Global Registration of Dynamic Range Scans for Articulated Model Reconstruction

WILL CHANG
University of California, San Diego
and
MATTHIAS ZWICKER
University of Bern

We present a method to reconstruct articulated 3D models from dynamic, moving range scan sequences. The main contribution is a novel global registration algorithm that aligns all scans to a common pose, and reconstructs a full 3D model from the geometry of these scans. Unlike other registration algorithms, we express the surface motion in terms of a reduced, articulated deformable model and solve for joints and skinning weights. This allows a user to interactively manipulate the reconstructed 3D model in order to create new poses and animations.

We express the global registration as an optimization of simultaneously estimating the alignment and articulated structure for all scans. Compared to a sequential registration approach, the global registration estimates the correct articulated structure that is based on the motion observed in all frames, resulting in a more accurate registration. In addition, we employ a graph-based representation for the weight function, which is successful in handling difficult topological cases well. We show that we can automatically reconstruct a variety of 3D models, without the use of markers, user-placed correspondences, a segmentation, or a template. In addition, our algorithm also supports reconstructing reasonable piecewise rigid approximations to non-rigid motion sequences.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric Algorithms*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Surface Fitting*

General Terms: Algorithms, Measurement

Additional Key Words and Phrases: Range scanning, articulated model, non-rigid registration, animation reconstruction

ACM Reference Format:

Authors' addresses: land and/or email addresses.

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

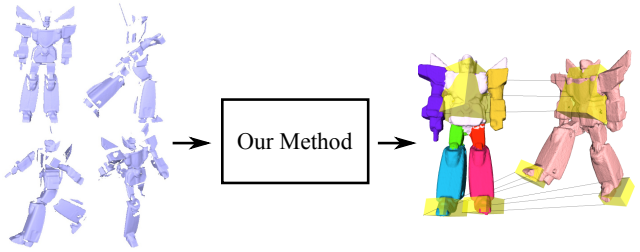


Fig. 1. Our method can automatically reconstruct articulated, poseable models from a sequence of single-view dynamic range scans.

1. INTRODUCTION

While 3D scanning has traditionally been focused on acquiring static, rigid objects in the past, recent advances in real-time 3D scanning has opened up the possibility of capturing dynamic, moving subjects. Range scanning has become both practical and cost-effective, providing high-resolution, per-pixel depth images at high frame rates. However, despite the many advances in acquisition, many challenges still remain in the processing of dynamic range scans to reconstruct complete, animated 3D models.

The vision is to automatically reconstruct detailed, poseable models that animators can directly plug into existing software tools and create new animations. The main challenge is to resolve the occlusion and missing data that occur in range scans, due to a limited view of a 3D subject from any single viewpoint. Scans taken from many different viewpoints must be aligned and integrated together in order to reconstruct a complete surface. When the subject moves in each frame, we must also track the spatially varying surface motion accurately to obtain a good alignment. Since the tracking is not performed by the scanner, it must be estimated in the processing step by directly matching the surface geometry. An additional challenge is to improve the usability of the reconstructed model by expressing the surface motion in terms of a small set of parameters. This can allow animators to easily create new animations and performances of the subject.

We present an algorithm to address these challenges by reconstructing a rigged, articulated 3D model from dynamic range scans. Given a sequence of range scans of a moving subject, our algorithm automatically aligns all scans to produce a complete 3D model. This is accomplished without the assistance of markers, user-placed correspondences, a template, or a segmentation of the surface. Our method is unique because we perform the alignment by estimating the parameters of a reduced, articulated deformation model. In contrast to methods that focus only on registration or reconstruction of

the original recording, our method produces a 3D model that can be interactively manipulated with no further post-processing.

Our algorithm automatically estimates the articulated model using an alternating optimization approach inspired by the pairwise registration method of Chang and Zwicker [2009]. Our contributions are:

—A global registration algorithm that optimizes the registration simultaneously over all frames,

—A novel registration formulation that produces a 3D model with skinning weights learned from incomplete examples,

—An improved robust registration technique to automate the global registration with coarse alignments of adjacent frames.

We demonstrate the effectiveness of our algorithm by reconstructing several synthetic and real-world datasets. We also present a simple extension of our algorithm to interactively manipulate the resulting 3D model.

2. RELATED WORK

Template-Based Reconstruction. A popular approach to reconstruct deforming sequences of range scans is to fit a template to the observed scan data. A template provides many advantages in tracking and fitting the data, with the expense of requiring the user to scan or model it in advance. Our work addresses the more general problem of reconstructing the template automatically from the range scans.

Many techniques rely on tracked marker locations to automatically fit a template model to the scanned point cloud data [Allen et al. 2002; 2003; Anguelov et al. 2005; Pauly et al. 2005]. For the specific case of deforming garments, the method by Bradley et al. [2008] automatically tracks a few key locations to fit the template. The pairwise registration by Anguelov et al. [2004] does not require markers and is robust to the initial pose of the scan, but it requires a template and uses a global optimization that is expensive to compute. Markerless shape capture is also possible when the range scan sequence has a high frame rate. For example, it is possible to capture human faces by fitting a template face model to a structured-light range scan video sequence [Zhang et al. 2004; Weise et al. 2009]. The resulting face animation can be used to create new animations or track novel sequences in real-time, but again the template must be known in advance. Li et al. [2009] automatically reconstruct a non-rigid range scan video sequence and reproduce the fine surface detail observed in the range scans. However, this also requires a coarse template of the subject to be scanned prior to the tracking step. Although our work is focused on articulated subjects, the articulated assumption allows us to handle larger temporal spacing between scans, therefore producing a complete, rigged model without using a template.

Templates are also used for estimating shape using multiview silhouette/video data [de Aguiar et al. 2008; Vlasic et al. 2008; Gall et al. 2009] or sparse marker data [Park and Hodgins 2006; 2008]. Although these methods address the same problem of capturing deformable geometry, they do not address how to process high-resolution range scan data taken from just one or two views. Also, while the surface detail in our work comes directly from the range scans, most of the surface detail in these methods come directly from the template, or it is added as a post-process using dense normal maps computed by shape from shading [Ahmed et al. 2008].

Template-less Reconstruction. To tackle the reconstruction problem without a template, many researchers have considered modeling a dynamic range scan sequence as a surface in four-dimensional space and time, rather than a single 3D surface that changes its configuration over time. Mitra et al. [2007] use kinematic properties of this 4D space time surface to track points and register multiple frames of a rigid object. Süßmuth et al. [2008] and Sharf et al. [2008] explicitly model and reconstruct the 4D space-time surface using an implicit surface representation. However, these techniques require the surface to be sampled densely in both space and time, which is an assumption that our method does not require. In addition, the latter method does not track points to produce correspondence between frames, and it is more appropriate for filling in missing surface data not observed by the scanner.

A closely related work is the statistical optimization approach by Wand et al. [2009]. This method aligns multiple frames by solving the surface motion in terms of an adaptive displacement field. This motion representation handles smooth deformations well, but our representation is more compact and accurate for representing articulated motion. Wand et al. [2009] align and merge pairs of adjacent frames in a hierarchical fashion, gradually building the template shape hierarchically as well. In contrast, we simultaneously align all frames at once using an explicit piecewise rigid deformation model. In addition, our method is more robust to large movements and produces a fully rigged, poseable 3D model, rather than just reconstructing the original recorded motion sequence.

Our method is partly inspired by the articulated motion capture and reconstruction method of Pekelny and Gotsman [2008]. However, this method requires the user to manually segment a range scan in advance, whereas we automatically solve for the segmentation using the motion observed in all frames.

Unsupervised Pairwise Registration. While our method is designed for aligning multiple range scans, several methods for aligning a pair of scans are related to our work as well. A closely related work is the method by Chang and Zwicker [2009], which solves for the alignment between a pair of range scans by estimating the parameters of a reduced deformable model. A possibility is to apply this method directly for multiple scans, using a sequential pairwise registration and accumulation approach. However, in this case the correct articulated structure is not estimated properly, because it considers the movement in only two frames at a time (see Section 8.5). Also, unless a very high resolution is used, the grid-based representation of the weights cannot handle difficult topological cases with close or nearby surfaces. As we will demonstrate in the results section, we overcome these limitations to handle multiple frames and difficult topological cases effectively.

The transformation sampling and optimization approach by Chang and Zwicker [2008] is used in our work to initialize the registration between pairs of adjacent frames. However, this technique is too slow to apply for an entire sequence of range scans. We improve the performance of this method by subsampling the geometry. Our use of a graph to represent the deformation model is related to the approach by Li et al. [2008] and [Sumner et al. 2007]. However, we solve for weights on the graph nodes, as opposed to solving for a separate affine transformation at each node. The method by Huang et al. [2008] also uses a graph, but they use it as an approximation of geodesic distances in order to extract a set of geodesically consistent correspondences. However, this approach is problematic when a large amount of surface data is missing.

Deformation Modeling from Examples. Our inverse kinematics system resembles that of FaceIK [Zhang et al. 2004] or MeshIK [Sumner et al. 2005], which extrapolate a set of examples to match user constraints. However, the deformation model that we produce is a parametric model that explicitly models parts and joints, as opposed to a data-driven method that blends a set of

Algorithm 1: ARTICULATED GLOBAL REGISTRATION

Data: A sequence of range scans, denoted $(F_0, \ldots, F_{n-1})$
Result: Sample set S of the completed surface, weights $\mathcal{W}$ for each sample $\mathbf{x} \in S$, rigid transformations $\mathcal{T}$ for each part for each frame

1 **begin**
2 Compute the initial registration between each pair of adjacent frames (Section 4);
3 Subsample initial sample set S from F_0, and construct a Euclidean k-nearest neighbor graph on S;
4 $F_{\text{last}} \leftarrow F_0$;
5 **while** $F_{last} \neq F_{n-1}$ **do**
6 Let F_{new} be the next frame after F_{last};
7 Load and apply the initial registration result for $F_{\text{last}} \rightarrow F_{\text{new}}$ (Section 4);
8 Handle missing parts in F_{new} (Section 7);
9 Optimize $\mathcal{T}, \mathcal{W}$ $(S, E, \mathcal{T}, \mathcal{W}, F_0, \ldots, F_{\text{new}})$ (Algorithm 2);
10 Resample S from all frames $(F_0, \ldots, F_{\text{new}})$ (Section 6);
11 $F_{\text{last}} \leftarrow F_{\text{new}}$;
12 **return** $S, \mathcal{W}, \mathcal{T}$;
13 **end**

example meshes. Therefore, our interactive IK system does not use the original examples at run-time and only uses the reconstructed deformation parameters (skinning weights and joints) to pose the 3D model.

Our deformation modeling approach is closer to the example-based skeleton extraction work [Anguelov et al. 2004; Schaefer and Yuksel 2007; de Aguiar et al. 2008]. However, while these approaches estimate the deformation parameters using a set of complete examples that are already in correspondence, we estimate them directly from incomplete range scan data.

3. ALGORITHM OVERVIEW

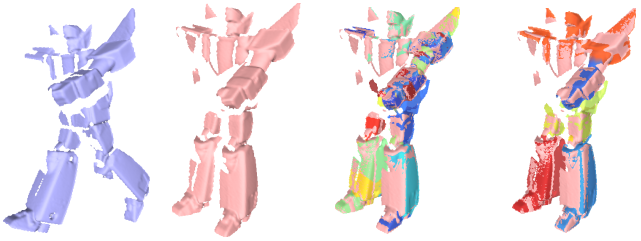
The input to our algorithm is a sequence of n range scans, where the subject is moving in each scan. We denote this sequence as $F_0, \ldots, F_{n-1}$. We also expect this to be in temporal order, so that there is sufficient overlap between frames to align the scans.

The goal of our algorithm is to align all scans to a common pose and express the surface motion using a reduced set of parameters. We pose this problem as a skinning problem: finding transformations per frame and weights per vertex. When we apply these transformations to each scan according to the weights, all scans should be aligned with each other.

The basic structure of our method is shown in Algorithm 1. The first step is to solve for a coarse initial registration for each pair of adjacent frames $(F_0, F_1), (F_1, F_2), \ldots, (F_{n-2}, F_{n-1})$ (line 2). We use the transformation sampling and optimization approach by Chang and Zwicker [2008]. This method is used because it can align a pair of scans while being robust to missing data and large motions. We also improve the speed of this method so that it is suitable for aligning multiple frames.

The second step is to refine this initial registration and produce a global registration of all frames (lines 3–11). The main idea is to optimize the transformations and weights simultaneously across all frames to align them to a common reference pose. The frames are introduced sequentially, one at a time, into the global registration (lines 5–11). For each frame, we load the initial registration (line 7), handle occluded parts (line 8), optimize the transformations $\mathcal{T}$ and weights $\mathcal{W}$ to simultaneously align the frames (line 9), and update the sample set S that is used for the optimization (line 10). After finishing the registration for the entire sequence, we resample the surface densely and reconstruct a mesh of the completed surface.

During the global registration, some parts may entirely disappear (and reappear) in several frames. To handle these cases, we check if there are too few matching samples for each part. If this is the case, then the part is marked as occluded and is subsequently excluded from the optimization. Also, when a part reappears, perhaps in a different location, we have a strategy to track the part again during the global registration.



(a) Source (blue) and Target (red) 21325 points total
(b) Alignment Using All Points: 1330.4 sec
(c) Alignment Using 1000 Points: 87.3 sec

Fig. 2. Comparison showing performance improvement for the coarse initial registration. With the same parameters, optimizing on a subset of the points produces a similar registration in a fraction of the time.

4. COARSE INITIAL REGISTRATION

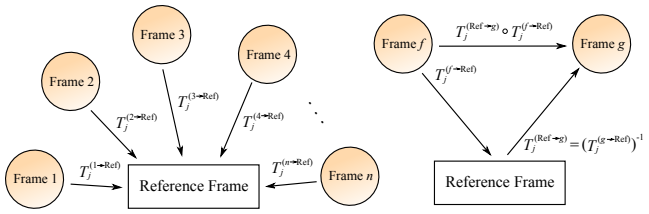
The first step is to solve for a coarse initial registration for each pair of adjacent frames. Since the scans have missing data and their poses can be far apart, the algorithm of Chang and Zwicker [2008] is well suited for producing a robust registration. It consists of two steps: (1) sampling rigid transformations from feature-based correspondences between the scans, and (2) optimizing the assignment of these transformations onto each vertex of the scans, so that it produces the best alignment.

However, with range scans that typically have thousands of points, this method is too slow to process an entire range scan sequence with many frames. To improve performance, we perform the optimization on a small subset of the points in the scans, instead of optimizing on all points. This makes sense for articulated movement, because the number of unique transformations producing the movement is small compared to the number of scanned points.

The details of the method are the same as originally described by Chang and Zwicker [2008]. However, the optimization is restricted to the small subset of points (e.g. 500-1000) uniformly sampled from the scans. To substitute for the edges of the triangle mesh that were used for specifying smoothness constraints, we use a k-nearest neighbor graph constructed on the subset of points, where k is typically 15. After the optimization, we propagate the transformations assigned to the subset to all remaining points using nearest-neighbor interpolation. This finally produces the coarse registration used as an initialization for the global registration.

5. GLOBAL REGISTRATION

The global registration step optimizes for the best transformations and weights that simultaneously align all initialized frames. Before discussing the details of the algorithm, we first describe our system



(a) Configuration of Transformations (b) Composition Between Frames

Fig. 3. Organizing the transformations for simultaneous registration. (a) We solve for the set of transformations that align each input frame to the reference frame F_0. (b) We can transform between any pair of frames f and g by first transforming from f to the reference and applying the inverse transformation to g.

of representing transformations and the weights consistently across multiple scanned frames.

Representing Transformations. The transformations express the movement of the surface in each frame. We designate one of the frames as a *reference frame* and define the transformations relative to this reference[1]. We use the notation $T_j^{(f\to\text{Ref})}$ to denote the jth transformation for frame F_f, which transforms in the direction *from* frame f *to* the reference frame (Figure 3a).

Each $T_j^{(f\to\text{Ref})}$ consists of a rotation matrix $R \in SO(3)$ and translation vector $\vec{\mathbf{t}} \in \mathbb{R}^3$. To apply this transformation to a point $\mathbf{x} \in \mathbb{R}^3$, we notate $T_j^{(f\to\text{Ref})}(\mathbf{x}) = R\mathbf{x} + \vec{\mathbf{t}}$. We also express the relative transformation $T_j^{(f\to g)}$ between any two frames f to g by transforming to the reference and then transforming to the desired frame (Figure 3b). We express this as

$$\begin{aligned} T_j^{(f\to g)}(\mathbf{x}) &= \left(T_j^{(g\to\text{Ref})^{-1}} \circ T_j^{(f\to\text{Ref})}\right)(\mathbf{x}) \\ &= R_j^{(g\to\text{Ref})^\top}\left[\left(R_j^{(f\to\text{Ref})}\mathbf{x} + \vec{\mathbf{t}}_j^{(f\to\text{Ref})}\right) - \vec{\mathbf{t}}_j^{(g\to\text{Ref})}\right]. \quad (1) \end{aligned}$$

Therefore, once we know the transformations on each frame, we can transform between any two frames. This definition makes it easy to specify and solve for the alignment for multiple frames.

Representing Weights. Although the transformations are applied to each point, we do not solve for a separate transformation for every point. This would result in too many transformations to solve in the optimization. Instead, we associate the transformations to the points indirectly by assigning weights to each point. Each weight is a vector $\mathbf{w}_\mathbf{x}$, where the j^{th} component indicates the influence of transformation j to the point $\mathbf{x}$. This is analogous to "skinning" a model.

By changing the weights during the optimization, we can dynamically adjust where each transformation is being applied. Having this level of indirection makes sense for an articulated subject, where a relatively small number of transformations can express the movement of the surface.

We apply several more improvements to this basic idea. Our first observation is to solve for a weight on a small subset of the points, instead of solving for a separate weight on every point. This reduces the number of variables in the optimization and makes it more efficient. We call this subset the *sample set* S. Each member of S, which we call a *sample point*, is a scanned point $\mathbf{x} \in \mathbb{R}^3$ selected from an input frame F_f. (In the subsequent text, we will implicitly assume that the sample point $\mathbf{x}$ is associated with the frame F_f.) After we have determined the weights for S, we extrapolate weights to the rest of the points using a nearest-neighbor-like interpolation.

Our second observervation is that when the frames are registered, many samples from different frames will overlap. Since there is no need to define the same weight multiple times on the same location, we resample S to remove overlapping locations. This further improves the efficiency in the optimization.

Third, we solve for binary weights, where one component is exactly 1 and the rest are 0. This is because solving for smooth weights during registration leads to overfitting of both transformations and weights [Chang and Zwicker 2009]. Therefore, all components of $\mathbf{w}_\mathbf{x}$ are 0 and only one component $w_{\mathbf{x}j^*} = 1$. Here, we use j^* to indicate the index of the component with 1.

Finally, in addition to the sample set S, we construct a k-nearest neighbor graph on S, which we call the *all-samples graph (ASG)*. The connectivity of this graph serves as smoothness constraints that help the optimization form large, contiguous parts.

5.1 Optimization Objective

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 form contiguous regions. With weights α, β, γ for each term, we write the entire objective as

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

Fitting Objective $\mathcal{E}_{\text{fit}}$. In this term, we measure the alignment distance between all frames using the sample points. For each sample point $\mathbf{x} \in S$, we transform it all other frames and measure how close it is to the scanned data of these frames. Here, we use the distance to the closest correponding point in the other frame. For example, for a point $\mathbf{x}$ in frame f, we measure its alignment distance to frame g by transforming it to frame F_g (using $T_{j^*}^{(f\to g)}(\mathbf{x})$) and finding the closest corresponding point $\mathbf{y}_{j^*}^{(g)} \in F_g$. Here, j^* is the index of the transformation assigned to $\mathbf{x}$, i.e. the component of $\mathbf{w}_\mathbf{x}$ with 1.

Once we have the corresponding point, we compute the total alignment distance using the formula

$$\mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W}) = \sum_{\mathbf{x}\in S} \sum_{\text{Valid } \mathbf{y}_{j^*}^{(g)}} d\left(T_{j^*}^{(f\to\text{Ref})}(\mathbf{x}), T_{j^*}^{(g\to\text{Ref})}\left(\mathbf{y}_{j^*}^{(g)}\right)\right). \quad (3)$$

Here we have computed the distance $d(\cdot,\cdot)$ between $\mathbf{x}$ (in F_f) and its corresponding point $\mathbf{y}_{j^*}^{(g)}$, where both points have been transformed to the reference frame (see Figure 4). The resulting values are summed up over all sample positions $\mathbf{x}$ and all frames g 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:

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

where $\vec{\mathbf{n}}_\mathbf{y}$ is the surface normal of $\mathbf{y}$, which is transformed to the reference frame as well. We use the weights $\eta_{\text{pt}} = 0.2$ and $\eta_{\text{pl}} = 0.8$ for our experiments.

We add some more important details for computing the closest corresponding point.

[1]This is similar to the approach used by Neugebauer [1997] for registering scans of rigid objects.

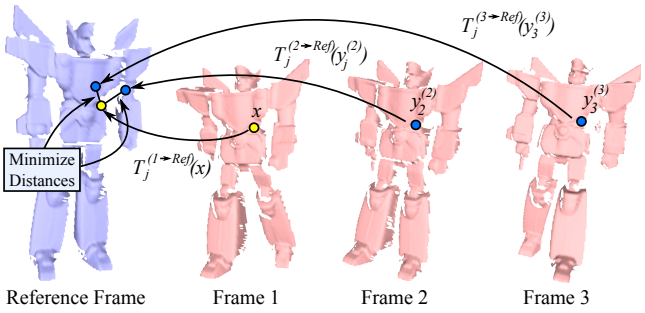


Fig. 4. To measure alignment, we compute distances between sample points $\mathbf{x}$ (yellow) and corresponding points $\mathbf{y}_j^{(g)}$ (blue) transformed to the reference frame F_{ref}. We add up these distances to measure the alignment of all frames in the sequence. We optimize for the transformations and weights that minimize this total distance.

(1) It may be the case that $\mathbf{x}$ may not have a corresponding point in F_g due to missing data. We use well-known heuristics to detect these cases, and we mark the corresponding point $\mathbf{y}_j^{(g)}$ as invalid. The three heuristics we use are [Pekelny and Gotsman 2008]:

—The distance between these points exceeds a threshold τ_d,

—The angle between the normals exceeds a threshold τ_n,

—The corresponding point lies on the boundary, and the distance exceeds a smaller threshold τ_b.

(2) Notice that the closest point will change depending on which transformation we use to transform $\mathbf{x}$ to frame g. Therefore, we maintain a separate closest point $\mathbf{y}_j^{(g)}$ for each j. These are used to evaluate the alignment distance per transformation when we optimize the weights. However, if the closest point for component j^* is invalid, then most likely there is no corresponding point in the frame. In this case we invalidate all corresponding $\mathbf{y}_j^{(g)}$ for all j.

(3) Finally, we do not find a closest corresponding point when $g \leq f$; i.e. we only match closest points forward in the sequence and not backwards. Thus, we do not match corresponding points for all n^2 pairs of frames, but roughly $n^2/2$ pairs instead.

Joint Objective $\mathcal{E}_{\text{joint}}$. The joint term constrains neighboring transformations to agree on a common joint location. It ensures that the parts stay connected to each other and do not drift apart. We support automatically detecting and constraining two types of joints: 3 DOF ball joints and 1 DOF hinge joints.

We define the joint locations in the reference frame F_{ref}. A hinge joint specifies that two transformations are connected along a line in $\mathbb{R}^3$, which means that both transformations transform this line to exactly the same location. We call this line the *hinge axis*, which can be described using the parametric form $\mathbf{u} + t\vec{\mathbf{v}}$, where $t \in \mathbb{R}$. A ball joint says that the transformations connect on a single point $\mathbf{u} \in \mathbb{R}^3$. We express a ball joint in the same form as the hinge, except that $\vec{\mathbf{v}} = \vec{\mathbf{0}}$. An example of hinge joints detected for the robot model is illustrated in Figure 5 (left).

Once we know these joint locations and types, we can constrain the transformations to map the joint locations to the same place (Figure 5, middle & right). Let us represent a joint between transformations for label i and j using the tuple $(\mathbf{u}_{ij}, \vec{\mathbf{v}}_{ij})$. We additionally set a valid/invalid flag for each tuple, depending on whether there actually is a joint between transformations i and j. Now, we

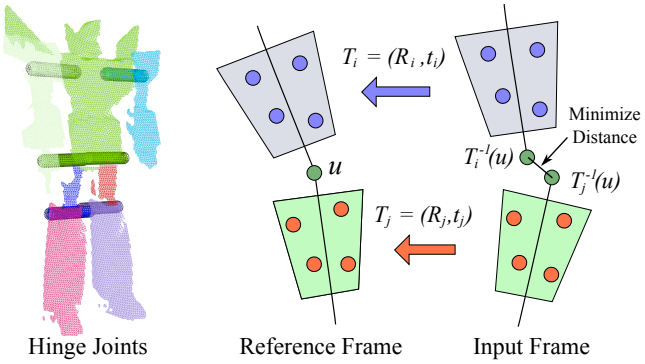


Fig. 5. Estimating and constraining joints in our optimization. (Left) we show hinge joints that are automatically estimated. (Middle & Right) $\mathcal{E}_{\text{joint}}$ constrains the transformed locations of $\mathbf{u}$ to agree on the same point by minimizing the distance between the transformed locations.

can constrain the joints using the term $\mathcal{E}_{\text{joint}}$:

$$\mathcal{E}_{\text{joint}}(\mathcal{T}) = \sum_{\text{All } F_f} \sum_{\substack{\text{Valid Joints} \\ (\mathbf{u}_{ij}, \vec{\mathbf{v}}_{ij})}} \sum_{t \in \mathbb{R}^3} \left\| T_i^{(f \to \text{Ref})^{-1}}(\mathbf{u}_{ij} + t\vec{\mathbf{v}}_{ij}) - T_j^{(f \to \text{Ref})^{-1}}(\mathbf{u}_{ij} + t\vec{\mathbf{v}}_{ij}) \right\|^2. \quad (5)$$

Here, we use 20 values of t in the range $[-10s..10s]$ where s is the mesh resolution (or grid sample spacing)[2]. In the case of a ball joint, we set $\vec{\mathbf{v}}_{ij} = 0$, so this term constrains only one point $\mathbf{u}_{ij}$. For a hinge joint, it constraints a set of points along the hinge axis. Inverses of the transformations are used in this term because the joint locations are defined on the reference frame.

Detecting Joint Locations. To detect joints and estimate their locations, we first find which pairs of transformations (i, j) are likely to share a joint in between, and we determine the exact location using the transformations $T_i^{(1\to\text{Ref})}, T_j^{(1\to\text{Ref})}, T_i^{(2\to\text{Ref})}, T_j^{(2\to\text{Ref})}, T_i^{(3\to\text{Ref})}, T_j^{(3\to\text{Ref})}(\mathbf{x}), \ldots$ for each frame.

We use the connectivity of the ASG to find pairs of transformations (i, j) likely to have a joint. To simplify the discussion, we say that an edge in the ASG is incident to transformation i if one of its end points $\mathbf{x} \in S$ has weight $\mathbf{w}_{\mathbf{x}i} = 1$. If either of the following ratios exceeds a threshold (set to 15%):

$$\frac{\#\text{ edges incident to both } i, j}{\#\text{ edges incident to } i}, \quad \frac{\#\text{ edges incident to both } i, j}{\#\text{ edges incident to } j} \quad (6)$$

we take the pair i, j as a candidate for sharing a joint. Also, the average of all endpoints of edges incident to both i, j gives a rough estimate of the joint location. This average is computed on the reference frame, and we denote it as $\mathbf{u}_{\text{est}}$.

Once we have a set of candidate pairs (i, j) and estimated joint locations $\mathbf{u}_{\text{est}}$, we solve for joint locations $\mathbf{u}$ on the reference frame based on the solved transformations. We perform a least-squares minimization for each pair (i, j):

$$\underset{\mathbf{u} \in \mathbb{R}^3}{\text{argmin}} \sum_{\text{All Frames } F_f} \left\| T_i^{(f \to \text{Ref})^{-1}}(\mathbf{u}) - T_j^{(f \to \text{Ref})^{-1}}(\mathbf{u}) \right\|^2. \quad (7)$$

We solve the minimization using the SVD, and we detect hinges by examining if the ratio of the smallest singular value to the sum of

[2] A similar approach is also used by Knoop et al. [2005].

the singular values is less than a threshold (set to 0.1). If this is the case, then we truncate the smallest singular value to zero and solve for the equation of the line $\mathbf{u}' + t\vec{\mathbf{v}}'$ satisfying the system. Finally, for the hinge joint parameters $(\mathbf{u}, \vec{\mathbf{v}})$, we take the point $\mathbf{u}$ on this line that is closest to $\mathbf{u}_{\text{est}}$ and normalize $\vec{\mathbf{v}} = \vec{\mathbf{v}}'/\|\vec{\mathbf{v}}'\|$.

If the joint is not a hinge, it is a ball joint and we determine a single joint location $\mathbf{u}$. In this case, we add an additional regularization term $\lambda \|\mathbf{u} - \mathbf{u}_{\text{est}}\|^2$ in the optimization, where λ is typically 0.1 [Pekelny and Gotsman 2008]. This additional term helps to pull the location closer to $\mathbf{u}_{\text{est}}$ in case the joint is close to being a hinge and admits multiple solutions.

Weight Objective $\mathcal{E}_{\text{weight}}$. Constraining the solution to solve for binary weights transforms the problem into a discrete labeling problem, where we try to find an optimal assignment of transformations to the sample points $\mathbf{x} \in S$. The goal of the weight objective is to constrain neighboring samples to have a similar weight. This way, sets of samples with the same weight form well-connected and contiguous regions on the ASG.

We use a simple constant penalty when two neighboring weights are different:

$$\mathcal{E}_{\text{weight}}(\mathcal{W}) = \sum_{(\mathbf{x},\mathbf{y})\in E} I\left(\mathbf{w}_{\mathbf{x}} \neq \mathbf{w}_{\mathbf{y}}\right), \tag{8}$$

where $I(\cdot)$ is 1 if the argument is true and 0 otherwise, and E is the set of all edges in the ASG. This is known as the Potts model, a discontinuity-preserving interaction term widely used for labeling problems [Boykov et al. 2001].

5.2 Optimization

To perform the optimization, we divide the solver into two phases and alternate between each phase until the solution converges (see Algorithm 2). In the first phase, we keep the weights fixed and solve for the transformations (lines 4-11), and in the second phase, we keep the transformations fixed and solve for the weights (lines 15-23). This strategy works well in practice and produces a good alignment within a few iterations. Also, we try to detect if previously occluded parts have reappeared in the new frame (line 12).

In our experiments, we observed that the transformations for a frame does not change much after the frame is first introduced. Therefore, we solve for the transformations only on the newest k frames that have been optimized. We can think of this as a "sliding window" in which to optimize the transformations. Lowering the value of k improves the speed of the registration, while raising this value may produce a more accurate registration at the cost of speed. Note that this only affects optimizing the transformations; the weights are always optimized using all frames.

Optimizing the Transformations. For optimizing the first phase, we 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, while keeping the weights fixed. Since the closest corresponding points $\mathbf{y}_j^{(g)}$ depend on the transformations, we use an iterative approach in the spirit of the iterative closest point (ICP) algorithm [Besl and McKay 1992]. We first keep the transformations fixed and compute the closest points, then we keep the corresponding points $\mathbf{y}_j^{(g)}$ fixed and optimize the transformations, and we repeat this alternation until convergence. Also, we only update the corresponding points for component j^*, since the index j^* is determined because the weights are kept fixed.

We perform the optimization using the Gauss-Newton algorithm, linearizing the objective function in each iteration by substituting a linearized form of each rigid transformation. To solve for the transformations on a limited number of frames, we can simply remove the variables/constraints (and also not update target points) involving transformations from frames outside of the set of interest. This significantly reduces the time to perform this step.

Algorithm 2: OPTIMIZE $\mathcal{T}, \mathcal{W}\ (S, E, \mathcal{T}, \mathcal{W}, F_0, \ldots, F_{\text{new}})$

Data: Sample set S with associated weights $\mathcal{W}$, transformations for all frames $\mathcal{T}$, A list of edges E of the constructed ASG, all initialized input frames $F_0, \ldots, F_{\text{new}}$
Result: Optimized transformations and weights $\mathcal{T}, \mathcal{W}$

```
1  begin
2      Select a subset of frames to optimize the transformations
       (e.g. a sliding window of 1–10 frames);
3      while Not converged do
4          begin (Phase 1: Solve for the transformations T)
5              Re-estimate joint locations and types;
6              while Not converged do
7                  Update the closest points y_{j*}^{(g)} for all x ∈ S
                   and frames F_g;
8                  Construct the sparse matrices for E_fit and E_joint;
9                  Solve linear system and update
                   transformations;
10                 Check convergence criteria;
11         end
12         Handle reappearing parts in F_new by aligning missing
           parts with unmatched surface points (Section 7);
13         Check convergence criteria;
14         if converged then break;
15         begin (Phase 2: Solve for the weights W)
16             Update the closest points y_j^{(g)} for all x ∈ S,
               frames F_g, and j;
17             Precompute E_fit for each x ∈ S and j;
18             Create a graph for E_weight using the edges E of the
               ASG;
19             Solve discrete labeling on this graph using
               α-expansion;
20             Discard parts that are too small;
21             Reuse unassigned weight components by splitting
               regions with highest E_fit error;
22             Update the weights for each x ∈ S;
23         end
24 end
```

Optimizing the Weights. For the second phase, we solve for the weights of each sample point $\mathbf{x}$ that minimize the terms $\alpha\ \mathcal{E}_{\text{fit}} + \gamma\ \mathcal{E}_{\text{weight}}$, while keeping the transformations fixed. Since we constrain the weights to be binary, we are essentially determining j^* for each sample point that minimize the total error. We solve this discrete optimization problem using the α-expansion algorithm [Boykov et al. 2001; Boykov and Kolmogorov 2004; Kolmogorov and Zabih 2004]. Here, we use the ASG directly to specify smoothness constraints between points. To save computation time during the optimization, we precompute $\mathcal{E}_{\text{fit}}$ and store the values in a hash table for quick access. We compute and store the summand of $\mathcal{E}_{\text{fit}}$ for all samples $\mathbf{x}$, all j, and all frames g.

After the optimization, it may be the case that some transformations are applied to very few sample points. If the number of sample points for a transformation is less than 1% of the total number of sample points, then we remove the transformation completely and

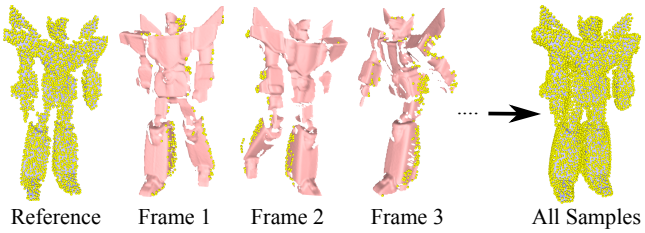


Fig. 6. We represent the vertex weight function using sample points taken from all input frames. For each frame, we only keep samples that are not overlapping with samples from previous frames.

substitute different weights (taken from nearby samples). This is useful for cleaning up noise in the solution.

We further reduce the registration error by reusing transformations that are not applied to any sample points. We split the region with the highest registration error in half and introduce the unused transformation by replacing the weights in one of these halves [Chang and Zwicker 2009]. The splitting is performed like the k-means algorithm, with two randomly selected samples in the region. To aid our method, the user also specifies a maximum number of rigid transformations B that should be used to approximate the surface motion. The splitting process is continued until the highest registration error is below a threshold (typically $0.1s$), or we have reached the maximum number of transformations.

Checking for Convergence. We perform the convergence check (Algorithm 2, lines 13-14) right after solving for the transformations, because the optimization is usually able to refine the transformations further after the weights have changed. To detect if the optimization for the transformations has converged, we monitor the change of the objective function by examining the value of the minimized residual. We apply the criterion $|F_k - F_{k+1}| < \epsilon(1 + F_k)$ (where $\epsilon = 1.0 \times 10^{-6}$) and stop the iteration if this condition is met. We also have a maximum number of iterations, typically set to about 20–30 iterations, and stop if we exceed this maximum number. In our experiments, we observed that in most cases the optimization converges in about 10–15 iterations. However, the optimization may enter an oscillating mode, where the closest points in each iteration of the T-step switch back and forth indefinitely between a few points. Because of this, convergence is not guaranteed; but in practice we have not encountered any major problems.

In the next few sections, we discuss the remaining details of our method that involves the sample set S and handling missing parts.

6. SAMPLE SET AND ALL-SAMPLES GRAPH (ASG)

The sample set is used as a sparse representation of the weights. In this section, we fill in details about how we perform the sampling, detect and remove overlapping samples, and interpolate weights. We also provide details about transferring the coarse registration result to S.

Creating the Sample Set and ASG. We create the sample set by first uniformly sampling a set of points U_f in each frame f. We sample a fixed fraction of the points in each frame using the best-candidate technique [Mitchell 1991].

We then merge all samples by sequentially adding the points from each U_f to S. During this process, we detect and remove overlapping points in U_f that overlap with the currently selected points in S [Pekelny and Gotsman 2008]. For detection, we first transform all points currently in S to frame F_f. For each $\mathbf{x} \in U_f$, we find the closest point $\mathbf{y}$ in S and compute $d_{\text{pt}} = \|\mathbf{x} - \mathbf{y}\|$ and $d_{\text{OnPl}} = \sqrt{\|\mathbf{x} - \mathbf{y}\|^2 - ((\mathbf{x} - \mathbf{y}) \cdot \vec{\mathbf{n}}_{\mathbf{y}})^2}$. We consider $\mathbf{x}$ to be overlapping if these distances are smaller than a user-given threshold τ_s; we threshold d_{pt} if the surface normals differ by more than 90 degrees, otherwise we threshold d_{OnPl}. This process removes redundant samples and improves the efficiency of the algorithm.

We construct the ASG by transforming all samples to the reference frame and computing the k-nearest neighbor graph of the samples, with $k = 15$. To prevent undesired smoothness constraints between separate (but spatially near) parts, we measure the length of each edge in all frames and discard edges that stretch in length more than 2 times. We observed that pruning edges between connected parts may bias the discrete labeling optimization and prevent changes in the boundary location. Since this may cause the optimization to converge to the wrong local minimum, we keep all edges between parts that have a joint.

Interpolating Sample Weights. After we finish aligning each frame, we need to add the frame's samples to S. In this case, we also need to determine the weights for these new samples based on the weights already assigned to S. In addition, when we resample S from scratch to create a new sample set, we need to transfer the weights from the old set to the new one.

To accomplish these tasks, we interpolate the weights in S using a nearest-neighbor like approach [Pekelny and Gotsman 2008]. We first transform all points in S to frame F_f. Then, we partition them into separate sets V_j for each transformation j. To determine the weight of a new location $\mathbf{p}$, we compute the distance of $\mathbf{p}$ to the nearest point in each V_j. The set V_{j^*} with the closest distance wins: the binary weight of $\mathbf{p}$ has 1 in component j^*. In some cases, the distances are too close to declare a clear winner. We compute a score by inverting all distances, and then normalizing them to add up to 1. If the maximum score is not greater than three times the upper quartile (median of the largest half) of all scores, we consider it an ambiguous case and mark the weight as invalid. The weight is also invalid if the winning transformation j^* is flagged as occluded for this frame.

Applying Coarse Registration to S. After computing the coarse initial registration, we need a mechanism to apply this alignment information in a format compatible with S. The coarse registration result aligning frame F_i to F_{i+1} gives a transformation for *every vertex* of F_i. However, we represent the alignment *indirectly* using weights on the sparse sample set S. We need to reduce the set of transformations so that we can apply a single transformation per weight component.

First, we divide S into regions grouped according to weight. For each region, we extract a single transformation from the coarse registration. For each point $\mathbf{x}$ in the region, we find the closest point $\mathbf{y}$ in F_i and store (in a list) the transformation that was assigned to $\mathbf{y}$ in the coarse registration. This results in a list of transformations for the region. Finally, we uniformly blend all transformations in the list using DLB [Kavan et al. 2008] to produce a single transformation. We can now apply this to the region's points to reproduce the coarse alignment.

In practice, this may produce a slightly different result from the coarse registration, but the differences were negligible. Also, since the transformations are specified relative to a reference frame, we take care to express the transformations properly to fit this format.

7. HANDLING MISSING PARTS

When a part of the surface is partially or completely missing in a frame, the transformation for this part may have few or no valid

correspondences constraining it in the optimization. In these cases, it may not be possible to solve for the rigid transformation of that part. In our algorithm, we automatically detect this and exclude these parts from the optimization (line 8 of Algorithm 1).

As before, let us divide S into parts grouped according to weight. To decide if a part is occluded, we update the closest points for this frame after loading and applying the coarse registration (line 4 of Algorithm 1). Then, we count the number of target positions $\mathbf{y}_j^{(g)}$ for each part. If this number falls below a small threshold (either below 5 points, or below 5% of the total number of samples for that label), then we consider the part as missing for this frame.

Instead of optimizing for the transformation of this part, we substitute a value computed based on the joint constraints with neighboring parts. If there are no neighbors, we use the value from the last frame; if there is exactly one, we copy the neighbor's value; and if there are two or more, we solve for the transformation that best fits all joint constraints [Pekelny and Gotsman 2008].

Missing parts also cause problems when optimizing the weights. Here, the algorithm must optimize the binary weight for each sample to minimize the registration error. The question is, what should be the "registration error" for a missing part? We need to assign some reasonable value for this case so that the weights are not assigned erroneously.

Recall that when we optimize the weights (Section 5.2), we compute and store the summand of $\mathcal{E}_{\text{fit}}$ for all samples $\mathbf{x}$, all j, and all frames g. Suppose that for some sample $\mathbf{x}$, transformation j is missing in frame g. For the error value of transformation j, we simply use the error value of transformation j^* (the index of the weight component with 1). If $j^* = j$ (i.e. j^* is also missing), we use the minimum error value among all non-missing transformations. This heuristic worked well for most cases in our experiments, except for a handful of instances where the missing transformation was assigned to a completely unrelated location.

Reappearing Parts. When an occluded part suddenly reappears in a new frame, we need to start tracking it again. Otherwise, the algorithm could mistakenly treat it as new surface geometry, thus duplicating the part multiple times in the reconstruction. Now, if the part happens to reappear nearby its last approximated location, then the algorithm will be able to find a sufficient number of closest points and automatically track the part again. However, if the part reappears in a completely different location, we need a different strategy since there will not be enough closest point correspondences. Note that this is not handled by our initialization step, because it can only align parts that appear in *both* the source and target.

To detect this case, we observe that a large number of scanned points in the frame will not overlap with the sample set S *after* initializing and optimizing the transformations. If the number of such unmatched points exceeds a threshold (10% of the total points in the frame), we attempt to align the occluded parts with these unmatched points. This is performed after each optimization of the transformations (Algorithm 2, line 12). Here, we use the same procedure to optimize for the transformations (Section 5.2), but with some changes where

—we optimize only for the occluded transformations,
—we set the closest point threshold τ_d and normal angle threshold τ_n much higher,
—and we increase the weight of the $\mathcal{E}_{\text{joint}}$ (β in Equation 2) to be very high.

Fig. 7. Reconstruction results for the robot dataset.

After this, we run the occlusion detection routine once more to check if we have obtained a sufficient number of target points to start tracking the part again.

8. EXPERIMENTAL RESULTS

8.1 Reconstruction

We implemented our algorithm in C++ and tested it with several real-world and synthetic datasets exhibiting articulated motion. After we have aligned all frames, we reconstruct a triangle mesh from a dense sampling of S produced using a small sample distance τ_s. We use the streaming wavelet surface reconstruction algorithm by Manson et al. [2008].

The car and robot datasets were acquired by Pekelny and Gotsman [2008] using a Vialux Z-Snapper depth camera. These sequences were created by animating the physical model, while capturing each frame from a different viewpoint. Each sequence has 90 frames, and consists of 5 and 7 parts, respectively. The results are shown in Figures 7 and 8. The top row shows some of the input frames in the sequence. Notice that there is a significant amount of occlusion in some of the frames. The second row shows the labeled sparse sample set S used by our algorithm, and the third row shows the dense sample set obtained using a smaller τ_{sample} in the post-processing step. There are still some holes on the surface, which are locations that were occluded in all input frames, or locations where the algorithm could not extrapolate the label (Section 6). The fourth row shows the reconstructed mesh using the algorithm by Manson

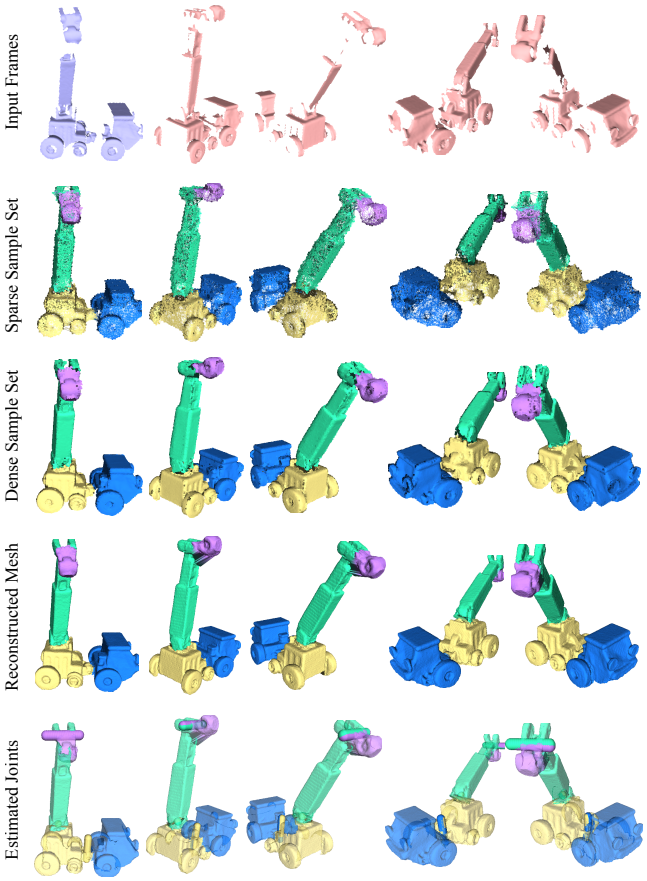


Fig. 8. Reconstruction results for the car dataset.

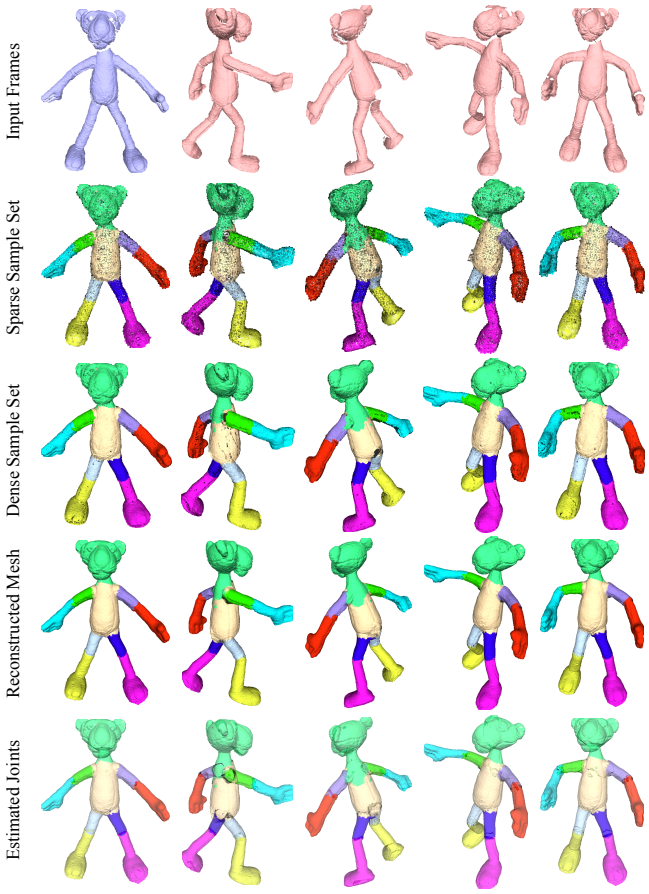


Fig. 9. Reconstruction results for the first Pink Panther dataset (small-motion case).

et al. [2008], with labels on each vertex obtained by taking the label of the nearest point in the dense sample set. Since we meshed a single, closed surface in the pose of the reference frame, there are some stretching artifacts on the boundary between neighboring parts. This could be corrected by meshing each part separately, or by meshing the surface in a pose where the parts are further apart. Finally, the fifth row shows the estimated joint locations. Hinge joints are represented by a short stick, where ball joints are representing using a sphere.

The reconstruction results for the robot and car datasets are good, demonstrating that we can obtain an accurate registration without a segmentation given as input by the user. For the car dataset, our algorithm preferred a simpler configuration of 4 parts, instead of creating a separate part for the small rotating base in the middle. We think that this is a reasonable reconstruction of the car, because the surface for the rotating base is quite small.

To test our algorithm on a more deformable subject, we acquired sequences of a bendable, poseable pink panther toy. These sequences were acquired using a Konica Minolta VI-910 laser scanner. Each sequence has 40 frames consisting of 10 parts each. In the first sequence, we animated the toy with small motions while capturing each frame at a different viewpoint. In the second sequence we created larger motions of the toy while changing the viewpoint. In addition, the furry texture on the toy created a significant amount of noise on the scanned surface. The reconstruction results, shown in Figures 9 and 10, are very good for both the small-motion and the large-motion case, except for some minor stretching artifacts on the reconstructed mesh, at the boundary between parts.

Finally, we generated synthetic depth sequences of a walking man, where the camera is rotating around the subject. These sequences were created by capturing the Z-buffer of an OpenGL rendering, and the modelview and projection matrices were inverted to convert the depth values into 3D coordinates. To test the effect of occlusion in our algorithm, we captured the first sequence using a single camera, and the second sequence using two cameras which were 90° apart. Since the frames were very close to each other, we did not use the initialization step for these sequences. Also, we reduced the sliding window size from 5 frames (for the first $\sim$10 frames) down to 1 frame (for the rest of the sequence).

The reconstruction results are shown in Figures 11 and 12. The first sequence was less successful due to the large amount of occlusion of the arms. In particular, both the left arm and right arm disappear during a front view and reappear in a back view, causing alignment errors in the middle of the sequence. This resulted in a "larger" left hand, because the algorithm did not align the hand well and added extra points for this part. Also, in some of these frames, the arm and hand partially appeared but was not tracked, and this resulted in some "floating parts." Nevertheless, the reconstructed mesh and segmentation nicely captures the overall shape and motion of the subject.

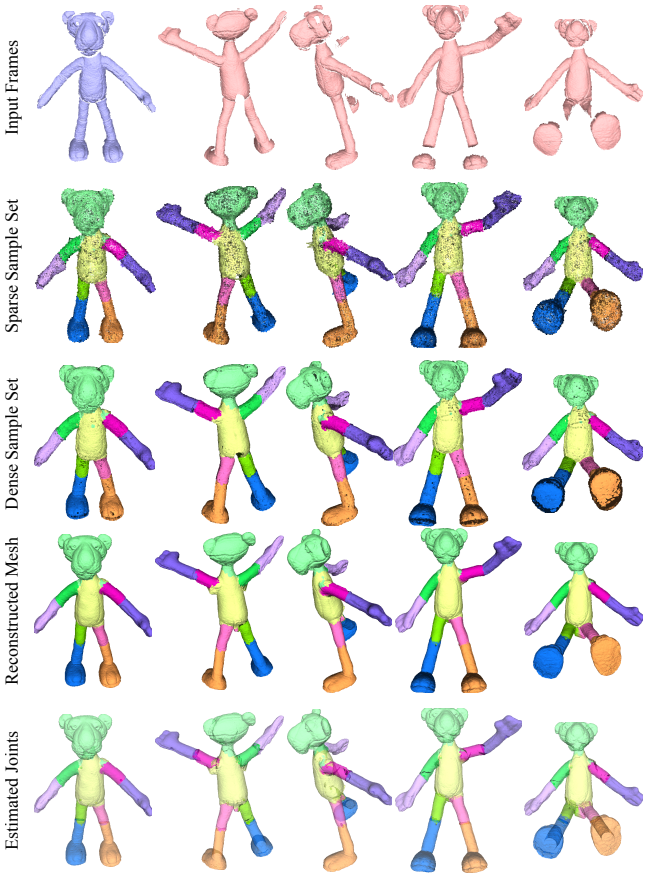


Fig. 10. Reconstruction results for the second Pink Panther dataset (large-motion case).

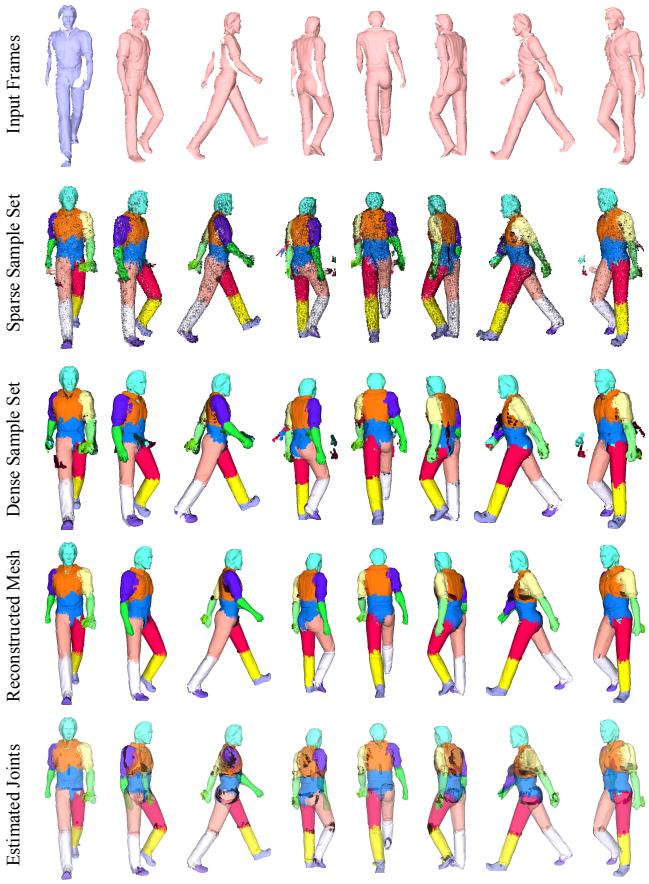


Fig. 11. Reconstruction results for the synthetic Walking Man dataset taken using a single virtual camera.

In the second walking man dataset acquired with two virtual cameras, the arm and hand do not disappear completely, and so the algorithm is able to track all parts accurately for the entire sequence. This results in a very accurate reconstruction (especially for the hands). However, the stretching artifacts on the reconstructed mesh (fourth row) are more noticeable. This happens in the region where the torso and arm connect together, and also the hip region where the surface stretches significantly.

8.2 Parameters

The main parameters of our algorithm are the the number of transformations B, weights for each term in the optimization and thresholds that control sampling and closest point computation. Although the user needs to specify the number of transformations to approximate the motion, the algorithm may settle on a smaller number of transformations if the registration error is small enough. An alternative strategy would be to have the user specify a maximum alignment error ϵ and make the algorithm add part labels until the alignment error is accurate within this ϵ. We did not explore this alternative, but this ϵ parameter would be similar to directly specifying the number of transformations.

We expressed many parameters relative to the grid sample spacing s, which is the average distance between samples in each frame. For the weights of each term in Equation 2, we used $\alpha = 1$, β between 0.1 and 1.5, and γ either $0.5s$ or s. For the uniform subsampling U_f (Section 6), we specified a fraction of points to sample for the entire sequence, typically between 6% and 20% depending on the density of the scans. For the sample spacing parameter τ_s, we used a value between $2s$ and $5s$ depending on how dense we wanted the sparse sample set to be. Finally, for determining if the closest point is valid (Section 0**??**), we used $\tau_d = 10s, \tau_n = 45^\circ$, and $\tau_b = s$. This changes when we match reappearing parts, for which we used τ_d between $50s$ and $100s$, τ_n between 45° and 80°, and $\beta = 100$. In our experiments, we experimented with a few different parameter settings but did not seriously optimize the parameters to give a better result.

8.3 Performance

The performance of our implementation using a single core of an Intel Xeon 2.5 GHz processor is reported in Table I. In the robot and car datasets, the most time-consuming part was the initialization, but in the other cases it was the global registration. The global registration step can execute faster if a smaller sliding window is used, with the trade-off of having a less accurate registration. Like most ICP-based algorithms, the most time-consuming part is the closest point computation, which can typically take 30% of the total time. Note that the times in the initialization step reported in Table I do not include the preprocessing time to compute spin images and estimate the principal curvature frame at each vertex.

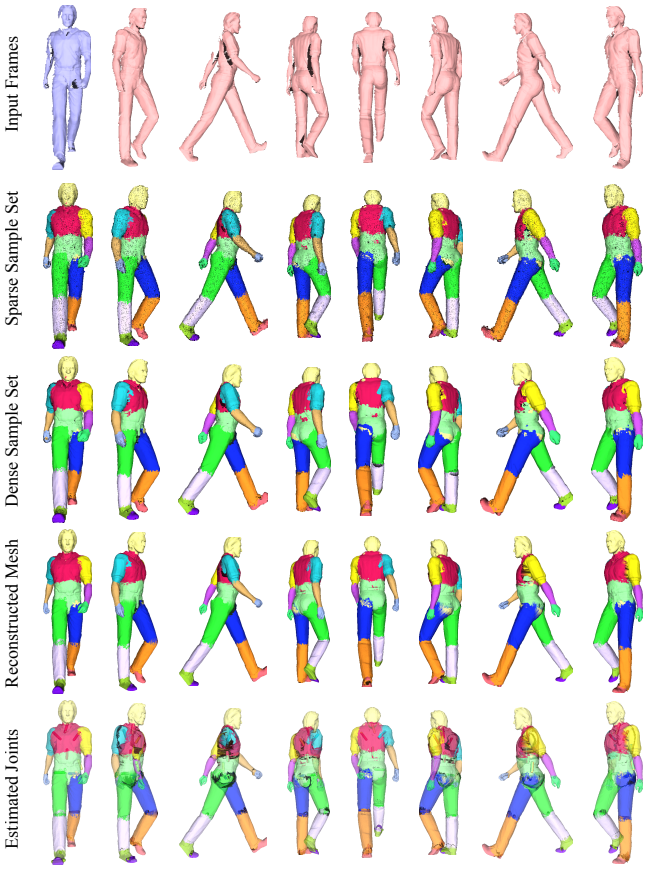


Fig. 12. Reconstruction results for the synthetic Walking Man dataset taken using two virtual cameras.

Table I. Performance statistics for our experiments. The timings are expressed in seconds, and the bottom row reports the average execution time per frame in each sequence.

Statistic	Robot	Car	PP1	PP2	Walking1	Walking2
Max Bones	7	7	10	10	16	16
Used Bones	7	4	10	10	14	16
Frames	90	90	40	40	121	121
Sliding Window	5	5	5	5	$5 \rightarrow 1$	$5 \rightarrow 1$
Points/Frame	9,391.2	5,387.86	36,683.9	30,003.1	19,843.7	39,699.7
Total Points	845,208	484,907	1,227,356	1,200,125	2,401,082	4,803,662
Samples	4,970	2,672	4,077	4,203	8,305	8,539
Edges in ASG	37,678	20,707	30,758	31,841	61,711	63,043
Initialization	7,357.68	2,652.57	1,826.27	1,828.98	69.38	134.74
Global Reg	2,287.61	1,200.04	2,184.68	2,624.4	5,574.86	19,789.0
Resampling ASG	264.44	117.93	67.90	68.06	876.32	1,617.07
Total Time	9,909.73	3,970.54	4,079.85	4,521.44	6,520.56	21,540.81
Average Time	110.11	44.12	102.00	113.04	53.89	178.02

8.4 Inverse-Kinematics Application

Solving for the weights and joints in the model is useful for reposing and animating the reconstructed model. To demonstrate this, we implemented a tool to perform inverse kinematics on the reconstructed model. In this system, the user specifies point constraints interactively by drawing boxes around a region of interest. Then, the user is able to select one of the constraint boxes and drag it around on the screen to manipulate the model. To perform IK, we use the transformation optimization (Section 5.2) to solve for the rigid transformations of each part that best satisfy the constraints. The details of the optimization are exactly the same as before, except that the joint locations are fixed, and the correspondences are given by the user. By running the optimization in a separate background thread, we were able to interactively manipulate the reconstructed model in real-time. Figure 13 shows examples of different poses of the robot created by our system.

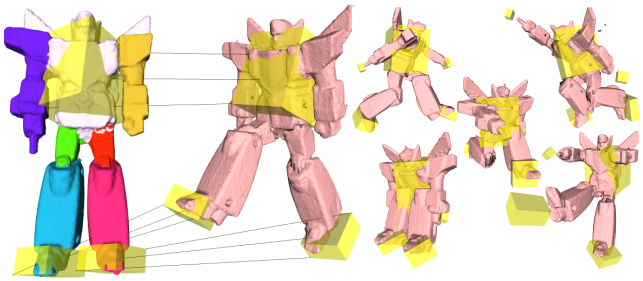
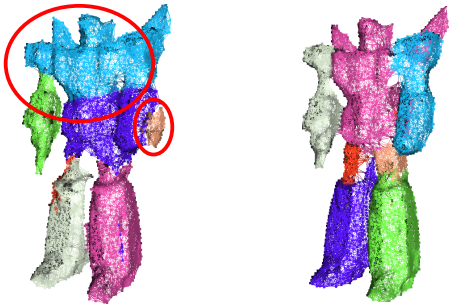


Fig. 13. Reposing the reconstructed robot. By using the solved weights and the hinge joints, our optimization can satisfy point constraints given by the user.



(a) Result Using Sequential Registration

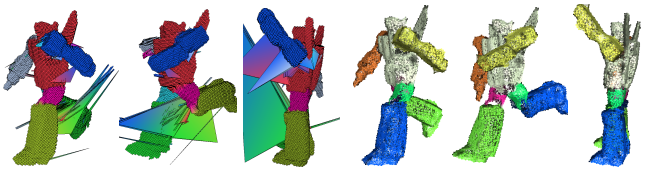
(b) Result Using Simultaneous Registration

Fig. 14. Comparing sequential and simultaneous registration. (a) As indicated by the large red circle on the upper body area, the sequential strategy gives an unreliable estimate of the articulated structure, because it only uses the movement observed in one frame. This leads to an imprecise registration, for example, in the left arm indicated by the smaller circle. (b) The simultaneous strategy can correctly estimate the structure that fits the movement observed in all frames. The registration is more precise, as well as the estimated surface geometry.

8.5 Sequential Registration vs. Simultaneous Registration

To illustrate the benefit of performing simultaneous registration, we compare our algorithm with a sequential registration pipeline. In a sequential registration method, we optimize each frame of the sequence one-by-one, accumulate new samples directly on the reference frame, and discard the frame before moving on to the next. Therefore, this strategy is essentially a pairwise registration that is applied repeatedly for each frame, because it only uses correspondences between the accumulated samples and the current frame for estimating both transformations and weights.

The main problem with the sequential registration approach is that it cannot reliably estimate the articulated structure (i.e. weights) based on the movement observed in just one frame. This complicates the situation further for occlusion detection and recovery, which rely on a reliable estimate of the articulated structure. A



(a) Result Using a Grid-Based Representation (144.00 sec/frame)

(b) Result Using a Graph-Based Representation (28.36 sec/frame)

Fig. 15. Comparing grid-based and graph-based weight representations. These images show the represented weight function, deformed into different poses according to the optimized transformations and weights. Notice the deformation artifacts with the grid-based representation, which is absent in the graph-based representation.

comparison between the sequential and simultaneous strategies is shown in Figure 14. Here, we have used the two strategies to align 40 robot frames, and we display the sparse ASG which roughly shows the estimated geometry. On the left, we can see that the sequential strategy did not produce a correct labeling. As a result, the registration was imprecise, and "extra" surfaces appear where the parts were not aligned properly (for example, on the left arm). On the right, we show a result obtained by simultaneous registration, where we kept the same parameters, used a 1-frame window for optimizing the transformations, and used the correspondences from all frames to optimize the weights. The result has a correct labeling that reflects the movement in all frames, and the registration and estimated geometry are precise.

8.6 Grid-Based Weights vs. Graph-Based Weights

To compare the benefit of using a graph for defining the weight function vs. using a grid, we implemented the simultaneous registration using a grid and compared the results. First, we found that the performance of the graph-based registration is much faster, because the grid-based method has an additional overhead of translating the weights from the grid to the samples. For processing the 90 frame robot sequence, the global registration took a total of 144.00 seconds per frame using the grid strategy, but it only took 28.36 seconds per frame for the graph based strategy (excluding initialization time).

Second, the graph-based representation dealt robustly with topology issues. An example of this is shown in Figure 15, where we display the grid and graph deformed according to the optimized weights and transformations. Unlike the graph based solution on the right, the grid based solution on the left shows many artifacts. This is because when the resolution of the grid is too coarse, a single grid cell overlaps multiple separate parts. In this example, there are several grid cells that overlap a little with both the right leg and the left leg of the robot. As a result, different weights are assigned to either side of the cell, so the cell "stretches" apart, causing the artifact that we see. This stretching behavior makes it difficult to look up weights for the scanned points inside this cell, and so we "lose" points in these situations. In contrast, for the graph-based strategy, since we define weights directly on each sample, it does not suffer from this issue. Furthermore, we can prune edges of the graph based on the optimized motion, so it can handle topologically difficult cases robustly.

8.7 Comparison with Wand et al. [2009]

We compare our articulated reconstruction with the deformable reconstruction method by Wand et al. [2009]. For the car, robot, and pink panther datasets, their method was not able to fully reconstruct these sequences because there was too much motion between the frames. Instead, they were able to reconstruct only some subsequences of these datasets. This is because they rely only on a local optimization using closest points, whereas our method uses a robust initialization that is able to automatically handle frames with large motion.

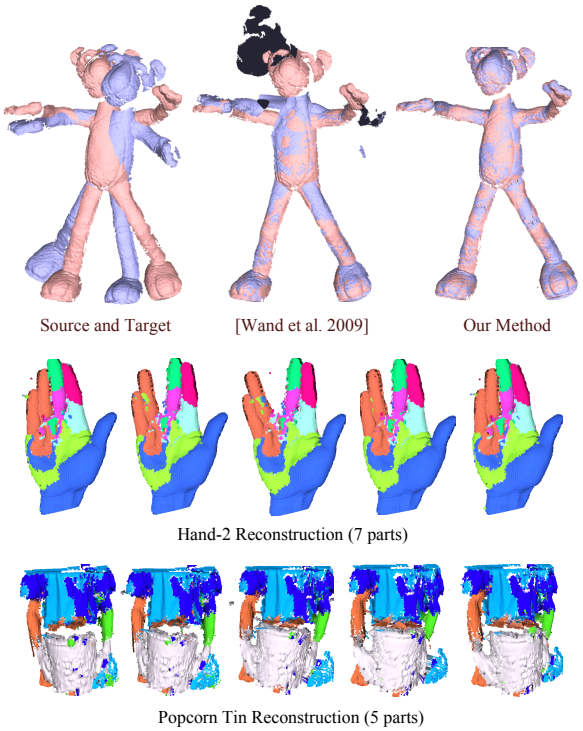


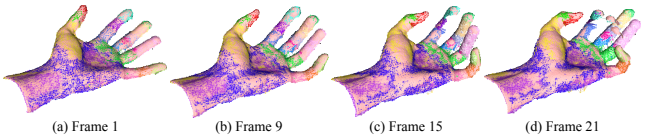
Fig. 16. Articulated registration on the hand-2 and popcorn tin datasets used by Wand et al. [2009]. Our algorithm is able to produce coarse approximations of the non-rigid motion exhibited in these datasets.

However, the method by Wand et al. [2009] was able to reconstruct subsequences of these datasets. To compare the quality of the registration, Figure 16 (left) shows a comparison for registering two robot frames. For this example, the end result is similar, but the fully deformable approach of Wand et al. [2009] is numerically more difficult to handle than our method. This is because they require a relatively high resolution deformation field (a 24^3 resolution regular grid; using a 12^3 resolution did not work), whereas we only estimated a total of 7 rigid transformations. Thus, our articulated approach is likely to be numerically superior when applicable.

We also tested our algorithm on several examples from Wand et al. [2009]. Figure 16 (right) shows reconstructions of the hand-2 and popcorn tin datasets, and Figure 17 shows a result for the grasping hand (hand-1) dataset. These sequences exhibit non-rigid motion, especially the popcorn tin dataset. Our algorithm can successfully capture the overall shape and produce a coarse articulated motion of the subject. However, we see that it does not reproduce fine details in the surface deformation.

9. SUMMARY AND CONCLUSION

We have presented a method to reconstruct an articulated shape from a set of range scans. From a sequence of range scans, we solve for the division of the surface into parts (weights) and the motion for each part (transformations) to align all input scans. For



(a) Frame 1 (b) Frame 9 (c) Frame 15 (d) Frame 21

Fig. 17. Registration for a grasping hand sequence [Weise et al. 2007], where the hand starts from an open pose and gradually closes to a grasping pose. Shown are the input data (displayed as a red color mesh) and the sparse ASG. Our algorithm tracks the hand well in the first part of the animation, where most of the surface is visible. In (c), the surface of the fingers start to gradually disappear, and the middle segment of the index finger starts to lose track and rotate backwards. In (d), the algorithm loses track of the middle and ring fingers, because most of these fingers are occluded (except for the fingertips).

this purpose, we first improved a transformation sampling and assignment strategy to obtain a robust initialization of the registration between pairs of adjacent frames in the sequence. Then, we formulated a simultaneous registration for all input frames to minimize registration error. This optimization included joint constraints that preserves the connectivity of each part, and automatically handles cases where parts are occluded or they reappear. We demonstrated that we can reconstruct a full 3D articulated model without relying on markers, an user-provided segmentation, or a template. Finally, we have demonstrated that the reconstructed model is deformable and can be interactively manipulated into new poses using a simple extension of our optimization algorithm. The main advantages of our method is that it can align range scans with fast motion and significant occlusion, and that it produces a rigged 3D model.

A limitation of our method is that there needs to be enough overlap between adjacent frames in the range scan sequence to obtain a good alignment. For example, if one frame captures the surface on the front of the object, and the next frame has the surface from the back of the object, then there will be not enough overlap to match these frames together in the registration. This means that the order of the range scans in the sequence should maintain a reasonable amount of overlap between every adjacent pair of frames. A temporal ordering of the scans, for example, would produce a sequence with a reasonable amount of overlap. Sometimes even this is not enough when there is severe occlusion. For example, our algorithm loses track of the fingers in the hand sequence because of too much missing data, as shown in Figure 17.

Another shortcoming of our ICP-based registration is the handling of "slippable" parts such as cylinders. For example, the fingers of a hand example shown in Figure 17 have cylindrical symmetry, and the ICP registration could converge into a state where the segments of the fingers are "twisted" or rotated about the axis of symmetry (Figure 17c). Although hinge joints could disambiguate cylindrical symmetries, we found that it was difficult to estimate accurate hinge joints in this case.

Currently our method is applicable for reconstructing articulated subjects and coarsely capturing non-rigid subjects. However, it would be interesting to adapt our algorithm for high-quality non-rigid reconstruction. For this case, estimating "flexible" transformations would be appropriate, for example, estimating affine transformations with additional surface displacements. Also, it would be useful to find a way to optimize for smooth weights without causing overfitting. We believe that there should be a middle ground between solving for separate transformation for every sample point [Li et al. 2008] and our method of solving for the weight at each sample point.

Our algorithm does not estimate scale, so it cannot handle the range scans where the scale of the object changes. While this was not a problem for any of our examples, automatically estimating scale changes could help capture regions that are stretching.

We would also like to reduce the parameters in our algorithm. An alternative to specifying various thresholds is to use robust error metric similar to the work of Nishino and Ikeuchi [2002]. In this case, the outliers would automatically be identified during the optimization, without a need to specify hard thresholds.

Finally, we would like to investigate ways of improving the performance of the algorithm. In particular, since our method estimates the weights and transformations for all frames simultaneously, we need to keep all of the input scans in memory. We would like to develop a streaming version of our algorithm that reduces the memory requirements and allows us to process longer sequences. In addition, once a reasonable segmentation is obtained, only the transformations need to be solved for each frame. We believe that this could be implemented in real-time to be used for markerless motion capture applications.

ACKNOWLEDGMENTS

We would like to thank M. Wand for providing comparisons and useful feedback, and S. Buss for helpful discussions. We also wish to thank Y. Pekelny and C. Gotsman for sharing the car and robot datasets, T. Weise for the grasping hand dataset, O. Schall for the hand-2 dataset, and P. Fong for the popcorn tin dataset. Additional thanks to G. Debunne for providing the libQGLViewer library, D. M. Mount and S. Arya for providing the ANN library, and Y. Boykov, O. Veksler, R. Zabih for providing an implementation of their graph cuts algorithm, and J. Manson for providing surface reconstruction software.

REFERENCES

AHMED, N., THEOBALT, C., DOBREV, P., SEIDEL, H.-P., AND THRUN, S. 2008. Robust fusion of dynamic shape and normal capture for high-quality reconstruction of time-varying geometry. In *CVPR*.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2002. Articulated body deformation from range scan data. *ACM SIGGRAPH 21,* 3, 612–619.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM SIGGRAPH*. 587–594.

ANGUELOV, D., KOLLER, D., PANG, H., SRINIVASAN, P., AND THRUN, S. 2004. Recovering articulated object models from 3d range data. In *Uncertainty in Artificial Intelligence Conference (UAI)*.

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. Scape: shape completion and animation of people. In *ACM SIGGRAPH*. 408–416.

ANGUELOV, D., SRINIVASAN, P., PANG, H.-C., KOLLER, D., THRUN, S., AND DAVIS, J. 2004. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *NIPS*.

BESL, P. J. AND MCKAY, H. 1992. A method for registration of 3-d shapes. *IEEE TPAMI 14,* 2, 239–256.

BOYKOV, Y. AND KOLMOGOROV, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI 26,* 9 (September), 1124–1137.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE TPAMI 23,* 11, 1222–1239.

BRADLEY, D., POPA, T., SHEFFER, A., HEIDRICH, W., AND BOUBEKEUR, T. 2008. Markerless garment capture. *ACM SIGGRAPH 27*.

CHANG, W. AND ZWICKER, M. 2008. Automatic registration for articulated shapes. *Comput. Graph. Forum (Proc. SGP) 27,* 5, 1459–1468.

CHANG, W. AND ZWICKER, M. 2009. Range scan registration using reduced deformable models. *Comput. Graph. Forum (Proc. Eurographics) 28,* 2, 447–456.

DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H.-P., AND THRUN, S. 2008. Performance capture from sparse multi-view video. *ACM SIGGRAPH*.

DE AGUIAR, E., THEOBALT, C., THRUN, S., AND SEIDEL, H.-P. 2008. Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum (Proceedings of Eurographics) 27,* 2, 389–397.

GALL, J., STOLL, C., DE AGUIAR, E., THEOBALT, C., ROSENHAHN, B., AND SEIDEL, H.-P. 2009. Motion capture using joint skeleton tracking and surface estimation. In *CVPR*.

HUANG, Q.-X., ADAMS, B., WICKE, M., AND GUIBAS, L. J. 2008. Non-rigid registration under isometric deformations. *Computer Graphics Forum (Proceedings of SGP) 27,* 5, 1449–1457.

KAVAN, L., COLLINS, S., ZÁRA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics 27,* 4.

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

KOLMOGOROV, V. AND ZABIH, R. 2004. What energy functions can be minimized via graph cuts? *IEEE TPAMI 26,* 2, 147–159.

LI, H., ADAMS, B., GUIBAS, L. J., AND PAULY, M. 2009. Robust single view geometry and motion reconstruction. In *ACM SIGGRAPH ASIA, to appear*.

LI, H., SUMNER, R. W., AND PAULY, M. 2008. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum (Proceedings of SGP) 27,* 5, 1421–1430.

MANSON, J., PETROVA, G., AND SCHAEFER, S. 2008. Streaming surface reconstruction using wavelets. In *SGP*.

MITCHELL, D. P. 1991. Spectrally optimal sampling for distribution ray tracing. *ACM SIGGRAPH*, 157–164.

MITRA, N. J., FLORY, S., OVSJANIKOV, M., GELFAND, N., GUIBAS, L. J., AND POTTMANN, H. 2007. Dynamic geometry registration. In *SGP*. 173–182.

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

NISHINO, K. AND IKEUCHI, K. 2002. Robust simultaneous registration of multiple range images. In *ACCV*.

PARK, S. I. AND HODGINS, J. K. 2006. Capturing and animating skin deformation in human motion. *ACM Trans. Graph. (Proc. SIGGRAPH) 25,* 3, 881–889.

PARK, S. I. AND HODGINS, J. K. 2008. Data-driven modeling of skin and muscle deformation. *ACM Trans. Graph. (Proc. SIGGRAPH) 27,* 3.

PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M., AND GUIBAS, L. J. 2005. Example-based 3d scan completion. In *SGP*. 23.

PEKELNY, Y. AND GOTSMAN, C. 2008. Articulated object reconstruction and markerless motion capture from depth video. *Computer Graphics Forum (Proceedings of Eurographics) 27,* 2.

SCHAEFER, S. AND YUKSEL, C. 2007. Example-based skeleton extraction. In *SGP*. 153–162.

SHARF, A., ALCANTARA, D. A., LEWINER, T., GREIF, C., SHEFFER, A., AMENTA, N., AND COHEN-OR, D. 2008. Space-time surface reconstruction using incompressible flow. *ACM SIGGRAPH ASIA*.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. In *ACM SIGGRAPH*. 80.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIC, J. 2005. Mesh-based inverse kinematics. *ACM Trans. Graph. (Proc. SIGGRAPH) 24,* 3, 488–495.

SÜSSMUTH, J., WINTER, M., AND GREINER, G. 2008. Reconstructing animated meshes from time-varying point clouds. *Computer Graphics Forum (Proceedings of SGP) 27,* 5, 1469–1476.

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated mesh animation from multi-view silhouettes. *ACM SIGGRAPH*.

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

WEISE, T., LEIBE, B., AND GOOL, L. J. V. 2007. Fast 3d scanning with automatic motion compensation. In *CVPR*.

WEISE, T., LI, H., GOOL, L. V., AND PAULY, M. 2009. Face/off: Live facial puppetry. In *Eighth ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Space-time faces: high resolution capture for modeling and animation. In *ACM SIGGRAPH*. 548–558.