

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 to align all scans to a common pose, which gathers geometry from all scans to reconstruct a full 3D model. 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 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, this 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 can handle difficult topological cases well. We show that we can reconstruct a variety of 3D models completely automatically, without the use of markers, user-placed correspondences, a segmentation, or a template. In addition, our algorithm can reconstruct reasonable piecewise rigid approximations to non-rigid motion sequences as well.

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.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© YYYY ACM 0730-0301/YYYY/09-ARTXXX \$10.00

DOI 10.1145/XXXXXXXX.YYYYYYY

<http://doi.acm.org/10.1145/XXXXXXXX.YYYYYYY>

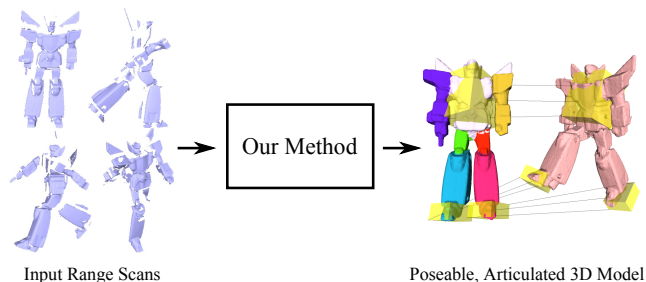


Fig. 1. Our method automatically reconstructs articulated, poseable models from a sequence of dynamic range scans.

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 first main challenge is to resolve the occlusion and missing data that occur due to a limited view of a 3D subject from any single viewpoint. Thus, scans taken from many different viewpoints must be aligned and integrated together in order to reconstruct a complete surface. This relates to the second challenge of tracking the spatially varying movement of the scanned surface, which must be performed accurately to obtain a good alignment. Since each range scan is just a snapshot of surface geometry, no correspondences are tracked between frames. Therefore, correspondences must be estimated in the processing step by directly matching the surface geometry. The third challenge is to improve the usability of the reconstructed model, by not only enabling playback of the original performance, but also to facilitate synthesizing new animations and performances of the subject. The vision is to automatically reconstruct detailed, poseable models that animators can directly plug into existing software tools to create new animations.

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. The uniqueness of our method is that 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, similar to the pairwise registration method of Chang and Zwicker [2009]. However, we overcome the limitations of this previous work with the following main contributions:

- We optimize the registration and articulated model parameters simultaneously over all frames, allowing the algorithm to reduce accumulation of registration error and fit parameters that properly reflect the surface movement observed in all frames.
- We replace the grid-based representation of the weights with a graph-based representation, which allows adaptive sampling and better handling of topologically difficult situations.
- We improve a robust registration technique to efficiently initialize the registration between adjacent frames in the sequence. This completely automates the registration process, without requiring the user to tweak parameters or manually place correspondences.

The main advantages of our method is that it can handle range scans with fast motion and significant occlusion, and that it produces a rigged 3D model. However, our method is mainly applicable to articulated subjects, and may produce a rather crude approximation of the surface geometry and motion for non-rigid cases. 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

In this section, we review related work and point out the main differences of our work from these methods.

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, at 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 a few 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 approach 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 and produce a complete, rigged model without using a template.

Templates are also used for estimating shape using multiview silhouette/video data or sparse marker data. These techniques significantly differ from our approach mainly because of the difference in input data that is processed to reconstruct the geometry. While we address shape capture from high-resolution 3D point clouds, these approaches fit a template to multiview silhouettes [de Aguiar et al. 2008; Vlastic et al. 2008; Gall et al. 2009] or a set of tracked markers [Park and Hodgins 2006; 2008]. Also, while the surface detail in our approach comes directly from the range scans, most surface detail in these approaches come directly from the template, or added as a post-process using dense normal maps computed by shape from shading [Ahmed et al. 2008].

Templateless Registration and 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. However, this technique requires the surface to be sampled densely in both space and time, which is an assumption that our method does not require. Süßmuth et al. [2008] and Sharf et al. [2008] explicitly model and reconstruct the 4D space-time surface using an implicit surface representation. However, they also require a dense sampling in space and time. 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 employs a hierarchical approach to align a sequence of range scans. It gradually builds the template shape as it solves for the alignment between the scanned frames, while the motion of the surface is represented using an adaptive, hierarchical displacement field. In our method, we simultaneously align all frames using an explicit piecewise rigid deformation model. As we will show in the results section, our deformation representation is more compact and accurate for representing articulated motion. 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.

The closest 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 6.5). 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 demonstrated in the results section, we significantly improve and extend this original approach to handle multiple frames and difficult topological cases more effectively.

Unsupervised Pairwise Registration. While our work is designed for handling multiple range scans, several methods for unsupervised pairwise registration are related as well. 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, the original global optimization is too slow to apply for an entire sequence of range scans. Therefore, we improve the performance of this method by subsampling the geometry. The use of a graph to represent the deformation model in our work is related to the approach by Li et al. [2008] and [Sumner et al. 2007]. However, we define and 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. This approach is problematic when a large amount of surface data is missing, which is not the case for our method.

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 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

We first outline the basic structure of our articulated reconstruction method. The input to our algorithm is a sequence of range scans, where each frame of the sequence is denoted F_0, \dots, F_n . The sequence is expected to be in temporal order so that there is sufficient overlap between frames to align the scans. In addition, the user specifies the maximum number of rigid parts B that the algorithm should use to approximate the deformation of the surface.

The output of our algorithm is a set of sample points S (representing the surface of the reconstructed subject), a set of rigid transformations \mathcal{T} (B transformations for each frame F_i), and a set of weight vectors \mathcal{W} (one vector for each point in S). Each weight vector has B real components, one for each rigid part, where each component indicates the strength (or influence) of the corresponding part.

Our strategy is to solve for the transformations and weights (on each frame) that align all frames to a common pose. The optimization is performed in alternating fashion, like the pairwise registration by Chang and Zwicker [2009]. Also, the transformations and weights are optimized only using a subset of the points on each frame (i.e. the sample set S) to improve the performance of the optimization. Unlike the original pairwise technique which used a regular grid to define the weights, we define the weights directly on each sample in S . This simplifies the algorithm by removing the overhead of translating between the locations of the surface samples and the grid cells. The regular grid connectivity, which was needed for specifying smoothness constraints between weights, is now replaced by a Euclidean k -nearest neighbor graph on the sample positions. This provides a more flexible structure to avoid sampling issues and topological problems.

Algorithm 1: ARTICULATED GLOBAL REGISTRATION

Data: A sequence of range scans, denoted (F_0, \dots, F_n)

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 4.1);
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  $F_{\text{last}} \neq F_n$  do
6     Let  $F_{\text{new}}$  be the next frame after  $F_{\text{last}}$ ;
7     Load and apply the initial registration result for
        $F_{\text{last}} \rightarrow F_{\text{new}}$  (Section 4.1);
8     Check if any parts are occluded in  $F_{\text{new}}$  (Section 4.5);
9     OPTIMIZE  $\mathcal{T}, \mathcal{W}(S, E, \mathcal{T}, \mathcal{W}, F_0, \dots, F_{\text{new}})$ 
       (Algorithm 2);
10    Resample  $S$  from all frames  $(F_0, \dots, F_{\text{new}})$  and
       construct ASG (Section 4.4);
11     $F_{\text{last}} \leftarrow F_{\text{new}}$ ;
12  return  $S, \mathcal{W}, \mathcal{T}$ ;
13 end
```

During the global registration, 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, a single rigid transformation directly transforms each sample point $\mathbf{x} \in S$, selected according to which weight component is turned “on.” Later in a post-processing step, we can determine smooth weights that blend the transformations using an interpolation scheme, e.g. linear blend skinning (LBS) or dual-quaternion linear blending (DLB). So unless otherwise noted, we will use the terms “weight,” “binary weight,” and “label” interchangeably.

The basic structure of our method is shown in Algorithm 1. The first step is to solve for an initial registration for each pair of adjacent frames $(F_0, F_1), (F_1, F_2), \dots, (F_{n-1}, F_n)$ (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. In Section 4.1, we develop an improved version of this method to speed up the optimization. The final output of this step is an initial registration between adjacent frames, given as a set of rigid transformations that are assigned to each point on the two frames.

The second step is to refine this initial registration and produce a global registration of all frames to a common pose. The basic 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 (line 10).

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. We will discuss this more in Section 4.5.

Now we describe each component of our algorithm in detail. After performing the registration on the entire sequence, we can resample the surface densely, reconstruct a mesh representing the completed surface, and fit smooth skinning weights to obtain a final polished reconstruction.

4. ALGORITHM DETAILS

4.1 Initialization

The algorithm by Chang and Zwicker [2008] finds a registration between a pair of meshes (source mesh and target mesh). It consists of two steps: (1) sampling rigid transformations that describe the movement between the source and target, and (2) optimizing the assignment of these transformations on each point of the source and target, so that it produces the best alignment of the two meshes. This optimization is a discrete labeling problem that is solved efficiently using graph cuts [Boykov et al. 2001]. 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. Therefore, we simplify the method by solving the discrete optimization only on a small subset of the points in each frame.

First, the sampling of the transformations is done exactly the same as the original method. Next, we modify the discrete optimization by restricting the optimization problem on a small number of points (e.g. 500-1000), uniformly sampled on the source and target meshes. (Note: these sampled points are completely separate from the sample set S used in the global registration.) 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 sampled points, where k is typically 15. Finally, the data, smoothness, and symmetric consistency terms in the optimization are the same as the original method, but their evaluation is simply restricted to the sampled points and the nearest neighbor graph. One exception is the data term, where we compute the distance to the closest point among *all* points in the other frame (as opposed to the closest *sampled* point in the other frame), in order to get a more accurate measurement. Once we solve the labeling problem on the sampled points, we can propagate these labels to all remaining points of each frame by finding the closest sample and assigning its label.

A comparison with the original method is shown in Figure 2. Although we obtain a good alignment in both cases, the improved method achieves a significant speedup. Also, the use of the graph improves the connectivity between parts that may be disconnected in the original mesh. In some cases, we even observed that this produced a correct registration where the original method did not.

After initializing the registration between adjacent frames, the solved transformations and labels are used as initial values for the global registration discussed in the next section. Since the global registration optimizes the alignment using a set of samples S , we need to propagate the information to S . We discuss these details further when we describe the sample set later in Section 4.4.

4.2 Global Registration

Once a frame is initialized, it is introduced into the global registration step. This step optimizes for the best transformations and weights that simultaneously align all initialized frames. 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

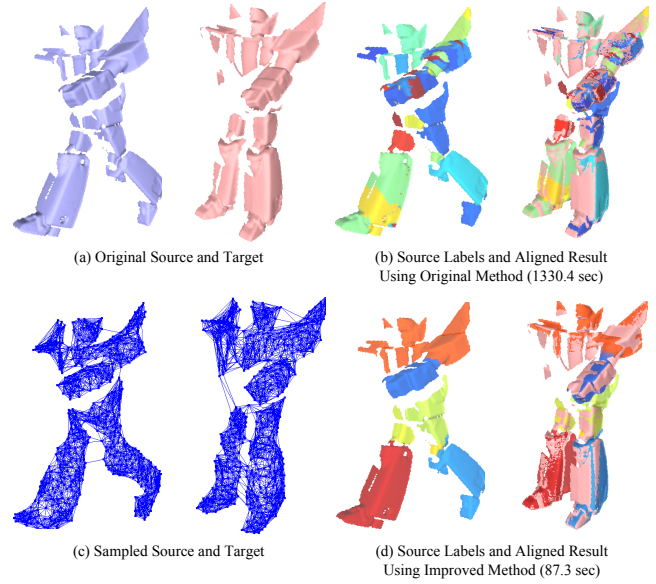


Fig. 2. Comparison between the original and improved initial registration methods. The top row (a,b) shows the result with the original method, while the bottom row (c,d) shows the result using a graph of 500 samples on each frame. With the same parameters, the improved method produces a similar alignment in a fraction of the original time.

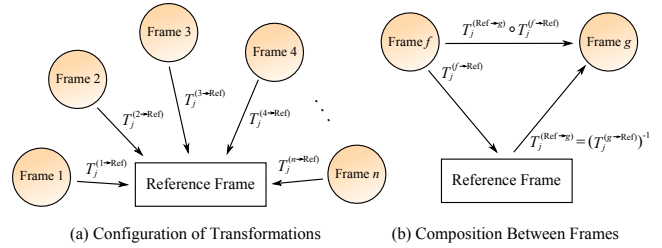


Fig. 3. Organizing the transformations for simultaneous registration. (a) We solve for the set of transformations that align each input frame to the reference frame F_0 . (b) We can transform between any pair of frames f and g by first transforming from f to the reference and applying the inverse transformation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

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

$$\operatorname{argmin}_{\mathcal{T}, \mathcal{W}} \alpha \mathcal{E}_{\text{fit}}(\mathcal{T}, \mathcal{W}) + \beta \mathcal{E}_{\text{joint}}(\mathcal{T}) + \gamma \mathcal{E}_{\text{weight}}(\mathcal{W}). \quad (1)$$

Next, we describe each term and our optimization procedure in more detail.

4.2.1 Organization of the Transformations. To organize the transformations for each frame concisely, we designate one of the frames as a *reference frame* and define rigid transformations relative to this reference¹. This definition makes it easy to specify and solve for the alignment for multiple frames. We use the notation $T_j^{(f \rightarrow \text{Ref})}$ to denote the j th transformation for frame F_f , which

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

transforms in the direction *from* frame f to the reference frame (Figure 3a).

Each transformation $T_j^{(f \rightarrow \text{Ref})}$ is composed of a rotation $R \in SO(3)$ and translation $\vec{t} \in \mathbb{R}^3$. To apply the transformation to a point $\mathbf{x} \in \mathbb{R}^3$, we use the notation $T_j^{(f \rightarrow \text{Ref})}(\mathbf{x}) = R\mathbf{x} + \vec{t}$. We can also transform between any two frames f to g by transforming first to the reference frame and then applying the inverse transformation to the desired frame (Figure 3b). This is expressed using an inverse operator $(\cdot)^{-1}$ and a composition operator \circ , which corresponds to the formula

$$\begin{aligned} T_j^{(f \rightarrow g)}(\mathbf{x}) &= \left(T_j^{(g \rightarrow \text{Ref})} \circ T_j^{(f \rightarrow \text{Ref})} \right) (\mathbf{x}) \\ &= R_j^{(g \rightarrow \text{Ref})} \top \left[\left(R_j^{(f \rightarrow \text{Ref})} \mathbf{x} + \vec{t}_j^{(f \rightarrow \text{Ref})} \right) - \vec{t}_j^{(g \rightarrow \text{Ref})} \right]. \quad (2) \end{aligned}$$

Therefore, once we know the transformations that relate the pose of each frame to the reference, we can transform any point (that has a weight) from any source frame to any target frame.

4.2.2 Sample Set S and All-Samples Graph (ASG). The set of samples S plays a key role in the global registration step. It serves as a coarse representation of the reconstructed 3D model, and it is used to optimize the alignment between the frames. 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 .) Associated with each sample \mathbf{x} is a vector of weights $\mathbf{w}_\mathbf{x} = [w_{\mathbf{x}1}, w_{\mathbf{x}2}, \dots, w_{\mathbf{x}B}]$. During our optimization we will solve for binary weights (labels), thus 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 positive component.

In addition to the sample set S , we construct a graph structure over S , which we call the *all-samples graph (ASG)*. The connectivity of this graph serve as smoothness constraints that help the optimization form large, contiguous parts. The graph is constructed by forming the k -nearest neighbor graph on the sample set S . More details on sampling S and constructing the ASG is discussed in Section 4.4.

4.2.3 Fitting Objective \mathcal{E}_{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 , the weight $\mathbf{w}_\mathbf{x}$ assigned to \mathbf{x} and the transformations \mathcal{T} tell us the transformed location of \mathbf{x} on all other frames. Therefore, in this term, we transform \mathbf{x} to all other frames and measure how close it is to the scanned data of these frames.

To measure the proximity of a transformed point $\mathbf{x}' = T_j^{(f \rightarrow g)}(\mathbf{x})$ to frame F_g , we take the distance to the closest point $\mathbf{y}_j^{(g)}$ in F_g . There are four important details to add to this basic strategy. (1) Notice that the closest point will change depending on which of the B transformations we use to transform \mathbf{x} to frame g . Therefore, we keep a separate closest point $\mathbf{y}_j^{(g)}$ for each j . (2) It may be the case that \mathbf{x}' may not have a corresponding point in F_g due to missing data. To handle this case, for each sample point \mathbf{x}' (i.e. transformed to frame g), we mark the corresponding target point $\mathbf{y}_j^{(g)}$ as *invalid* if (a) the distance between these points exceeds a threshold τ_d , (b) the angle between the normals exceeds a threshold τ_n , or (c) the target point lies on the boundary and the distance exceeds a smaller threshold τ_b [Pekelný and Gotsman 2008]. (3) If the target point is invalid for the currently assigned label j^* (which is the best approximation so far), then mostly likely the target point is actually missing in the frame. Therefore, we invalidate

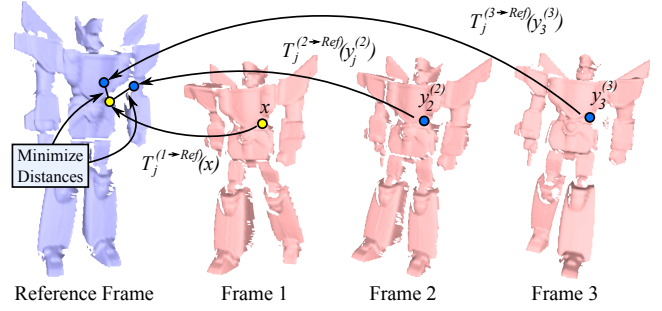


Fig. 4. To measure alignment, we compute distances between sample points \mathbf{x} (yellow) and target points $\mathbf{y}_j^{(g)}$ (blue) transformed to the reference frame F_{ref} . We add up these distances to measure the alignment of all frames in the sequence. We optimize for the transformations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

all target positions $\mathbf{y}_j^{(g)}$ for all j if $\mathbf{y}_{j^*}^{(g)}$ is invalid. (4) Finally, when we construct S , we only add samples from each frame that is unobserved in all previous frames. Therefore, for frames g previous to f (i.e. $g \leq f$), we do not search for a target point in that frame, since we expect it to be missing.

Given these corresponding points, we can precisely quantify the alignment distance between the sample points of all frames. Mathematically, we measure the alignment distance using the following expression:

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

Here, we have computed the distance $d(\cdot, \cdot)$ between \mathbf{x} and valid corresponding target points $\mathbf{y}_{j^*}^{(g)}$ on 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.

An alternative to the above formula is to compute the distance in frame F_g , as $d(T_{j^*}^{(f \rightarrow g)}(\mathbf{x}), \mathbf{y}_{j^*}^{(g)})$. However, this involves a composition of the transformations, which makes it more difficult to solve in the optimization step. This is because, since we substitute a separate linearization for each transformation, a composition of transformations would result in a multiplication of two linearized transformations, resulting in a less accurate approximation.

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

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

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

4.2.4 Joint Objective $\mathcal{E}_{\text{joint}}$. The joint term constrains neighboring transformations to agree on a common joint location. This term ensures that the parts stay connected to each other and not drift apart. Our method supports 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 always agree on 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 $t \in \mathbb{R}$. In

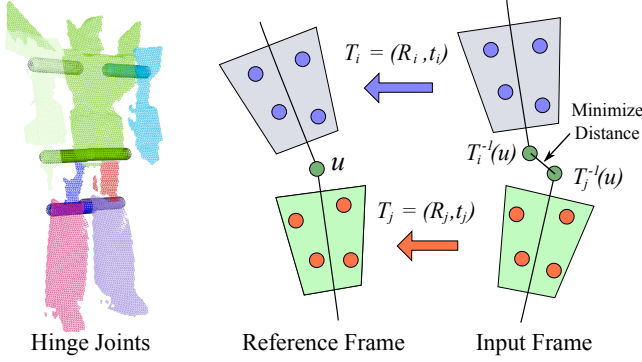


Fig. 5. Estimating and constraining joints in our optimization. (Left) we show hinge joints that are automatically estimated. (Middle & Right) $\mathcal{E}_{\text{joint}}$ constrains the transformed locations of \mathbf{u} to agree on the same point by minimizing the distance between the transformed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

contrast to the hinge joint, a ball joint says that the transformations agree only on a single point $\mathbf{u} \in \mathbb{R}^3$. We can also express a ball joint in the same form as the hinge joint, 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 can constraint the joints using the term $\mathcal{E}_{\text{joint}}$:

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

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, $\vec{\mathbf{v}}_{ij} = \vec{\mathbf{0}}$, so this term constrains only one point \mathbf{u}_{ij} . For a hinge joint, it constraints a set of points along the hinge axis. The inverse of the transformations are used in this term because the joint locations are defined on the reference frame.

4.2.5 Detecting Joint Locations. Since we have multiple frames, we estimate joint locations based on the solved transformations for each frame. But first, we first need to know which pairs of transformations are likely to share a joint in between. To determine this, we examine the ASG to see which pairs of labels are neighboring on this graph.

Consider the set of the edges E' in the ASG that have different labels assigned to the end points. If we have a large number of edges with labels i and j , this would indicate that transformations i and j are likely to share a joint. On the contrary, a small number of edges (or none) would indicate that the transformations are not related. To determine which label pairs are significant, for each pair i, j we count the number of edges $e \in E'$ whose labels are i, j . Also, for each label i , we count how many edges $e \in E'$ are incident to i (i.e.

at least one endpoint is labeled i). The following ratios

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

give a measure of how dominant i, j are for labels i and j , respectively. If either of these ratios exceeds a threshold (set to 15%), then we take the pair i, j as a candidate for sharing a joint.

The edges also give a rough estimate of where we would expect the joint location. For each i, j candidate, we compute the average of all the endpoint locations (on the reference frame) of edges with label i, j . This position, which we denote as \mathbf{u}_{est} , gives us a guess of where the joint location is likely to be.

Once we have a set of candidate label pairs i, j and estimated joint locations \mathbf{u}_{est} , we solve for the true joint locations \mathbf{u} on the reference frame using the transformations estimated so far at all frames. This is done by performing a least-squares minimization for each pair (i, j) :

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

For hinge joints, the solution will be a set of points (on the reference frame) lying on the hinge axis. When we solve the above least-squares minimization using the 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 (set to 0.1 in our implementation). 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}_{est} and normalize $\vec{\mathbf{v}} = \vec{\mathbf{v}}' / \|\vec{\mathbf{v}}'\|$.

If the joint is not a hinge, it will be a ball joint where 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 [Pekelný and Gotsman 2008]. This additional term helps to pull the location closer to \mathbf{u}_{est} in case the joint is close to being a hinge and admits multiple solutions.

4.2.6 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 (interpreted as “labels”) to the sample points $\mathbf{x} \in S$. The goal of the weight objective term is to ensure that neighboring samples have a similar label, so that labels form large, contiguous regions on the ASG. Therefore, for $\mathcal{E}_{\text{weight}}$ we use a simple constant penalty when two neighboring samples in S are assigned different labels. We express this using the formula

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

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 the Potts model, which is a discontinuity-preserving interaction term [Boykov et al. 2001].

4.3 Optimization

To solve 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.

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

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

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

Result: Optimized transformations and labels \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  $\mathcal{T}$ )
5       Re-estimate joint locations and types;
6       while Not converged do
7         Update the closest points  $\mathbf{y}_{j^*}^{(g)}$  for all  $\mathbf{x} \in S$ 
           and frames  $F_g$ ;
8         Construct the sparse matrices for  $\mathcal{E}_{\text{fit}}$  and  $\mathcal{E}_{\text{joint}}$ ;
9         Solve linear system and update
           transformations;
10        Check convergence criteria;
11      end
12      Handle reappearing parts in  $F_{\text{new}}$  by aligning occluded
           label regions with unmatched surface points
           (Section 4.5);
13      Check convergence criteria;
14      if converged then break;
15      begin (Phase 2: Solve for the labels  $\mathcal{W}$ )
16        Update the closest points  $\mathbf{y}_j^{(g)}$  for all  $\mathbf{x} \in S$ ,
           frames  $F_g$ , and  $j \in [1..B]$ ;
17        Precompute  $\mathcal{E}_{\text{fit}}$  for each  $\mathbf{x} \in S$  and  $j \in [1..B]$ ;
18        Create a graph for  $\mathcal{E}_{\text{weight}}$  using the edges  $E$  of the
           ASG;
19        Solve discrete labeling on this graph using
            $\alpha$ -expansion;
20        Discard labeled regions that are too small;
21        Reuse unassigned labels by splitting regions with
           highest  $\mathcal{E}_{\text{fit}}$  error;
22        Update the labels for each  $\mathbf{x} \in S$ ;
23      end
24 end

```

In our experiments, we observed that the transformations for a frame does not change much after the first global registration pass when the frame is first introduced. Therefore, we provide an option in the global registration to solve for the transformations only on the newest k frames that have been introduced to the global registration. We can think of this as solving for the transformations on a sliding window of k frames. 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 the step for optimizing the transformations. The weights are still optimized globally over all frames.

Also, during the global optimization we try to detect if previously occluded parts have reappeared in the new frame (line 12). We discuss how we handle these cases in Section 4.5.

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 1, while keeping the weights fixed. Since the target positions $\mathbf{y}_j^{(g)}$ corresponding to each sample $\mathbf{x} \in S$ changes depending on the transformations, we use an iterative approach in the spirit of ICP [Besl and McKay 1992]

and alternate between updating the transformations and the corresponding target positions until convergence. Since the weights are fixed, only the non-zero weight components will contribute to the \mathcal{E}_{fit} in this step. Therefore, we only need to update the target positions $\mathbf{y}_{j^*}^{(g)}$ for the currently assigned label j^* at each sample \mathbf{x} .

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. Specifically, each rigid transformation $g = (R, \vec{t}) \in SE(3)$ is linearized about the current estimated transformation g_k (k here is the index of the iteration in the Gauss-Newton algorithm) using the formula $(I + \hat{\xi})g_k$, where $\hat{\xi} \in se(3)$ is a *twist* matrix [Ma et al. 2003]. For linearizing the inverse of a rigid transformation, we substitute $g_k^{-1}(I - \hat{\xi})$. Also, when linearizing the point-to-plane term, we simplify it further by applying only g_k to the normal vector \vec{n}_y , instead of applying the additional rotational part of $\hat{\xi}$ to \vec{n}_y . These manipulations transform the objective into a sparse linear system of equations, which we solve using TAUCS [Toledo 2003]. To complete the Gauss-Newton iteration, we apply the exponential map to convert the linearized solution parameters back to rigid transformations, and we multiply these to the current estimates g_k to update the solution.

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 can significantly reduce the time needed to perform this step.

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 a label j^* for each sample point that minimize the total error. We solve this discrete labeling 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.

Unlike the transformations, we solve for the weights always over all frames. This means that we need to compute the \mathcal{E}_{fit} term over all frames and all labels. Since the transformations are kept fixed in this stage, we can precompute \mathcal{E}_{fit} to save computation time during the optimization. Here, we compute the distance using Equation 4, for all samples \mathbf{x} , all labels j , and all frames g . We then store the distance values in a hash table for quick look-up during the optimization.

After the optimization, it may be the case that some labels are only assigned to a few sample points, or other labels may not be assigned at all. In the first case, we can discard these labels to obtain a solution that is simpler and not substantially different from the original solution. In our implementation, we discard a label if the percentage of samples assigned to that label is less than some small threshold (set to 1% in our implementation). In the second case, we reuse the unassigned labels by splitting the regions with the highest registration error in half, introducing the new label in one of these halves [Chang and Zwicker 2009]. This process is continued until the highest registration error is below a threshold (typically 0.1s), or until there are no remaining labels.

Checking for Convergence. We have placed 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

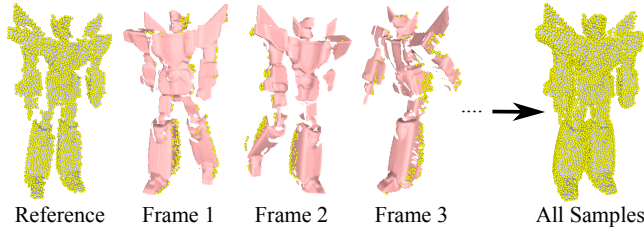


Fig. 6. We use sample points on all input frames to measure the global alignment. For each frame, we only keep the samples that are from new geometry that has not been observed in any previous frames.

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 remaining details of the algorithm involving the sample set S and handling occlusion.

4.4 Sample Set and All-Samples Graph (ASG)

Our purpose in using the sample set is to sparsely represent the entire observed surface, while minimizing the redundancy of geometric information. This will maximize the performance of the optimization, while still giving an accurate registration of the surfaces. Our approach is to select a well-distributed set of points in each registered frame and merge all of the sets together. During the merging step, we minimize redundancy in S by removing overlapping points. This sampling and merging process is performed every time a new frame is added to the global registration, so that any changes in the registration or newly observed parts of the surface are incorporated into S . Once S has been sampled, we assign weights to each sample point in S based on the weights of the previously used sample set. Finally, we construct the ASG (a k -nearest neighbor graph on S), taking care to remove undesired connections between separate parts. We discuss these steps in detail in the sections below.

Creating the Sample Set and ASG. We first uniformly sample a set of points U_f in each frame f using the best-candidate technique [Mitchell 1991]. We pick a fixed fraction of the points, where this fraction is a user-specified parameter. The user may adjust it lower to compute the registration faster, or higher to obtain a more precise alignment.

At the beginning of the global registration, S is simply equal to U_0 . However, when a new frame is added, we resample S to obtain a new set of samples, S' . This resampling is performed by merging all U_f using a sequential strategy. Starting with $S' = U_0$, we iterate through each successive U_f and determine (1) which points are not overlapping with the currently selected samples S' , and (2) we determine the weights of each $\mathbf{x} \in U_f$ using the weights of the previous sample set S . Then, we add all non-overlapping samples that have a valid weight to S' , and repeat this until we have processed all frames.

Once the new sample set S' has been created, we construct the ASG on S' by transforming all samples to the reference frame and computing the k -nearest neighbor graph of the samples, using $k = 15$. 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 too much. 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 are connected by a joint. Finally, S' and its graph replace the old sample set S and its graph to begin the registration process for the next frame in the sequence.

Finding Overlap and Extrapolating Weights. For resampling S to create S' , we apply the strategies outlined by Pekelny and Gotsman [2008] to remove the redundant overlapping samples and extrapolate the weights. To detect overlapping samples, we first transform S' to frame F_f . Then, for each $\mathbf{x} \in U_f$, we find the closest point \mathbf{y} in S' , and we compute the point-to-point distance $d_{pt} = \|\mathbf{x} - \mathbf{y}\|$ and the *point-on-plane* distance $d_{onpl} = \sqrt{\|\mathbf{x} - \mathbf{y}\|^2 - ((\mathbf{x} - \mathbf{y}) \cdot \mathbf{n}_y)^2}$. Finally, \mathbf{x} is determined to be overlapping with S' if these distances are smaller than a user-given threshold τ_s . Here, we threshold using d_{pt} if the surface normals differ by more than 90 degrees; otherwise we threshold using d_{onpl} .

To extrapolate the weights from the original set S to the new samples U_f , we first partition all the samples $\mathbf{x} \in S$ into separate sets V_j for each label j , and transform these sets into frame F_f . To determine the label for a new sample point $\mathbf{x} \in U_f$, we first compute the distance from \mathbf{x} to the closest point \mathbf{y}_j of each set V_j , using Equation 4. These distances (there will be one for each label j) are then inverted and normalized to give a score for that label, and the label with the maximum score is taken as the label for the new point \mathbf{x} . However, if this label is marked as occluded for this frame, or 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 label as invalid.

Propagating the Initial Registration. The result of the initial registration between adjacent frames F_i and F_{i+1} from Section 4.1 is a set of transformations and their assignment to each point in F_i and F_{i+1} . Since the global registration works only with weights defined on a separate sample set S , we need to propagate this initial registration information to S .

At the very beginning of the global registration where we process the first pair of frames F_0 and F_1 , S is simply equal to U_0 . Therefore, we can directly copy over the labels and the transformations from the initial registration. Since the global registration works with at most B labels, we take care to only use the top B transformations (in the initial registration between F_0 and F_1) that have the largest regions. When a transformation for a sample point $\mathbf{x} \in U_0$ is invalid (not in the top B), the transformation of the closest point $\mathbf{y} \in F_0$ that has a valid transformation is used.

In the case where we want to initialize adjacent frames F_i and F_{i+1} where $i > 0$, the situation is different. First, the points in S are not necessarily from F_i because we remove overlapping samples when resampling S . Also, weights are already defined on S as well, so the problem is to determine what should be the initial transformation for each label j , where the transformations are extracted from the initial registration between F_i and F_{i+1} .

Our strategy to handle this case is to (1) transform S to frame F_i , and (2) for each sample $\mathbf{x} \in S$ with label j , we find its closest corresponding point \mathbf{y} in F_i (using the strategy outlined in Sec-

tion 4.2.3) and find the transformation that was assigned to \mathbf{y} in the initial registration. Now, when we make a list of these transformations for all sample points having label j , usually there is more than one unique transformation. However, we want to determine a single rigid transformation for each labeled region j . Therefore, we blend these transformations using DLB [Kavan et al. 2008], where each transformation is weighted according to the fraction of samples among label j that has the transformation.

This strategy produces a slightly different result from the original registration, but it was sufficiently close in practice. Also, since the transformations are specified relative to a reference frame in the global registration (see Section 4.2.1), we take care to express the transformation properly by inverting and multiplying it with the transformations of frame F_i .

4.5 Treating Occlusion

When a part of the surface is partially occluded 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 that particular frame. In our algorithm, we automatically detect if this happens and exclude these transformations from the optimization. We perform this occlusion check right after we initialize a new frame (see Algorithm 1).

To decide if a part is occluded, we update the closest points for this frame after the initialization. Then, we count the number of target positions $\mathbf{y}_j^{(g)}$ for each label. 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 label as occluded for this frame.

To handle occluded parts when optimizing for the transformations, we remove the variables and constraints involving these occluded labels from the objective term. Instead, we guess a reasonable transformation based on the joint constraints with neighboring transformations. If there are no neighbors, we use the value from the last frame; if there is exactly one, we copy its value; and if there are two or more, we solve for the transformation that best fits all joint constraints [Pekelny and Gotsman 2008].

For optimizing the weights, the situation is more involved. Since the weights are optimized globally over all frames, we cannot remove a label entirely from the optimization just because it was occluded in some frames. Therefore, we must decide on an appropriate \mathcal{E}_{fit} value when it involves a frame with an occluded label.

There are several possibilities to handle this case. We could use the guessed transformation to compute a value of \mathcal{E}_{fit} , but this approximation could be wildly inaccurate in some cases. Alternatively, we could use a constant error value, but deciding on an appropriate value is problematic. A zero error or very low error could cause the occluded label to be assigned to completely unrelated locations, because it produces a lower error than the actual “correct” label. On the other hand, a high value may discourage from assigning this label to actual “correct” locations, e.g. where the surface of the occluded part is partially visible in the frame. In our experience, the best solution is to use the error value of the current label assigned to the sample. If the current label is occluded as well, then we use the minimum error among all unoccluded labels at that sample. This prevents the occluded label from replacing “correct” labels, since this value plus the extra penalty of violating the smoothness constraint is greater than the error of assigning the original label. At the same time, this does not penalize cases where the occluded label should be assigned to the sample. This heuristic worked well for most cases in our experiments, except for a handful

of instances where the occluded label 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 4.3), 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 (Section 4.2.3),
- and we increase the weight of the $\mathcal{E}_{\text{joint}}$ (β in Equation 1) to be very high.

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.

5. POST-PROCESSING

After we have aligned all frames, and we can reconstruct a triangle mesh of the entire model. To do this, we just resample the set S with a small sample distance τ_s , where we use all of the points in each frame to resample S (instead of taking the uniform subsamples U_f as candidates). This results in a dense sample set S , which we use to reconstruct a surface mesh using the streaming wavelet surface reconstruction algorithm by Manson et al. [2008] (kindly provided by the authors). Since our algorithm produces binary weights, there may be deformation artifacts at boundaries between parts. To reduce these artifacts, we can solve or interpolate smooth skinning weights as a post-processing step. However, we do not apply this smoothing step in our results, so that we can give a clearer picture of the reconstruction quality using our binary weight pipeline.

6. EXPERIMENTAL RESULTS

6.1 Reconstruction

We implemented our algorithm in C++ and tested it with several real-world and synthetic datasets exhibiting articulated motion. 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 reconstruction results using our algorithm 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



Fig. 7. Reconstruction results for the robot dataset.

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 4.4). The fourth row shows the reconstructed mesh using the algorithm by Manson 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

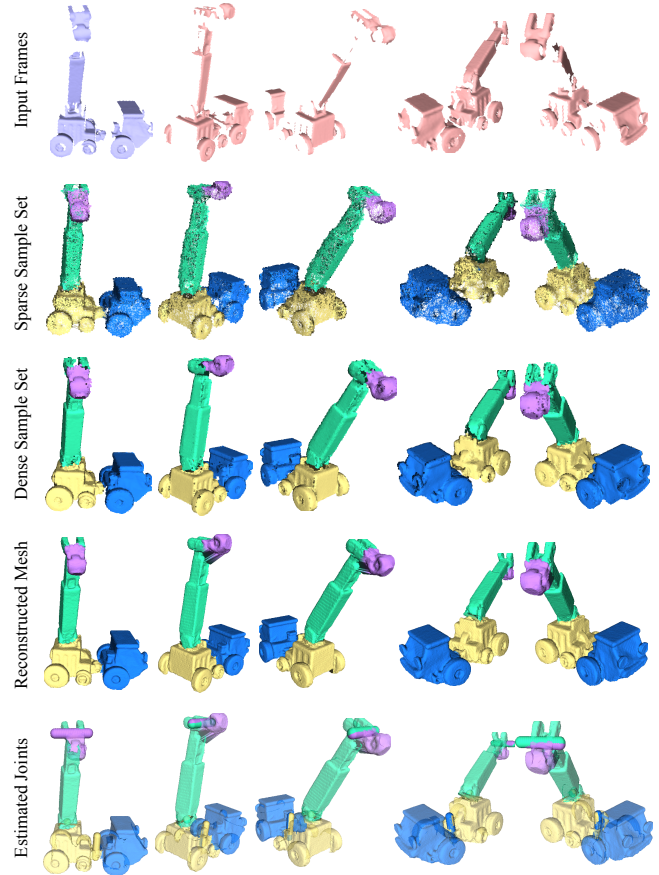


Fig. 8. Reconstruction results for the car dataset.

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 ~ 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,

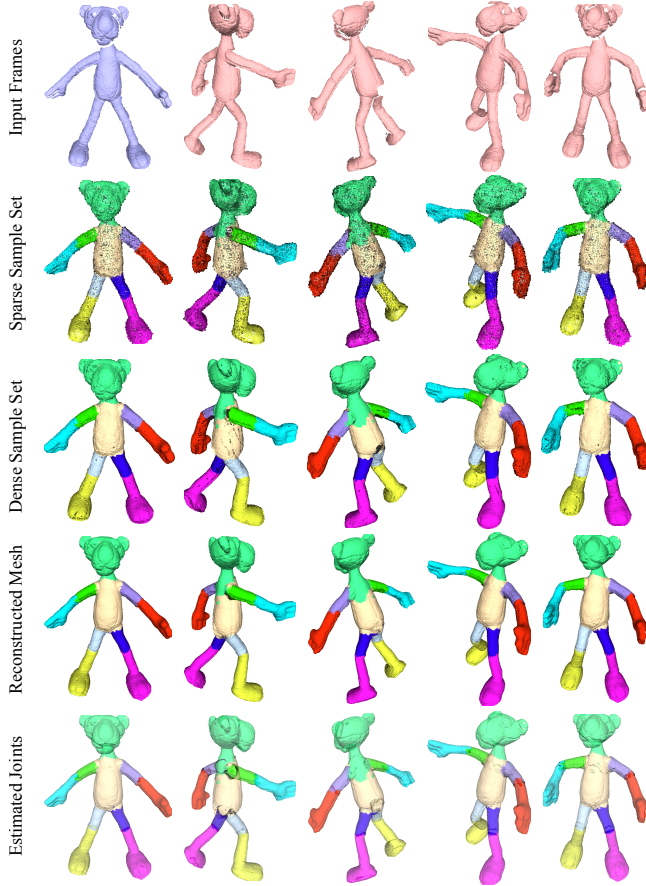


Fig. 9. Reconstruction results for the first Pink Panther dataset (small-motion case).

and this resulted in some “floating parts.” Nevertheless, the reconstructed mesh and segmentation nicely captures the overall shape and motion of the subject.

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.

6.2 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.

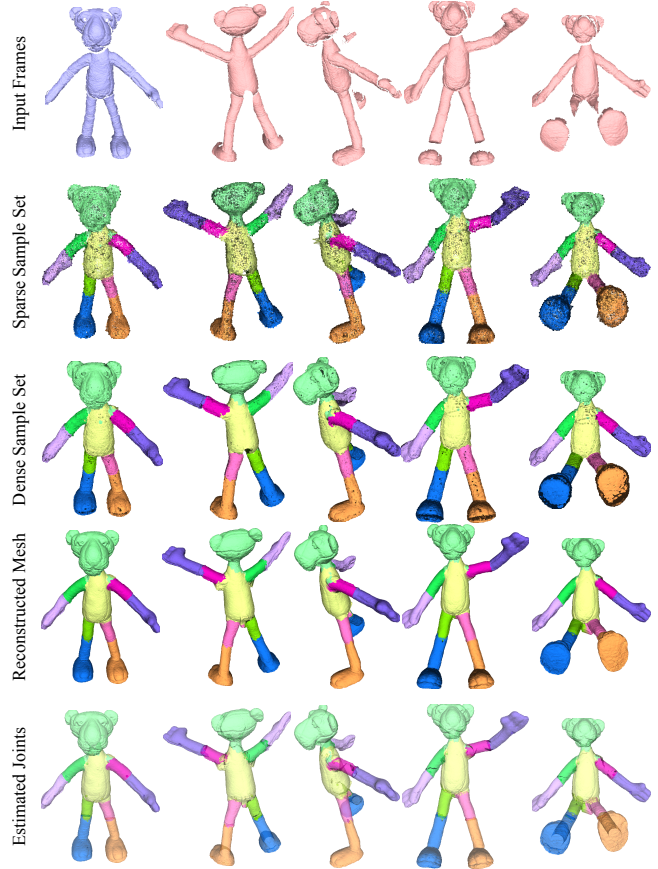


Fig. 10. Reconstruction results for the second Pink Panther dataset (large-motion case).

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 1, 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 4.4), 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 4.2.3), 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.

6.3 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 to-

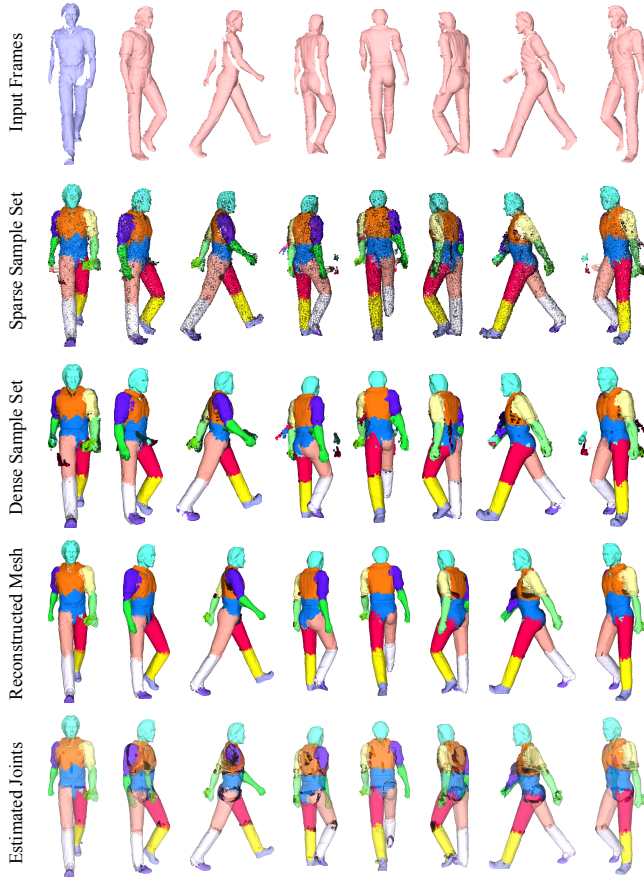


Fig. 11. Reconstruction results for the synthetic Walking Man dataset taken using a single virtual camera.

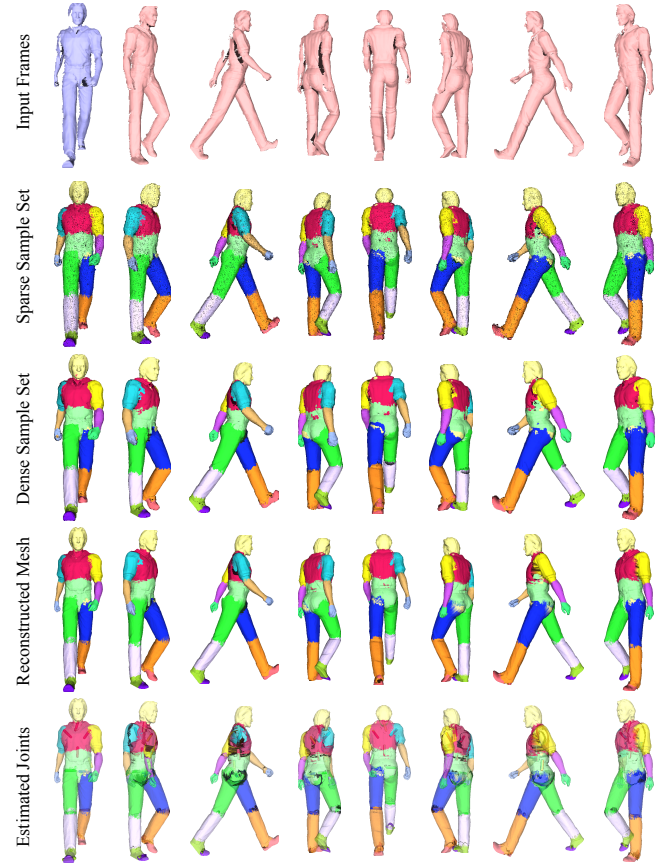


Fig. 12. Reconstruction results for the synthetic Walking Man dataset taken using two virtual cameras.

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 → 1	5 → 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

tal 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.

6.4 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

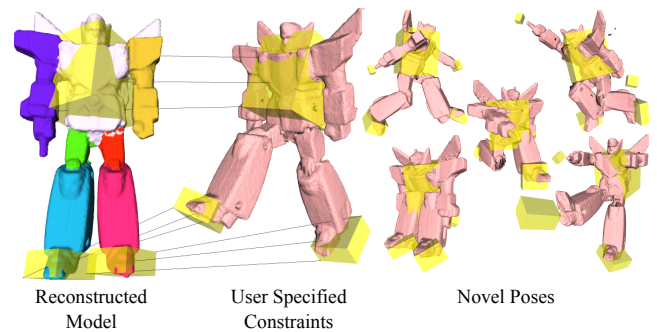


Fig. 13. Reposing the reconstructed robot. By using the solved weights and the hinge joints, our optimization can satisfy point constraints given by the user.

use the transformation optimization (Section 4.3) 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.

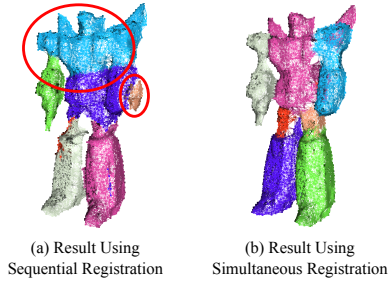


Fig. 14. Comparing sequential and simultaneous registration. (a) As indicated by the large red circle on the upper body area, the sequential strategy gives an unreliable estimate of the articulated structure, because it only uses the movement observed in one frame. This leads to an imprecise registration, 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.

6.5 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 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.

6.6 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

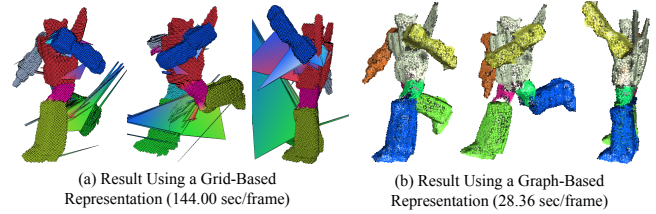


Fig. 15. Comparing grid-based and graph-based weight representations. These images show the represented weight function, deformed into different poses according to the optimized transformations and weights. Notice the deformation artifacts with the grid-based representation, which is absent in the graph-based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

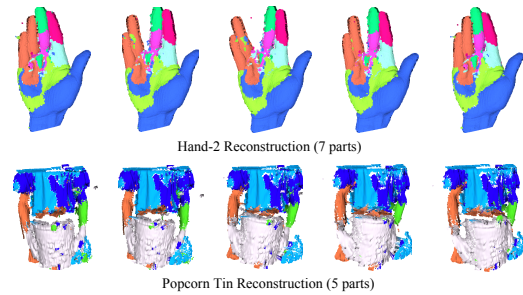


Fig. 16. Articulated registration on the hand-2 and popcorn tin datasets used by Wand et al. [2009]. Our algorithm is able to produce coarse approximations of the non-rigid motion exhibited in these datasets.

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.

6.7 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.

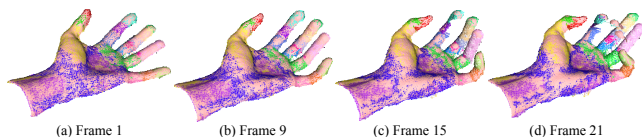


Fig. 17. Registration for a grasping hand sequence [Weise et al. 2007], where the hand starts from an open pose and gradually closes to a grasping pose. Shown are the input data (displayed as a red color mesh) and the sparse ASG. Our algorithm tracks the hand well in the first part of the animation, 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).

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.

7. 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 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.

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.
- MA, Y., SOATTO, S., KOSECKA, J., AND SASTRY, S. S. 2003. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag.
- 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.
- TOLEDO, S. 2003. Taucs: A library of sparse linear solvers, version 2.2. <http://www.tau.ac.il/~stoledo/taucs>.
- 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.