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 skinning weights, 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

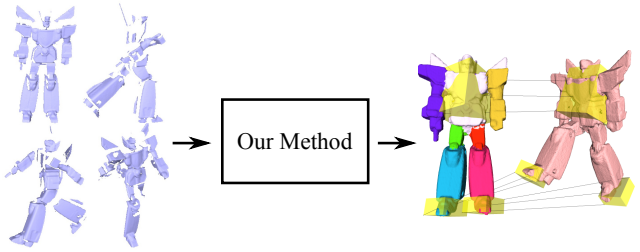


Fig. 1. Our method can automatically reconstruct articulated, poseable models from a sequence of single-view dynamic range scans.

1. INTRODUCTION

While 3D scanning has traditionally focused on acquiring static, rigid objects, recent advances in real-time 3D scanning have 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.

Our research vision is to automatically reconstruct detailed, poseable models that animators can directly plug into existing software tools and use to create new animations. The main challenges to achieve this are to resolve the occlusion and missing data that occur in range scans, and to recover the structure of the deformation exhibited by the scanned object. Missing data is due to a limited view of a 3D subject from any single viewpoint at any point in time. Therefore, we must align and integrate scans taken over time and from different viewpoints to reconstruct a complete surface. Since the subject moves from frame to frame, we must also track the spatially varying surface motion to align the data over time. We recover the structure of the object deformation by expressing the surface motion in terms of a deformation model using a small set of parameters. The recovered motion model allows 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. We formulate our approach as a single optimization problem that simultaneously aligns partial surface data and recovers the motion model, similar to the pairwise registration method of Chang and Zwicker [2009]. This is accomplished without the assistance of markers, user-placed correspondences, a template, or a segmen-

tation 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 main 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 initial pairwise alignments of adjacent frames.

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. Our method is most useful when it is impossible to acquire a complete, static pose of the subject, because we do not require a template shape nor a template rig with a predefined skeletal structure. However, our method is mainly applicable to articulated subjects, and it may produce a rough piecewise rigid approximation of the surface 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

Template-Based Reconstruction. A popular approach to reconstruct deforming sequences of range scans is to fit a template to the 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 track 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].

Templateless 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.

The algorithm by Wand et al. [2009] reconstructs an animated 3D model from range scan sequences without using a template. Compared to the works mentioned above, this method is more robust to missing data in the scans. It 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. 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

Algorithm 1: ARTICULATED GLOBAL REGISTRATION

Data: A sequence of range scans, denoted $(F_0, \ldots, F_{n-1})$
Result: Dynamic sample graph G of the completed surface, weights $\mathcal{W}$ for each vertex $v \in G$, rigid transformations $\mathcal{T}$ for all parts and frames

1 **begin**
2 Compute initial pairwise registration of adjacent frames (Section 4);
3 Initialize dynamic sample graph G from F_0 (Section 6);
4 $i \leftarrow 0$;
5 **while** $F_i \neq F_{n-1}$ **do**
6 Apply initial pairwise registration of F_i and F_{i+1} (Section 6);
7 Detect occluded parts in F_{i+1} (Section 7);
8 Perform global registration of $\{F_0, \ldots, F_{i+1}\}$ (Algorithm 2);
9 Update dynamic sample graph G (Section 6);
10 $i \leftarrow i + 1$;
11 Resample G densely and reconstruct surface mesh (Section 8.1);
12 **return** $G, \mathcal{W}, \mathcal{T}$;
13 **end**

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 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 from scan to 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. In the following sections, we will describe each part of the algorithm in detail. In a preprocessing step we solve for an initial pairwise registration for each pair of adjacent frames $(F_0, F_1), (F_1, F_2), \ldots, (F_{n-2}, F_{n-1})$ (line 2, Section 4). 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.

In the core component of our approach we refine this initial registration and produce a global registration of all frames (lines 3–10). The main idea is to optimize the transformations and weights simultaneously across all frames to align them to a common reference pose. The optimization operates on a central data structure that we call the dynamic sample graph (DSG). The DSG is formed on a subset of points sampled from the input scans. We select the points such that they form a uniform sampling of the complete surface that has been registered so far. The graph is dynamic because we incorporate new data as it becomes available in new frames that are added to the optimization. The main advantage of the DSG is that it removes redundancies in the input range scans and, hence, makes the global registration tractable.

To give an overview of the optimization process outlined in Algorithm 1, we start by creating the initial DSG (line 3, Section 6). Then, the frames are introduced one at a time into the global registration (lines 5–10). For each frame, we apply the initial pairwise registration (line 6, Section 6) which gives an initial alignment of the new frame to the surface registered so far. We then detect occlusions and disocclusions of surface parts in the new frame (line 7, Section 7) and optimize the transformations $\mathcal{T}$ and weights $\mathcal{W}$ to simultaneously align all frames (line 8, Section 5.3). Lastly we update the DSG (line 9, Section 6) to incorporate new samples from the new frame, and we move on to the next frame. After finishing the entire sequence, the final post-processing step is to resample the surface densely and reconstruct a mesh of the completed surface (line 11, Section 8.1).

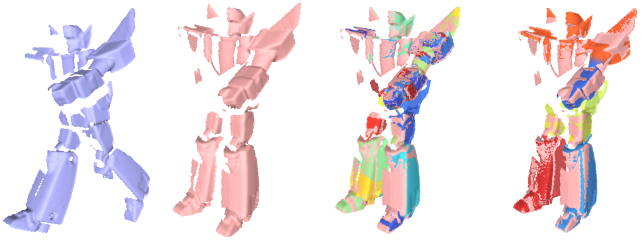
4. INITIAL PAIRWISE REGISTRATION

In a preprocessing step we solve for an initial pairwise 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 applying the transformations produces an alignment that minimizes distance between the scans while preserving their shape.

The details of the method are the same as originally described by Chang and Zwicker [2008]. 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 restrict the optimization to a small subset of points (typically a few thousand points) sampled uniformly from each scan using best-candidate sampling [Mitchell 1991]. This makes sense for articulated movement, where the number of unique transformations producing the movement is small compared to the number of scanned points. We build a k-nearest neighbor graph with $k = 15$ on the subset of points to specify the smoothness constraints necessary for the optimization [Chang and Zwicker 2008].

After the optimization, we propagate the transformations assigned to the subset to all remaining points using nearest-neighbor interpolation. This produces the initial pairwise registration that we will use as an initialization for the global registration.

A comparison of the optimization using all points versus using a subset is shown in Figure 2. Although we obtain a good alignment in both cases, the improved method achieves a significant speedup.



(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 initial pairwise registration. With the same parameters, optimizing on a subset of the points produces a similar registration in a fraction of the time. The color variation in (b,c) visualizes how the transformations are assigned to the surface.

Also, the use of the graph improves the connectivity between parts that may be disconnected in the original mesh.

5. GLOBAL REGISTRATION

The core part of our method is the global registration step (line 3-10, Algorithm 1), which optimizes for the best transformations and weights that simultaneously align all introduced frames. Before discussing the details of the algorithm, we first describe our deformation model in more detail.

5.1 Deformation Model

Transformations. We represent the surface motion using a set of rigid transformations in each frame. We designate the first frame as the *reference frame* and define the transformations relative to this reference[1]. Thus, each transformation moves a part of the surface in frame f to align to the corresponding part in the reference frame F_0. We use the notation $T_a^{(f\to\text{Ref})}$ to denote the ath transformation for frame f, which transforms in the direction *from* frame f *to* the reference frame (Figure 3a). To aid our method, the user specifies a maximum number of rigid transformations B used to approximate the surface motion.

Each $T_a^{(f\to\text{Ref})}$ 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 apply the formula $T_a^{(f\to\text{Ref})}(\mathbf{x}) = R_a^{(f\to\text{Ref})}\mathbf{x} + \vec{\mathbf{t}}_a^{(f\to\text{Ref})}$. We also express the relative transformation $T_a^{(f\to g)}$ between any two frames f to g by transforming to the reference and then transforming to the desired frame (Figure 3b) as follows:

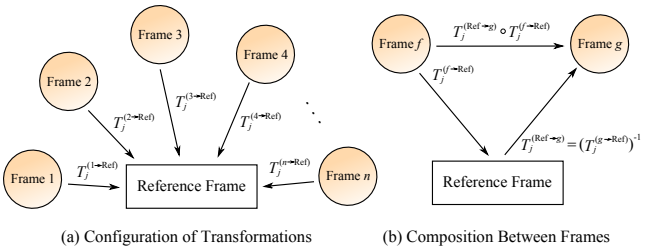
$$T_a^{(f\to g)}(\mathbf{x}) = \left(T_a^{(g\to\text{Ref})^{-1}} \circ T_a^{(f\to\text{Ref})}\right)(\mathbf{x})$$
$$= R_a^{(g\to\text{Ref})^\top}\left[\left(R_a^{(f\to\text{Ref})}\mathbf{x} + \vec{\mathbf{t}}_a^{(f\to\text{Ref})}\right) - \vec{\mathbf{t}}_a^{(g\to\text{Ref})}\right]. \quad (1)$$

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.

Weights. We associate the transformations to the points indirectly by assigning weights to each point. Each weight is a B-dimensional vector $\mathbf{w}(\mathbf{x})$, where the a^{th} component $w_a(\mathbf{x})$ indicates the influence of transformation a 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 small number of transformations can express the movement of the surface.

In our method, 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 $w_{j(\mathbf{x})}(\mathbf{x}) = 1$. Here, we use $j(\mathbf{x})$ to indicate the index of the component of $\mathbf{w}(\mathbf{x})$ with 1.

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



(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.

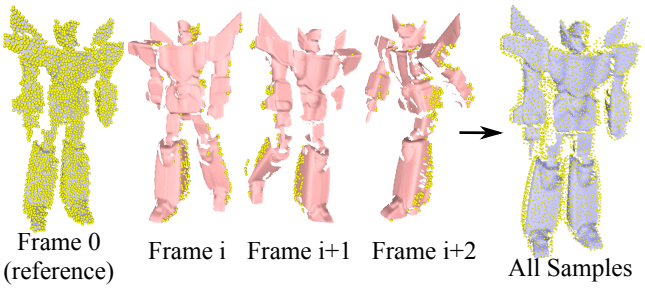


Fig. 4. Initially the vertices of the dynamic sample graph (DSG) are uniformly sampled from the reference frame (far left). As we introduce more frames (i, $i+1$, $i+2$), we add samples only from parts of the surface that are missing in previous frames. On the far right, we plot all sample points after transforming them to frame 0. Because of our careful selection strategy, the samples minimize redundancy and form an overall uniform sampling of the completed surface.

Dynamic Sample Graph (DSG). We introduce the dynamic sample graph (DSG) to efficiently represent the skinning weights and make the optimization over all frames tractable. The vertices of the DSG are a subset of the points from all input frames. We will optimize for skinning weights only on the vertices of the DSG, and interpolate the weights on all other scanned points, thus giving a sparse representation of the weight function. We will also use the edges of the DSG to impose additional constraints on nearby samples to incorporate the skinning model in the optimization. The DSG is dynamic because we add new samples from each new frame that is introduced into the optimization. We select the samples carefully to minimize redundancy, form a uniform distribution, and provide adequate coverage of the entire surface seen so far. This process is illustrated in Figure 4. We discuss details on how to sample the points and form the edges of the DSG later in Section 6.

5.2 Optimization Objective

The goal of the optimization is to solve for weights and transformations that align the DSG to all frames simultaneously. The optimization objective has three terms: (1) $\mathcal{E}_{\text{fit}}(\mathcal{T}, \mathcal{W})$, which measures the alignment distance between the DSG and all frames, (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 of neighboring points to be the same. With coefficients α, β, γ for each term, we write the entire objective as

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

Fitting Objective $\mathcal{E}_{\text{fit}}$. This term measures the alignment distance of the DSG to all frames. Let us denote a sample in the DSG by $\mathbf{x}$. With each sample we also store the index of the frame from which it was selected and denote this by $f(\mathbf{x})$. To evaluate the fitting objective, we transform each sample to all other frames and measure how close it is to the scanned data of these frames. We use a robust error metric based on the distance to the closest corresponding point on the other frame. For a sample $\mathbf{x}$ in frame $f(\mathbf{x})$, we measure its alignment distance to frame g by transforming it to frame g (using $T_{j(\mathbf{x})}^{(f(\mathbf{x})\to g)}(\mathbf{x})$) and finding the closest corresponding point $\mathbf{y}_{j(\mathbf{x})}^{(g)} \in F_g$. This notation represents the point on frame g closest to the transformed position of $\mathbf{x}$, assuming that we apply transformation $j(\mathbf{x})$. Note that the closest point in frame g depends on the transformation $j(\mathbf{x})$ that is assigned to $\mathbf{x}$. We make this explicit in our notation by using the subscript $j(\mathbf{x})$. Once we have the corresponding point, the total alignment distance is given by the formula

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

Here we have computed the distance $d(\cdot,\cdot)$ between $\mathbf{x}$ (in frame $f(\mathbf{x})$ and its corresponding point $\mathbf{y}_{j(\mathbf{x})}^{(g)}$, where both points have been transformed to the reference frame (see Figure 5). The resulting values are summed up over all sample positions $\mathbf{x}$ and all frames g to compute the total alignment distance.

We design the distance $d(\mathbf{x}, \mathbf{y})$ to be robust under missing data and outliers. This metric first determines whether the two points $\mathbf{x}, \mathbf{y}$ are valid corresponding points. We define three criteria for determining validity.

(1) Assume that the sample $\mathbf{x}$ is from frame $f(\mathbf{x})$. Then it will never have a valid corresponding point in any frame g that was added before frame $f(\mathbf{x})$, i.e., when $g < f(\mathbf{x})$. To see this, note that when we add a new frame, we add samples to the DSG only from surface parts that are missing in all previous frames (for details see Section 5.1). Therefore, for $g < f(\mathbf{x})$ no sample added from frame $f(\mathbf{x})$ will have a corresponding point in any frame g.

(2) Because of scanner occlusion, a sample $\mathbf{x}$ from frame f may not have a corresponding point in frame g even when $g > f$. Adapting the strategy from Pekelny and Gotsman [2008], we use simple thresholding to detect this case. The corresponding point $\mathbf{y}$ is invalid when

—the Euclidean distance $\|\mathbf{x} - \mathbf{y}\|$ exceeds a threshold τ_d,
—the angle between their normals exceeds a threshold τ_n,
—or the distance exceeds a smaller threshold τ_b when $\mathbf{y}$ lies on the boundary of F_g.

(3) When we optimize the weights (see Section Section 5.3), $j(\mathbf{x})$ becomes a variable, not a fixed constant. Therefore, we maintain a separate closest point $\mathbf{y}_a^{(g)}$ for each potential transformation $T_a^{(f\to g)}(\mathbf{x})$. Now, consider the current transformation $j(\mathbf{x})$ assigned to $\mathbf{x}$. If the closest point $\mathbf{y}_{j(\mathbf{x})}^{(g)}$ is invalid, then most likely there is no corresponding point at all in frame g. Thus, in this case we set all $\mathbf{y}_a^{(g)}$ for all transformations a as invalid.

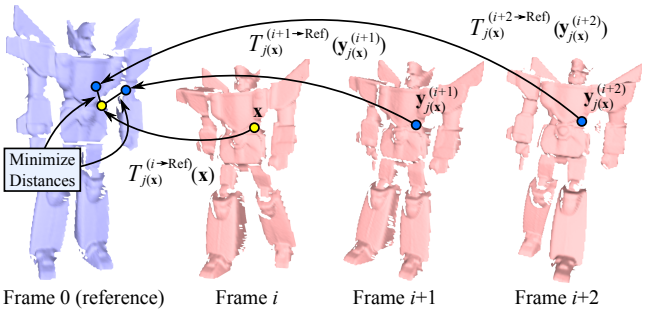


Fig. 5. To measure alignment, we compute distances between sample points $\mathbf{x}$ (yellow) and closest corresponding points $\mathbf{y}_{j(\mathbf{x})}^{(g)}$ (blue) transformed to the reference frame. We add up these distances to measure the alignment of all frames in the sequence. We optimize for the transformations and weights that minimize this total distance.

The reason for this strategy is that, even if the current transformation $j(\mathbf{x})$ assigned to $\mathbf{x}$ is the correct one, the region corresponding to $\mathbf{x}$ in a frame g may be occluded. In this case, the corresponding point $\mathbf{y}_{j(\mathbf{x})}^{(g)}$ will be invalid. By coincidence, however, one of the other transformations may move $\mathbf{x}$ very close to a valid corresponding point, but in a completely wrong location. As a result, the weight optimization will prefer to assign this incorrect transformation. Our strategy prevents this problem by conservatively declaring that there is no valid corresponding point for any transformation if the current transformation does not yield a valid correspondence.

If $\mathbf{x}, \mathbf{y}$ pass the three criteria, we include their distance in $\mathcal{E}_{\text{fit}}$; otherwise their distance is 0 and not included in the term. The distance for valid pairs is a weighted sum of the point-to-point and point-to-plane distance measures. The formula for d is

$$d(\mathbf{x}, \mathbf{y}) = \begin{cases} \eta_{\text{pt}}\, \|\mathbf{x} - \mathbf{y}\|^2 + \eta_{\text{pl}}\, \left((\mathbf{x} - \mathbf{y}) \cdot \vec{\mathbf{n}}_{\mathbf{y}}\right)^2 & \text{if } \mathbf{x}, \mathbf{y} \text{ is valid} \\ 0 & \text{otherwise.} \end{cases} \qquad (4)$$

where $\vec{\mathbf{n}}_{\mathbf{y}}$ is the surface normal of $\mathbf{y}$, transformed along with $\mathbf{y}$ using only the rotational part of the transformation. We use the weights $\eta_{\text{pt}} = 0.2$ and $\eta_{\text{pl}} = 0.8$ for our experiments.

Joint Objective $\mathcal{E}_{\text{joint}}$. The joint term constrains neighboring transformations to agree on a common joint location. 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. First, we will explain the definition of the joint constraints. Then, we will explain how the joint parameters are computed.

We define the joint locations in the reference frame. 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 6 (left).

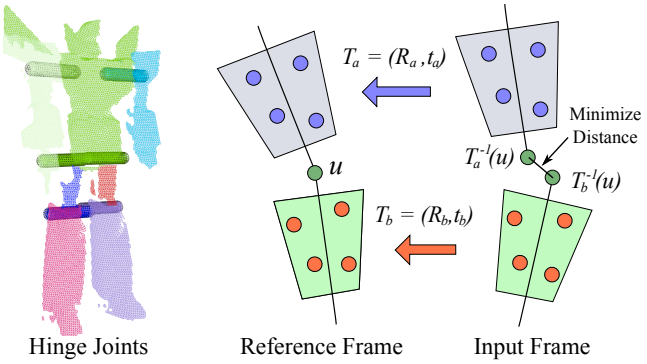


Fig. 6. Estimating and constraining joints in our optimization. (Left) we show hinge joints that are estimated automatically. The bars represent the hinge axes. (Middle & Right) $\mathcal{E}_{\text{joint}}$ constrains the transformed locations of $\mathbf{u}$ to agree on the same point by minimizing the distance between the transformed locations.

Once we know these joint locations and types, we can constrain the transformations to map the joint locations to the same place (Figure 6, middle & right). Let us represent a joint between transformations for parts a and b using the tuple $(\mathbf{u}_{ab}, \vec{\mathbf{v}}_{ab})$. We additionally set a valid/invalid flag for each tuple, depending on whether there actually is a joint between transformations a and b. In the objective, we constrain the joints using the term $\mathcal{E}_{\text{joint}}$:

$$\mathcal{E}_{\text{joint}}(\mathcal{T}) = \sum_{\text{All } F_i} \sum_{\substack{\text{Valid Joints} \\ (a,b)}} \sum_{t \in [-10s..10s]} \left\| T_a^{(i\to\text{Ref})^{-1}}(\mathbf{u}_{ab} + t\vec{\mathbf{v}}_{ab}) - T_b^{(i\to\text{Ref})^{-1}}(\mathbf{u}_{ab} + t\vec{\mathbf{v}}_{ab}) \right\|^2. \tag{5}$$

Here, we use 20 values of t spread in the range $[-10s..10s]$ where s is the mesh resolution (or grid sample spacing)[2]. For a hinge joint, this constrains a set of points along the hinge axis. In the case of a ball joint, we set $\vec{\mathbf{v}}_{ab} = 0$, so this term constrains only one point $\mathbf{u}_{ab}$. 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 (a, b) are likely to share a joint in between, and we determine the location using the transformations a b that we have solved for each frame.

We use the edges of the DSG to find pairs of transformations (a, b) likely to have a joint. If there are many edges in the DSG where one end has weight corresponding to transformation a and the other end with weight b, then these transformations neighbor each other and are likely to share a joint in between. On the other hand, if there are no such edges, then most likely there is not a joint between these transformations. To help our discussion, let an edge in the DSG be *incident to transformation* a if one of its end points $\mathbf{x} \in S$ has weight $j(\mathbf{x}) = a$. If either of the following ratios exceeds a threshold (set to 15%):

$$\frac{\#\text{ edges incident to both } a, b}{\#\text{ edges incident to } a}, \quad \frac{\#\text{ edges incident to both } a, b}{\#\text{ edges incident to } b} \tag{6}$$

we take the pair a, b as a candidate for sharing a joint. We then average all endpoints of edges incident to both a, b to obtain an estimate $\mathbf{u}_{\text{est}} \in \mathbb{R}^3$ of the joint location. Since we will define joint locations on the reference frame, we compute $\mathbf{u}_{\text{est}}$ on the reference frame. This estimate of the joint location serves to regularize the optimization below and to prune unreasonable estimates of the joint location.

Once we have a set of candidate pairs (a, b) 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 (a, b):

$$\underset{\mathbf{u}\in\mathbb{R}^3}{\text{argmin}} \sum_{\text{All Frames } F_i} \left\| T_a^{(i\to\text{Ref})^{-1}}(\mathbf{u}) - T_b^{(i\to\text{Ref})^{-1}}(\mathbf{u}) \right\|^2 + \lambda \left\| \mathbf{u} - \mathbf{u}_{\text{est}} \right\|^2 \tag{7}$$

The first term aims to find the location $\mathbf{u}$ that stays fixed under the transformations, and the second 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.

We first try to detect hinge joints using this minimization. We initially set $\lambda = 0$ and solve the least-squares problem using the SVD. The joint is a hinge if the ratio of the smallest singular value to the sum of the singular values is less than a threshold (set to 0.1). If this is the case, 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. The final hinge joint parameter $\mathbf{u}$ is the point on this line that is closest to $\mathbf{u}_{\text{est}}$, and $\vec{\mathbf{v}}$ is the normalized line direction $\vec{\mathbf{v}}'/\|\vec{\mathbf{v}}'\|$.

If the joint is not a hinge, it is a ball joint and we determine a single joint location $\mathbf{u}$ with $\vec{\mathbf{v}} = 0$. In this case, we solve the minimization once again, but with $\lambda = 0.1$ to pull the solution nearby the estimated location.

Finally, we perform a sanity check after solving for the joint location. If the distance between the solved and estimated locations exceeds the length of a hinge ($\|\mathbf{u} - \mathbf{u}_{\text{est}}\| > 20s$), we consider it an unreasonable estimate and discard the joint.

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 DSG.

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

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

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

5.3 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 6-10), and in the second phase, we keep the transformations fixed and solve for the weights (lines 16-22). This strategy works well in practice and produces a good alignment within a few iterations. Also, we try to detect if previously disappeared parts have reappeared in the new frame (line 12).

In our experiments, we observed that the transformations for a frame do not change much after the frame is first introduced and

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

Algorithm 2: OPTIMIZE $\mathcal{T}, \mathcal{W}$ $(DSG, \mathcal{T}, \mathcal{W}, F_0, \ldots, F_{i+1})$

Data: Dynamic sample graph (DSG), associated weights $\mathcal{W}$, transformations for all frames $\mathcal{T}$, and all initialized input frames $F_0, \ldots, F_{i+1}$
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 $\mathcal{T}$)
5 Re-estimate joint locations and types;
6 **while** *Not converged* **do**
7 Update the closest points $\mathbf{y}_{j(\mathbf{x})}^{(g)}$ for all $\mathbf{x} \in S$ and all F_g, where $g \in [0\,..\,i{+}1]$;
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 Detect reappearing transformations in $F_{\text{i+1}}$ by aligning occluded parts with unmatched surface points (Section 7);
13 Check convergence criteria;
14 **if** converged **then break**;
15 **begin** (Phase 2: Solve for the weights $\mathcal{W}$)
16 Update the closest points $\mathbf{y}_a^{(g)}$ for all samples $\mathbf{x}$, all F_g, and all transformations a, where $g \in [0\,..\,i{+}1]$ and $a \in [0\,..\,B]$;
17 Precompute $\mathcal{E}_{\text{fit}}$ for all $\mathbf{x}$ and for all a;
18 Create $\mathcal{E}_{\text{weight}}$ using the edges of the DSG;
19 Solve discrete labeling using α-expansion;
20 Discard parts that are too small;
21 Reuse unassigned weight components by splitting regions with highest $\mathcal{E}_{\text{fit}}$ error;
22 Update the weights for each sample $\mathbf{x}$;
23 **end**
24 **end**

optimized. Therefore, we solve for the transformations only on the newest c frames that have been optimized. We can think of this as a "sliding window" in which to optimize the transformations. Lowering the value of c 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 location of the closest corresponding points $\mathbf{y}_{j(\mathbf{x})}^{(g)}$ depend on the transformations, we use an iterative approach in the spirit of the iterative closest point (ICP) algorithm [Besl and McKay 1992] (line 6–10 in Algorithm 2). We first keep the transformations fixed and compute the closest points, then we keep the corresponding points $\mathbf{y}_{j(\mathbf{x})}^{(g)}$ fixed and optimize the transformations, and we repeat this alternation until convergence.

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 closest points) involving transformations from frames outside of the set of interest. This significantly reduces the time to perform this phase.

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 solving for the value of $j(\mathbf{x})$ for each sample point that minimizes 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 edges of the DSG directly to specify smoothness constraints between points. To save computation time during the optimization, we precompute $\mathcal{E}_{\text{fit}}$ in the DSG and store the values in a 2-dimensional hash table for quick access. We precompute and store the summand $d(\mathbf{x}, \mathbf{y})$ of $\mathcal{E}_{\text{fit}}$ separately per sample $\mathbf{x}$ and per transformation a, summed over all frames g.

After the optimization, it may be the case that some transformations are applied to too few samples of the DSG. To facilitate discussion, let us partition the DSG into its rigid parts, i.e. the subsets $S_a = \{\mathbf{x} \in \text{DSG} \mid j(\mathbf{x}) = a\}$. If the number of points for a rigid part is less than 1% of the total number of sample points, then we remove the rigid part from the DSG and replace the weight of its points with the weight of the closest point from a different part. This results in "unused" transformations that are not assigned to any samples.

Instead of completely throwing away these unused transformations, we can re-introduce them in a different location to reduce registration error. 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]. This adds more degrees of freedom, allowing the optimization to refine the alignment further for the region.

Specifically, we compute an average registration error for each rigid part and its current transformation by evaluating the fitting objective (Equation 3) for the points of each part separately and dividing by the number of points. We then split the part with the highest registration error into two. We split by randomly selecting two seed points and dividing the points of the part according to which seed is closer. Then, we leave one of the new parts as is, but we replace the weights of the points in the other with an unused transformation. This splitting process is continued until the highest registration error is below a threshold $0.1s$, or there are no unused transformations left.

Checking for Convergence. We perform a 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. Denoting the total error residual at iteration ι as E_ι, we apply the criterion $|E_\iota - E_{\iota+1}| < \epsilon(1 + E_\iota)$ (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 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.

6. MAINTAINING THE DYNAMIC SAMPLE GRAPH

The dynamic sample graph (DSG) is an important component that is involved in all stages of our algorithm. In this section, we discuss remaining details about how we manage the DSG, including how to transfer the initial pairwise registration (Section 4) into a format compatible with the DSG, how to update the DSG with new samples when a frame is added, and how to interpolate the sparse weight function defined on the vertices of the DSG.

Initializing the DSG. We initially create the DSG by uniformly sampling a set of points on the reference frame. As a preprocessing step, we sample a fixed fraction r of the points in every input frame using the best-candidate technique [Mitchell 1991]. This results in a Poisson-disk sampling of the points in each input frame. We denote the set of sampled points in frame f as U_f. Then, the initial set of samples in the DSG is exactly U_0. As we register each new F_i, we will select samples from U_i to add to the DSG.

Applying the Initial Pairwise Registration. Every time we introduce a new frame F_{i+1} into the global registration, we need to find the initial value of each transformation $T_a^{(i+1\to\text{Ref})}$, where a ranges from 0 to B. For this, we use the result of the initial pairwise registration between frame i and $i+1$. However, the initial pairwise registration specifies a transformation for *every vertex* of frame i, whereas we want to apply the initial registration to the current DSG.

To apply the initial pairwise registration, we first partition the DSG into its rigid parts, i.e. the subsets $S_a = \{\mathbf{x} \in \text{DSG} \mid j(\mathbf{x}) = a\}$. Now we determine an initial transformation for each rigid part by blending the corresponding transformations from the initial pairwise registration.

For each point $\mathbf{x} \in S_a$, we find the closest point $\mathbf{y}$ in frame i and store (in a list) the transformation that was assigned to $\mathbf{y}$ in the initial pairwise registration. This results in a list of transformations for the subset. Then, we uniformly blend all transformations in the list using Dual Quaternion Linear Blending (DLB) [Kavan et al. 2008] to produce an initial transformation T_a^{init} from frame i to frame $i+1$. Lastly, we concatenate with the transformation from frame i to produce the initial value: $T_a^{(i+1\to\text{Ref})} = T_a^{(i\to\text{Ref})} \circ T_a^{\text{init}\,-1}$. Since we blend transformations, this produces a slightly different result from the initial pairwise registration, but the differences were negligible in practice.

There is one exception to this procedure when we apply the initial pairwise registration of the first two frames F_0 and F_1. At this point in the algorithm, the DSG has just been created using the points in frame 0 (i.e. $S = U_0$), and the samples do not have any weights. In this case, S is exactly a subset of frame 0, so we directly copy the transformations from the initial pairwise registration while limiting the total number of unique transformations to the maximum B.

Updating the DSG. After applying the initial pairwise registration for the new frame F_{i+1}, the algorithm moves into the global registration phase to align F_{i+1} to the rest of the frames. After each global registration, we update the DSG to reflect changes in the registration. When we update, we actually resample the DSG from scratch. This is because the global registration changes the alignment of all frames, and certain samples that were not redundant before may become redundant and need to be removed.

To start, we create a new and empty DSG, and initialize its samples to U_0. Then, for each frame g, we add points from U_g to the DSG that do not overlap with the points added so far. Also, we only add points for which we can determine a valid weight using the old DSG. These techniques are inspired by the work of Pekelny and Gotsman [2008].

To decide overlap, we first transform the current points in the (new) DSG to frame g. Then, a point from U_g overlaps with the DSG if the distance to the closest DSG point is less than a threshold τ_s. To make this more robust to registration error, we project the distance onto the plane of the DSG point (with the DSG point's surface normal) if the surface normals of the two points differ by less than 90°.

We interpolate the weight values of the old DSG to determine the weight of new points from U_g. First, we transform the old DSG to frame g and divide the old DSG into rigid parts. Then, for each rigid part, we compute the closest distance between the new point and the part. We convert the distances to scores by normalizing them to sum to 1. The weight of the new point is the transformation of the part with the highest score, but only if the highest score is greater than three times the upper quartile (median of the largest half) of all scores. Otherwise, we consider it an ambiguous case, and a valid weight cannot be determined for the point. Also, the weight is invalid if the highest scoring transformation is marked as occluded for frame g (more details about occlusion in Section 7).

After the resampling is complete, we form edges on the new DSG. We transform it to the reference frame and compute the k-nearest neighbor graph of its samples ($k = 15$). To prevent undesired edges between separate (but spatially near) parts, we discard edges that stretch in length more than twice when transformed to each frame g. However, if the edge is between parts that share a joint, we do not discard the edge. This is because discarding these "joint edges" may bias the discrete labeling optimization and cause the boundary between parts to get "stuck" in particular locations.

After forming the edges, we finally discard the old one and replace it with the new. This concludes the updating process for the DSG.

7. HANDLING OCCLUSION

Detecting Occluded Parts. When a part of the surface is partially or completely occluded 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 7 of Algorithm 1).

As before, let us partition the DSG into its rigid parts. Right after loading and applying the initial pairwise registration (line 6 of Algorithm 1), we determine whether each transformation is occluded in frame $i+1$.

First, we update the closest corresponding points $\mathbf{y}_{j(\mathbf{x})}^{(i+1)}$ for each DSG point $\mathbf{x}$ and compute the distance $d(\mathbf{x}, \mathbf{y}_{j(\mathbf{x})}^{(i+1)})$ using the robust error metric (Equation 4). If the number of points in a rigid part with non-zero distance falls below a threshold (5, or 5% of the part), then we mark the part's transformation as "occluded" for frame $i+1$. The rigid part is considered "occluded" as well.

Excluding Occluded Transformations in the Optimization. When we solve for the transformations (Phase 1) and weights (Phase 2) in Algorithm 2, we need to handle occluded transformations and parts. In Phase 1, we do not solve for occluded transformations. Instead, 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].

While we can simply exclude transformations in Phase 1, we cannot do the same for optimizing the weights in Phase 2. Here, the algorithm must solve a binary weight for each sample to minimize the registration error. The problem is to define a useful "registration error" for a transformation that is occluded in the frame. We cannot assign a zero error, because the optimization would prefer to assign the weight for the occluded transformation. It also cannot be too high, because the optimization would prefer to not assign the weight for the occluded transformation to any samples at all.

Recall that when we optimize the weights in Phase 2 (Section 5.3), we compute and store the summand of $\mathcal{E}_{\text{fit}}$ for all samples $\mathbf{x}$, all transformations a, and all frames g. Suppose that for some sample $\mathbf{x}$, transformation a is occluded in frame g. In this case we use the error value of the currently selected transformation $j(\mathbf{x})$ for the occluded transformation a. If transformation $j(\mathbf{x})$ is also occluded, then we use the minimum error value among all non-occluded transformations. This strategy worked well in our experiments.

Detecting 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. If the part happens to reappear nearby its last seen 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 points. Our algorithm detects and handles this case during the global registration (line 12 of Algorithm 2). Note that detecting reappearing parts cannot be handled by our initial pairwise registration, because it can only align parts not occluded in *both* the source and target.

To detect if an occluded part is reappearing in frame $i+1$, we first transform the DSG to frame $i+1$ and determine the weights of each point of U_{i+1} by applying the algorithm of Section 6 again in a separate step. If a valid weight cannot be determined (i.e. it is an ambiguous case, or the highest scoring transformation is occluded), then we consider the point to be "unmatched" and add it to a set M of unmatched points. If there is a sufficient number of unmatched points ($|M| > 0.1|U_{i+1}|$), we interpret this as an indication that a previously occluded part is reappearing. Therefore, we attempt to match these points by aligning them with transformations that were previously marked as occluded for frame $i+1$.

Here, we use the same procedure as Phase 1 of Algorithm 2 to solve for the values of the occluded transformations (Section 5.3). However, we expect unmatched points to be far away from the current position of the DSG. Therefore, we adjust the optimization, where

—we only solve for the values of the occluded transformations by aligning occluded parts of frame $i+1$ to M,
—we set thresholds to higher values: $\tau_d = 50s$ and $\tau_n = 45^\circ$,
—and we increase the weight of the joint constraint to $\beta = 1000$.

These modifications ensure that we obtain an alignment even when the points in M are far away from the DSG, while the higher value of β prevents the optimization from falling into local minima. After solving for the occluded transformations, we run the occlusion detection routine once more to update the occlusion status of each transformation. **MZ: it would be great to show a figure with an example, where a reappearing part is detected and registered**

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 4 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 middle row shows the reconstructed mesh using the algorithm, with weights obtained by interpolating the weights on the sample set. The bottom row shows the estimated joint locations, where hinge joints are represented by a short stick and ball joints by a sphere. Both the reconstruction results and the weight estimation are faithful to the input data.

To test our algorithm on a more deformable subject, we acquired two range scan sequences of a pink panther toy using a Konica Minolta VI-910 laser scanner. We animated one sequence with a slower motion, while the other sequence had a faster motion. Reconstruction results are shown in Figure 9. Although the furry texture on the toy created noise on the scanned surface, we obtained a reasonable reconstruction of both the surface geometry and the weights.

Finally, we tested our algorithm on synthetic depth scans of a walking man, where the camera is rotating around the subject. To test the effect of occlusion in our algorithm, we captured two sequences, one using a single virtual camera, and the other using two virtual cameras 90° apart. The reconstruction results are shown in Figure 10. The results from both datasets are reasonable, but the first sequence was less successful due to the large amount of occlusion of the arms. With two virtual cameras, we obtained a better reconstruction that was able to reproduce the fine detail of the hands.

8.2 Parameters

The main parameters of our algorithm are the maximum number of transformations B, weights for each term in the objective function, and thresholds that control the sampling and closest point computation. We expressed many parameters relative to the grid sample spacing s, which is the average distance between the scanned points of the dataset.

Although the user needs to specify the maximum 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 could have the user specify a maximum alignment error ϵ and change the algorithm to add additional transformations until the alignment error is within ϵ.

The value of B for each dataset is shown in Table I. 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 instructed the algorithm to sample 6% to 20% of the points 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 the validity of the closest corresponding points, we

Fig. 7. Reconstruction results for the Robot dataset.

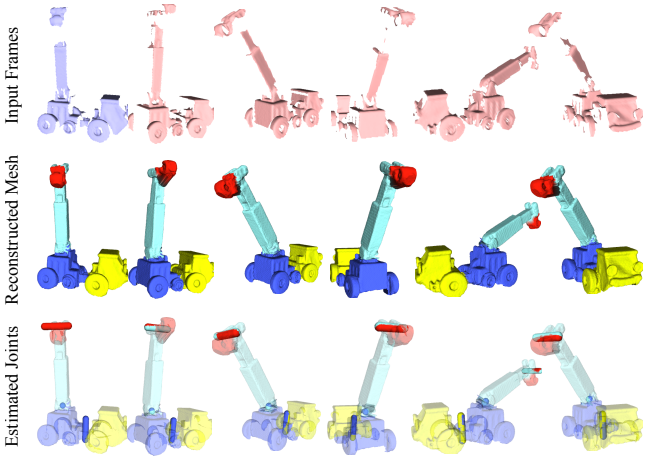


Fig. 8. Reconstruction results for the Car dataset.

used $\tau_d = 10s, \tau_n = 45°$, and $\tau_b = s$. When we match reappearing parts, we increased these values so that τ_d is 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

We performed our experiments using a single core of an Intel Xeon 2.5 GHz processor. The timing results are 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 other closest point matching 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 some preprocessing time to compute spin images and estimate the principal curvature frame at each vertex.

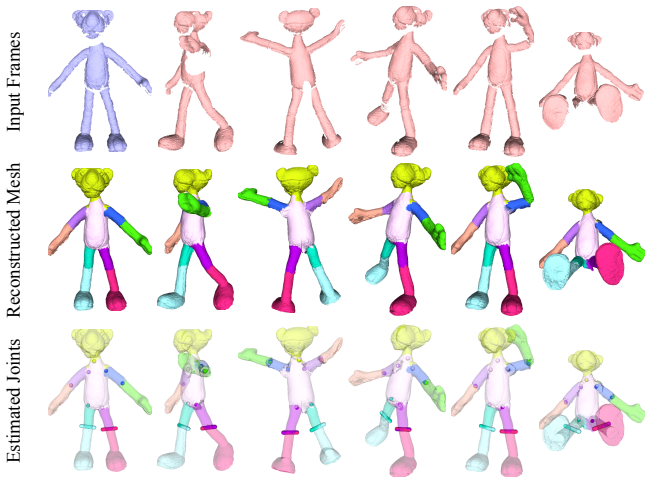


Fig. 9. Reconstruction results for the Pink Panther dataset with faster input motion.

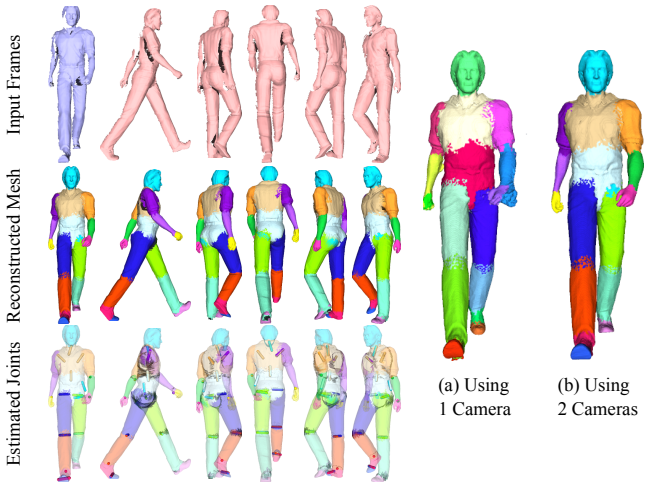


Fig. 10. Reconstruction results for the synthetic Walking Man dataset. On the right, (a) and (b) show a comparison of the reconstruction using scans from one and two virtual cameras.

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 DSG	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
Updating DSG	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

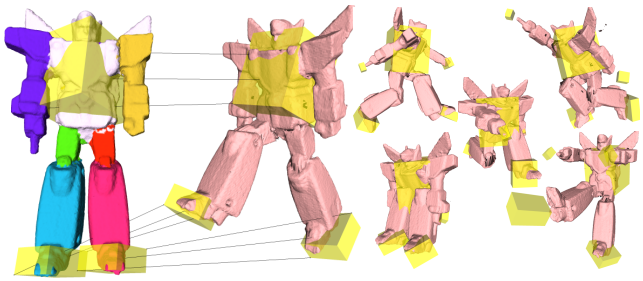


Fig. 11. Reposing the reconstructed robot. Using the solved weights and the hinge joints, we can perform interactive IK on the reconstructed model.

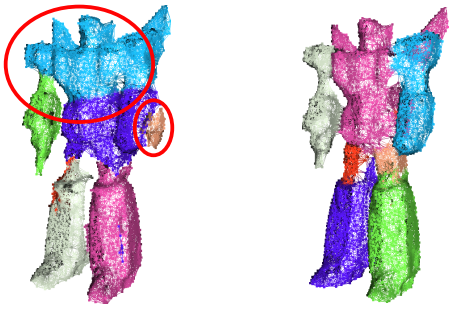
8.4 Inverse-Kinematics Application

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 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 5.3) 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 11 shows examples of different poses of the robot created by our system.

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. This strategy is essentially a pairwise registration that is applied repeatedly for each frame, because it only performs the registration between the accumulated samples and the current frame.

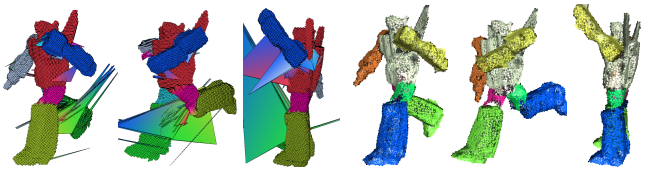
The main problem with the sequential registration approach is that it cannot reliably estimate the articulated structure (i.e. weights) based on the movement observed in just two frames at a time. This complicates the situation further for occlusion detection and recovery, which rely on a reliable estimate of the articulated structure. A comparison between the sequential and simultaneous strategies is shown in Figure 12. Here, we have used the two strategies to align 40 robot frames, and we display the DSG which roughly shows the reconstructed geometry. On the left, we can see that the sequential strategy did not produce the correct weights. 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 the result using simultaneous registration using the same parameters. The registration is more accurate, and the algorithm produced the correct weights that reflect the movement of all frames.



(a) Result Using Sequential Registration

(b) Result Using Simultaneous Registration

Fig. 12. Comparing sequential and simultaneous registration. (a) The sequential strategy gives an unreliable estimate of the articulated structure (large red oval), because it only uses the movement observed in two frames at a time. This leads to an imprecise registration (smaller red oval). (b) The simultaneous strategy can correctly estimate the weights that reflect the movement observed in all frames. The registration is more precise, as well as the estimated surface geometry.



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

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

Fig. 13. Comparing registration results using 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. The artifacts with the grid are absent when using the graph.

8.6 Grid-Based Weights vs. Graph-Based Weights

We compare the benefit of using a graph vs. using a grid for representing the weight function. We implemented the simultaneous registration using a grid and compared the results to a graph-based implementation. 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, while it only took 28.36 seconds per frame for the graph based strategy (excluding initialization time in both cases).

Second, the graph-based representation dealt robustly with topology issues. Since we can prune edges of the graph based on the optimized motion, the algorithm can handle topologically difficult cases robustly. An example of this is shown in Figure 13, where we display the grid and graph deformed according to the optimized weights and transformations. The grid based result shown on the left has many problems where grid cells stretch apart. This is because the limited resolution of the grid cannot resolve the left and right leg of the robot when they move close together. In contrast, the graph-based result shown on the right does not suffer from this issue.

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

Fig. 15. 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 DSG. 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).

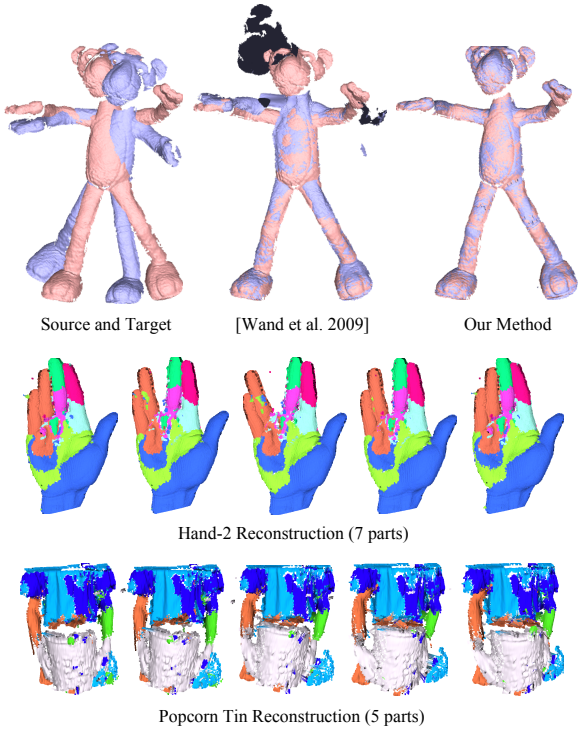


Fig. 14. Articulated registration on the hand-2 and popcorn tin datasets used by Wand et al. [2009]. Our algorithm is able to produce coarse approximations of the non-rigid motion exhibited in these datasets.

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 reconstruct the entire sequence because there was too much motion between the frames. This is because they rely only on a local optimization using closest points, whereas our method uses a robust initial pairwise initialization that is able to automatically handle frames with large motion. An example of this is shown in the top row of Figure 14. Our method (right) produces a correct registration, while their method (middle) fails for this pair.

We also tested our algorithm on several examples from Wand et al. [2009]. Figure 14 (right) shows reconstructions of the hand-2 and popcorn tin datasets, and Figure 15 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, it does not reproduce some fine details in the surface deformation.

9. SUMMARY AND CONCLUSION

We have presented a method to reconstruct an articulated 3D model 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 used an improved robust registration to solve for a initial pairwise registration between pairs of adjacent frames in the sequence. Then, we formulated a simultaneous registration of all input frames to minimize registration error. This optimization included joint constraints that preserves the connectivity between parts and automatically handled cases when parts are disappear or reappear. We demonstrated that we can reconstruct a full 3D articulated model without relying on markers, a segmentation, or a template. Finally, we demonstrated that the reconstructed model is deformable and can be interactively manipulated into new poses using a simple inverse-kinematics 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 from the front, and the next frame captures the surface from the back, 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 adjacent pairs of frames. A temporal ordering of the scans, for example, would produce a sequence with a reasonable amount of overlap. However, even this is not enough sometimes 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 15.

Another shortcoming of our ICP-based registration is the handling of "slippable" parts such as cylinders. For example, the fingers of the hand example in Figure 15 have cylindrical symmetry, so the ICP registration can converge into a state where the segments of the fingers are "twisted" or rotated about the axis of symmetry (Figure 15c). 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 trans-

formations 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.

We would also like to reduce the parameters in our algorithm. An alternative to specifying various thresholds is to use a 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, if we can detect when reasonable weights have been obtained, we can skip the weight optimization step to save time in the algorithm. We believe that an improved version of our algorithm along these lines can be implemented for real-time 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 the ANN library, Y. Boykov, O. Veksler, R. Zabih for an implementation of their graph cuts algorithm, M. Matsumoto for SFMT, S. Toledo for TAUCS, and J. Manson for 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.