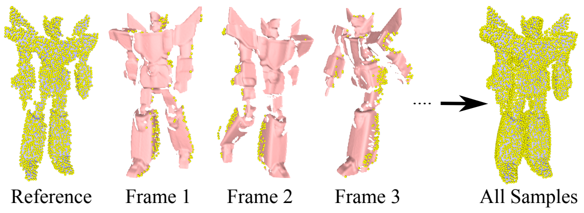

Figure 4: *We use sample points on all input frames to measure the global alignment. For each frame, we only keep the samples that are from new geometry that has not been observed in any previous frames.*

frames.

When we apply the transformations to each grid point (according to its weights), we essentially have a set of deformed grid cells overlapping the new frame (see Figure 3). To look up the weight for a point $\mathbf{p}$, we (1) find the grid cell containing p and (2) solve for its trilinear coordinates within the cell using Newton's method.

Recall that we measure the alignment using sample points (Section 3). To take into account the geometry of all processed frames, we gather all sampled locations (of all processed frames) onto the reference frame. This sample set S is constructed to represent well-spaced locations throughout the entire shape.

To increase performance, we remove overlapping sample locations, as illustrated in Figure 4. We first initialize S to be the samples on the reference frame. For the next frame, we transform the samples of f to the reference frame and remove points whose proximity to S is smaller than a threshold. Then we add the remaining points to S. We repeat this process for all remaining frames. For proximity, we use Euclidean distance to the closest point in S, projected *onto* the the tangent plane of the closest point [PG08]. For the threshold we typically use 2–5 times the range scan sample spacing.

Sometimes it is the case that a new scan is not sufficiently close to the last scan. In this case, the alignment may converge to an undesired local minimum. This is a failure mode for the algorithm, and the user is given an option to retry the alignment specifying different parameters. Also, often entire parts of the object are occluded in the range scan. We detect these parts automatically and remove their associated transformations and weights from the subsequent optimization [PG08].

4.3. Global Registration

Once a frame is initialized, it is introduced into the global registration step. This step optimizes for the best weights (denoted $\mathcal{W}$) and transformations (denoted $\mathcal{T}$) that simultaneously align all initialized frames to the reference frame. The optimization objective has three terms: (1) $\mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W})$, which measures the alignment distance of all frames to the reference, (2) $\mathcal{E}_{\text{joint}}(\mathcal{T})$, which constrains neighboring transformations to agree on a common joint location, and (3) $\mathcal{E}_{\text{weight}}(\mathcal{W})$, which constrains the weights to be smooth and to form contiguous regions. With weights α, β, γ for each term, we write the entire objective as

$$\underset{\mathcal{T},\mathcal{W}}{\operatorname{argmin}}\ \alpha\,\mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W}) + \beta\,\mathcal{E}_{\text{joint}}(\mathcal{T}) + \gamma\,\mathcal{E}_{\text{weight}}(\mathcal{W}). \qquad (2)$$

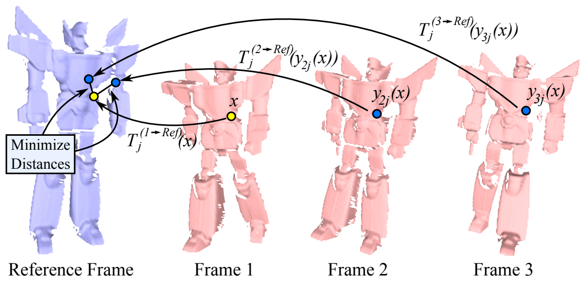

Figure 5: *To measure alignment, we compute distances between sample points* $\mathbf{x}$ *and target points* $\mathbf{y}_{kj}$ *mapped to the reference frame. Then, we optimize for the transformations and weights minimizing the total distance.*

Next, we describe each term in more detail and give details of how we solve the optimization. During the optimization, solving the weights in a continuous range leads to overfitting [CZ09]. To resolve this problem, we constrain the weights to be binary, where only one component can be 1 and the rest are 0.

Fitting Objective $\mathcal{E}_{\text{fit}}$. The key idea for this term is to measure the alignment distance between all frames using the sample points. For each sample point $\mathbf{x}$ on frame f, we keep a list of target positions $\mathbf{y}_{kj}(\mathbf{x})$ for each frame k and each transformation j. Then, we use the transformations and weights to map all positions to the reference frame. A good fit will transform $\mathbf{x}$ and $\mathbf{y}_{kj}(\mathbf{x})$ to the same location (Figure 5). We write this in the equation

$$\mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W}) = \sum_{\mathbf{x}\in S}\ \sum_{\text{Frames } k} \qquad (3)$$
$$\sum_{j=1}^{B} w_j(\mathbf{x})\, d\left(T_j^{(f\to\text{Ref})}(\mathbf{x}), T_j^{(k\to\text{Ref})}\left(\mathbf{y}_{kj}(\mathbf{x})\right)\right).$$

For each j, $T_j^{(f\to\text{Ref})}(\mathbf{x})$ transforms $\mathbf{x}$ to the reference frame, and $T_j^{(k\to\text{Ref})}\left(\mathbf{y}_{kj}\right)$ transforms $\mathbf{y}_{kj}(x)$ to the reference frame. Then, $d(\cdot,\cdot)$ measures the distance between the points, and the weight w_j "selects" one of the distances according to its value. We can think of this as selecting because the weights are binary. This is repeated for all frames k and all sample points $\mathbf{x}$ to compute the total alignment distance.

For $d(\cdot,\cdot)$ we use a weighted sum of the point-to-point and point-to-plane distance measures:

$$d(\mathbf{x},\mathbf{y}) = \eta_{\text{pt}}\ \|\mathbf{x}-\mathbf{y}\|^2 + \eta_{\text{pl}}\ \left((\mathbf{x}-\mathbf{y})\cdot\vec{\mathbf{n}}_{\mathbf{y}}\right)^2. \qquad (4)$$

For the point-to-plane distance, we also need the normal vector $\vec{\mathbf{n}}_{\mathbf{y}}$ of $\mathbf{y}$. This vector is transformed to the reference frame

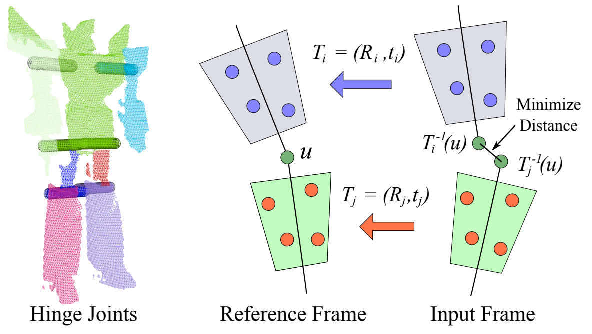

Figure 6: *On the left, we show hinge joints that are automatically estimated. These joints are constrained in $\mathcal{E}_{joint}$ as shown on the right. This term constrains the transformed locations of* **u** *to agree on the same point by minimizing the distance between the transformed locations.*

as well. We typically set the weights to be $\eta_{\text{pt}} = 0.2$ and $\eta_{\text{pl}} = 0.8$ for our experiments.

Joint Objective $\mathcal{E}_{\text{joint}}$. The joint term constrains neighboring transformations to agree on a common joint location in order to avoid degenerate configurations. We define joint locations on the reference frame and constrain the transformations to map these locations to the same point in all frames (see Figure 6, right side).

Our method supports detecting and constraining two types of joints: 3 DOF ball joints and 1 DOF hinge joints. Detection is performed via solving a minimization problem [AKP*04, PG08]. For hinge joints, the solution will be a set of points (on the reference frame) lying on the hinge axis. Solving with SVD, we can detect hinges by examining if the ratio of the smallest singular value to the sum of the singular values is less than a threshold. If this is the case, then we truncate the smallest singular value to zero and solve for the equation of the line satisfying the system. If this is not the case, we treat this joint as a ball joint and solve for its location, which is a single point **u**. To our knowledge, the detection of hinge joints in this manner has not been done before.

Once we have the joint locations and types, we constrain all joints to agree on the joint location using the term $\mathcal{E}_{\text{joint}}$:

$$\mathcal{E}_{\text{joint}}(\mathcal{T}) = \sum_{\text{Frames } f} \sum_{\text{Joints } (i,j)} \sum_{\mathbf{u} \in \text{Joint}} \left\| T_i^{(f\to\text{Ref})^{-1}}(\mathbf{u}) - T_j^{(f\to\text{Ref})^{-1}}(\mathbf{u}) \right\|^2. \tag{5}$$

In the case of a ball joint, there is only one point **u** where transformations i and j are constrained, and for the case of a hinge joint, and generate a set of points **u** along the hinge axis and invidually constrain each point [KVD05].

Weight Objective $\mathcal{E}_{\text{weight}}$. The binary weights transform the problem into a discrete labelling problem, where we try to find an optimal assignment of transformations (interpreted as "labels") to the grid cells. Thus, for $\mathcal{E}_{\text{weight}}$ we just use a constant penalty when two neighboring cells (c,d) have different weights. This is a simple form of the Potts model, which is a discontinuity-preserving interaction penalty [BVZ01].

4.3.1. Optimization

To solve the optimization, we divide the solver into two phases and alternate between each phase until the solution converges. In the first phase, we keep the weights fixed and solve for the transformations, and in the second phase, we keep the transformations fixed and solve for the weights. This strategy works well in practice and produces a good alignment within a few iterations [CZ09].

Before each iteration, we update the target positions $\mathbf{y}_{kj}(\mathbf{x})$ for each frame k and transformation j. These target positions are the corresponding locations used in $\mathcal{E}_{\text{fit}}$ for each sample point $\mathbf{x}$. If we have ground-truth correspondences, then we can have the same position $\mathbf{y}_{kj}(\mathbf{x})$ for all j. But in the absence of correspondences, we transform each sample point $\mathbf{x}$ from frame f to k for each j (using $T_j^{(k\to\text{Ref})} T_j^{(f\to\text{Ref})^{-1}}$) and take the closest point on frame k.

Some frames may not have a corresponding point due to missing data. To handle this case, for each sample point $\mathbf{x}$ (transformed to frame k), we invalidate the corresponding target point $\mathbf{y}_{kj}$ if (1) the distance between these points exceeds a threshold τ_d, (2) the angle between the normals exceeds a threshold τ_n, or (3) the target point lies on the boundary and the distance exceeds a smaller threshold τ_b [PG08]. In particular, we mark all $\mathbf{y}_{kj}(\mathbf{x})$ as invalid for all j if the target position for the current positive weight is invalid.

For optimizing the first phase, we need to (1) find the weights at each sample point $\mathbf{x}$ and (2) solve for the transformations minimizing the terms $\alpha\, \mathcal{E}_{\text{fit}}(\mathcal{T}, \mathcal{W}) + \beta\, \mathcal{E}_{\text{joint}}(\mathcal{T})$ from Equation 2. Recall that, in order to look up the weights in each frame, we must first use the current transformations to map the weight grid to each frame (Section 4.2). Because we look up the weights on deformed grid cells (whose location depends on the transformations used to map them to each frame), the weight values depend on the current transformations. Therefore, after updating the transformations, we transform the grid again and update the weights on each frame.

Numerically, we solve the resulting non-linear least squares problem using the standard Gauss-Newton algorithm. We found that optimizing the transformations simultaneously for many frames can be slow. To improve the performance of our implementation, we provide an option to solve for the transformations at a sliding window of frames. Typically we have used a window of 5–10 frames in our experiments.

For the second phase, we constrain the weights to be binary (see Section 4.3) and solve the resulting discrete optimization of $\alpha\, \mathcal{E}_{\text{fit}} + \gamma\, \mathcal{E}_{\text{weight}}$ using graph cuts [BVZ01, BK04, KZ04]. Here we provide a value of $\mathcal{E}_{\text{fit}}$ for each individual grid cell c and each transformation j by summing the distance over all sample points $\mathbf{x}$ that is contained within c.

Furthermore, we can precompute this value for all cells and all transformations to perform the optimization very quickly.

4.4. Updating Reference and Post-Processing

After the global alignment, we extend the weight grid to enclose the points of the newly introduced frame. To update the extent of the grid, we transform all scans to the reference frame and construct a new grid that contains all the points. If the limits of the grid expand, we take care to perform bookkeeping for precomputed quantities such as the weights of the points in each frame (Section 4.2) and $\mathcal{E}_{\text{fit}}$ (Section 4.3.1).

Finally, after finishing the registration of the entire sequence, we transform all scanned points to the reference frame to build a complete point set of the shape. Here, we remove redundant points using the same procedure outlined in Section 4.2 for the sample points. We then reconstruct a single surface from these points (and their associated normals) using the streaming wavelet surface reconstruction algorithm [MPS08].

Also, since we use discrete weights during the optimization, we also provide the option of solving for continuous weights as a post-processing step [CZ09].

5. Experimental Results

We implemented our algorithm in C++ and tested it with several real-world and synthetic datasets exhibiting articulated motion. All of our results were produced on an Intel 2.4 GHz Core 2 Duo using one core.

The car and robot datasets were acquired by Pekelny and Gostman [PG08]. These datasets had a total 90 frames each. The reconstruction results, shown in Figure 7 and Figure 9, demonstrate that we can obtain visually accurate results without a segmentation given by the user. Our method also gives good results for a more deformable subject. We generated 60 synthetic depth scans of a running man, where the camera rotating around the man and the pose changes in each frame. Even with the wide range of motion and occlusion in these frames, we can reconstruct the entire surface, shown in Figure 10. We also acquired scans of a bendable, poseable pink panther toy which was moved to a different pose in each scan. After aligning 16 frames, we were able to reconstruct the surface and skinning weights, even though the furry texture on the surface created a lot of noise in the range scan (Figure 11). For these last two examples, we applied the weight smoothing step as a post-process to reduce artifacts due to binary weights. Here, we applied a simple uniform gaussian blur of fixed radius (2–5 grid cells) on the weights. Also, in each dataset, the user often had to manually restart the alignment with different parameters, because the initial alignment converged to the wrong solution. For example, this happened for about 21 out of 90 frames for the car dataset, and 23 out of 90 frames for the robot dataset. The pink panther dataset required a restart for most frames, because it moved too much between the frames.

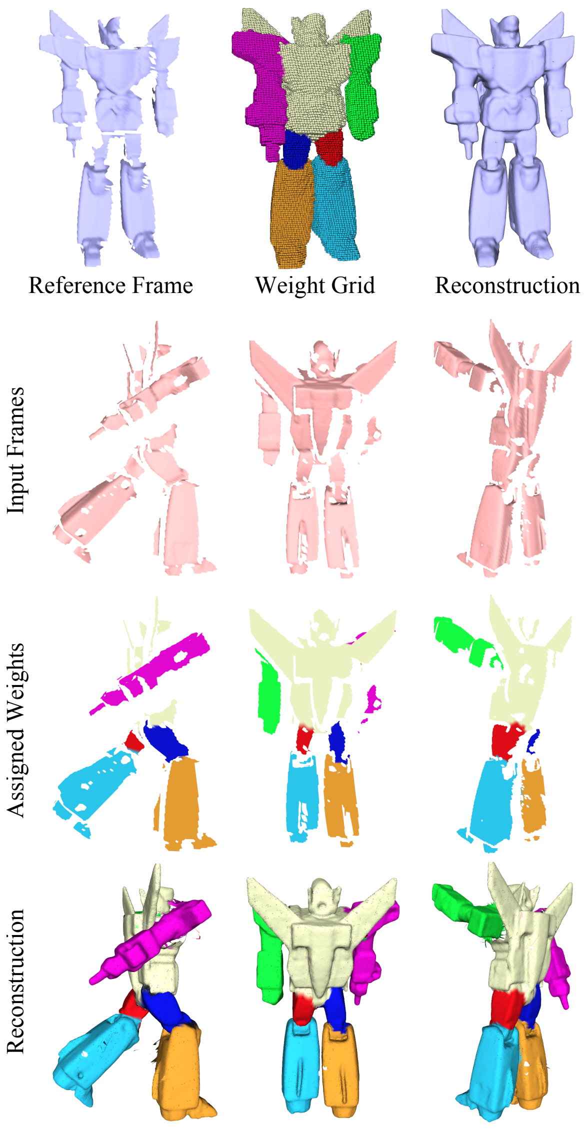

Figure 7: *Reconstruction results for the robot dataset.*

The performance of our implementation is reported in Table 1. The computational cost of our method was moderate, requiring about 4–6 minutes to align each frame. This includes the time for initializing each frame (including manual restarts) and simultaneous registration. Our algorithm has several parameters affecting the behavior of the optimization. The main parameters are the the number of transformations B, the resolution of the grid, weights $\alpha = 1, \beta = 1, \gamma = s$ adjusting the relative influence of each term in the optimization (Section 4.3), and the thresholds $\tau_d = 15s, \tau_n = 45^\circ, \tau_b = s$ that determine if a target point is valid (Section 4.3.1). Several of these parameters are expressed as a multiple of the range scan grid spacing s. The parameters that influenced the optimization most were the weight of the joint term β (varied between 0 and 1), the angle between

	Robot	Car	RunMan	PP
Bones	7	5	15	17
Grid Cells	32327	30630	23638	14824
Frames	90	90	60	16
Points	845208	484907	819464	314286
Samples	4595	2785	5080	9584
Setup	10.03	6.57	10.14	6.88
Init Frame	87.23	56.31	117.04	149.51
Global	94.01	129.71	137.95	75.88
Update	39.96	46.51	33.93	10.60
Total (sec)	231.24	239.10	299.06	242.87

Table 1: *Performance statistics for our experiments. Timings (in seconds) represent the average time spent per frame in each stage.*

normals τ_n (varied between 15° and 45°), and the distance threshold τ_d (varied between $10s$ and $40s$).

It is very useful to reconstruct the skinning weights associated with each object based on the range scans. To demonstrate this, we implemented a tool to perform inverse kinematics on the reconstructed shape model. In this system, the user specifies point constraints on the model, and we use our optimization of the transformations (Section 4.3.1) to satisfy these constraints. The result was an interactive tool for the user to intuitively re-pose and animate the reconstructed subject. Figure 8 shows examples of different poses of the robot created by our system. Please see the accompanying video for a recorded live session of the IK system in use.

6. Limitations and Future Work

We would like to investigate ways of reducing the parameters and automating the system. Our system is not robust to large motion in the input frames, requiring manual intervention to restart the alignment with different parameters. The registration could be automated further by incorporating techniques to track large motions in range scans. Also, other parameters could be estimated automatically. For example, we could automatically estimate the number of transformations by starting with a single transformation and splitting regions to lower the total error.

The grid used to define the weights is not aware of the surface topology. For example, the grid had trouble separating the spinning platform on the car model (the small blue rotating part in Figure 9), or between the two legs of the running man model (middle column, last row in Figure 10). Using a higher resolution for the grid improved the results, but did not resolve the situation when different parts come in contact. We believe one promising avenue for future work is to replace the grid with a more flexible structure. For example, we can define weights on the sample points directly and construct a graph between nearby sample points. Care must be taken to properly interpolate these weights on all scanned points while respecting the object's topology.

Also, it would be interesting to adapt this method for completely non-rigid examples. There should be a middle ground

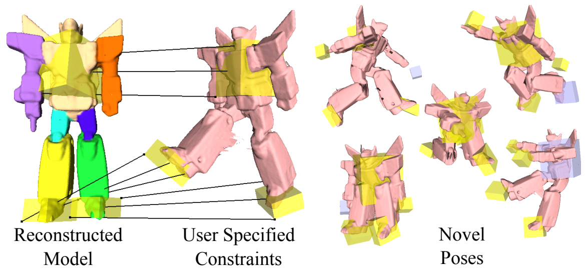

Figure 8: *Reposing the reconstructed robot. By using the solved weights and the hinge joints, our optimization can satisfy point constraints given by the user.*

between specifying a separate transformation on every sample point [SSP07] and solving for the weights at each sample point as we do.

7. Conclusion

We have presented a method to reconstruct a deformable shape from a set of range scans. From a set of dynamic range scans, we solve for the skinning weights and transformation matrices of the linear blend skinning (LBS) model that best align all input scans to a common reference pose. For this purpose, we formulated a simultaneous optimization for all input frames to minimze registration error. The advantage of our method is that it does not rely on markers, an initial segmentation, or template information to construct the model. Finally, we have demonstrated that the reconstructed skinning weights are useful for animating and re-posing the reconstructed object.

References

[ACP02] Allen B., Curless B., Popović Z.: Articulated body deformation from range scan data. *ACM SIGGRAPH 21*, 3 (2002), 612–619.

[ACP03] Allen B., Curless B., Popović Z.: The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH* (2003).

[ACPH06] Allen B., Curless B., Popović Z., Hertzmann A.: Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In *SCA* (2006).

[AKP*04] Anguelov D., Koller D., Pang H., Srinivasan P., Thrun S.: Recovering articulated object models from 3d range data, 2004.

[ASK*05] Anguelov D., Srinivasan P., Koller D., Thrun S., Rodgers J., Davis J.: Scape: shape completion and animation of people. In *SIGGRAPH* (2005).

[BK04] Boykov Y., Kolmogorov V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 26*, 9 (September 2004), 1124–1137.

[BM92] Besl P. J., McKay H.: A method for registration of 3-d shapes. *Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (1992), 239–256.

[BPS*08] Bradley D., Popa T., Sheffer A., Heidrich W., Boubekeur T.: Markerless garment capture. *ACM Transactions on Graphics 27* (2008).

[BR07] BROWN B. J., RUSINKIEWICZ S.: Global non-rigid alignment of 3-d scans. In *SIGGRAPH* (New York, NY, USA, 2007), ACM, p. 21.

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *PAMI 23*, 11 (2001), 1222–1239.

[CZ09] CHANG W., ZWICKER M.: Range scan registration using reduced deformable models. *Computer Graphics Forum (Eurographics)* (2009).

[dAST*08] DE AGUIAR E., STOLL C., THEOBALT C., AHMED N., SEIDEL H.-P., THRUN S.: Performance capture from sparse multi-view video. *SIGGRAPH* (2008).

[dATTS08] DE AGUIAR E., THEOBALT C., THRUN S., SEIDEL H.-P.: Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum 27* (2008).

[HSS*09] HASLER N., STOLL C., SUNKEL M., ROSENHAHN B., SEIDEL H.-P.: A statistical model of human pose and body shape. *Computer Graphics Forum 28* (2009).

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. In *SIGGRAPH* (2005).

[KVD05] KNOOP S., VACEK S., DILLMANN R.: Modeling joint constraints for an articulated 3d human body model with artificial correspondences in icp. In *IEEE-RAS International Conference on Humanoid Robots* (2005).

[KZ04] KOLMOGOROV V., ZABIH R.: What energy functions can be minimized via graph cuts? *PAMI 26*, 2 (2004), 147–159.

[MFO*07] MITRA N. J., FLORY S., OVSJANIKOV M., GELFAND N., GUIBAS L. J., POTTMANN H.: Dynamic geometry registration. In *SGP* (2007).

[Mit91] MITCHELL D. P.: Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH* (1991).

[MPS08] MANSON J., PETROVA G., SCHAEFER S.: Streaming surface reconstruction using wavelets. In *SGP* (2008).

[Neu97] NEUGEBAUER P. J.: Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images. *International Journal of Shape Modeling 3*, 1/2 (1997), 71–90.

[PG08] PEKELNY Y., GOTSMAN C.: Articulated object reconstruction and markerless motion capture from depth video. *EUROGRAPHICS 27*, 2 (2008).

[SAL*08] SHARF A., ALCANTARA D. A., LEWINER T., GREIF C., SHEFFER A., AMENTA N., COHEN-OR D.: Space-time surface reconstruction using incompressible flow. *SIGGRAPH ASIA* (2008).

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM Press, p. 80.

[SWG08] SÜSSMUTH J., WINTER M., GREINER G.: Reconstructing animated meshes from time-varying point clouds. *Computer Graphics Forum 27* (2008).

[SY07] SCHAEFER S., YUKSEL C.: Example-based skeleton extraction. In *SGP* (2007).

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. *SIGGRAPH* (2008).

[WAO*09] WAND M., ADAMS B., OVSJANIKOV M., BERNER A., BOKELOH M., JENKE P., GUIBAS L., SEIDEL H.-P., SCHILLING A.: Efficient reconstruction of non-rigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics 28* (2009).

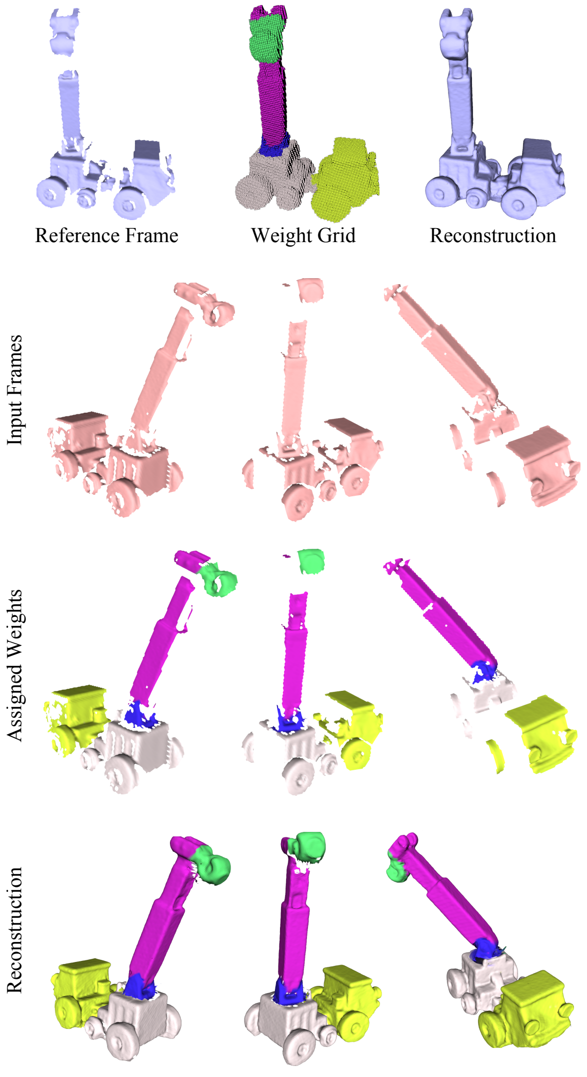

Figure 9: *Reconstruction results for the car dataset.*

[WJH*07] WAND M., JENKE P., HUANG Q., BOKELOH M., GUIBAS L. J., SCHILLING A.: Reconstruction of deforming geometry from time-varying point clouds. In *SGP* (2007).

[WLVG07] WEISE T., LEIBE B., VAN GOOL L.: Fast 3d scanning with automatic motion compensation. In *CVPR* (2007).

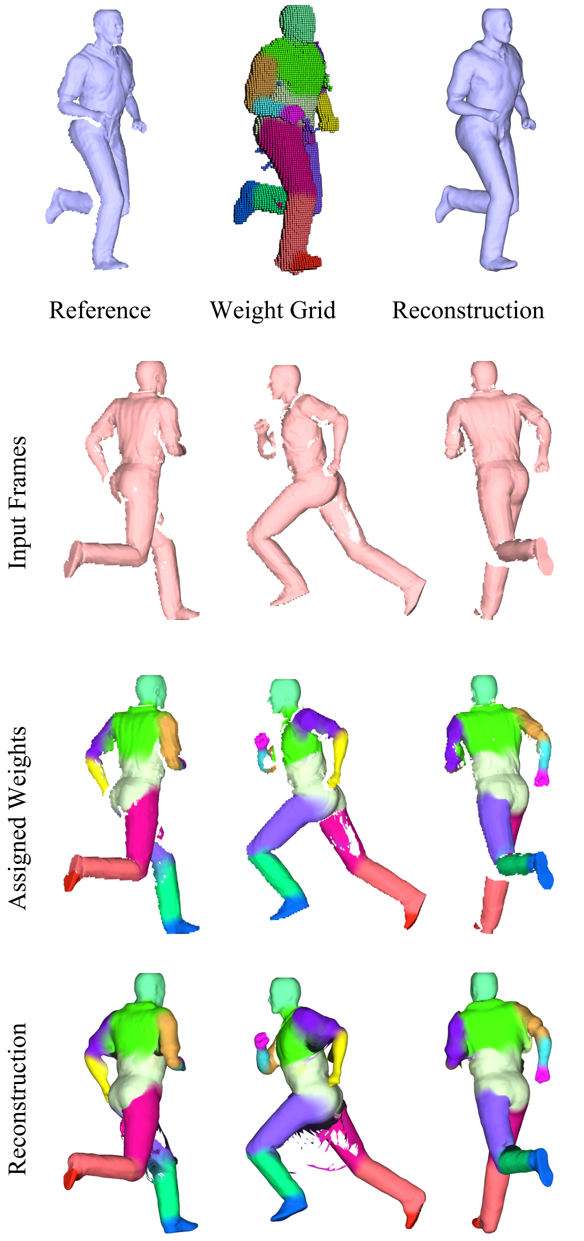

Figure 10: *Reconstruction results for the Running Man dataset.*

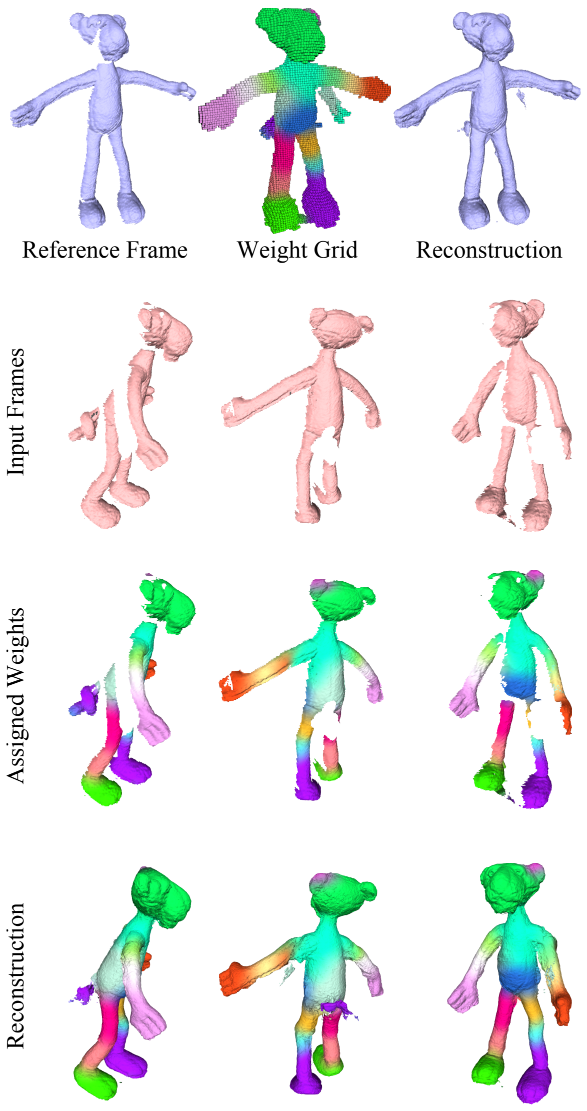

Figure 11: *Reconstruction results for the Pink Panther dataset.*