Reconstructing Articulated Rigged Models from RGB-D Videos

Dimitrios Tzionas[1,2] and Juergen Gall[1](✉)

[1] University of Bonn, Bonn, Germany
{tzionas,gall}@iai.uni-bonn.de
[2] MPI for Intelligent Systems, Tübingen, Germany

Abstract. Although commercial and open-source software exist to reconstruct a static object from a sequence recorded with an RGB-D sensor, there is a lack of tools that build rigged models of articulated objects that deform realistically and can be used for tracking or animation. In this work, we fill this gap and propose a method that creates a fully rigged model of an articulated object from depth data of a single sensor. To this end, we combine deformable mesh tracking, motion segmentation based on spectral clustering and skeletonization based on mean curvature flow. The fully rigged model then consists of a watertight mesh, embedded skeleton, and skinning weights.

Keywords: Kinematic model learning · Skeletonization · Rigged model acquisition · Deformable tracking · Spectral clustering · Mean curvature flow

1 Introduction

With the increasing popularity of depth cameras, the reconstruction of rigid scenes or objects at home has become affordable for any user [1] and together with 3D printers allows novel applications [2]. Many objects, however, are non-rigid and their motion can be modeled by an articulated skeleton. Although articulated models are highly relevant for computer graphic applications [3] including virtual or augmented reality and robotic applications [4], there is no approach that builds from a sequence of depth data a fully rigged 3D mesh with a skeleton structure that describes the articulated deformation model.

In the context of computer graphics, methods for automatic rigging have been proposed. In [3], for instance, the geometric shape of a static mesh is used to fit a predefined skeleton into the mesh. More detailed human characters including cloth simulation have been reconstructed from multi-camera video data in [5]. Both approaches, however, assume that the skeleton structure is given. On the contrary, the skeleton structure can be estimated from high-quality mesh animations [6]. The approach, however, cannot be applied to depth data. At the end, we have a typical chicken-and-egg problem. If a rigged model with predefined skeleton is given the mesh deformations can be estimated accurately [7] and if the mesh deformations are known the skeleton structure can be estimated [6].

In this paper, we propose an approach to address this dilemma and create a fully rigged model from depth data of a single sensor. To this end, we first create a static mesh model of the object. We then reconstruct the motion of the mesh in a sequence captured with a depth sensor by deformable mesh tracking. Standard tracking, however, fails since it maps the entire mesh to the visible point cloud. As a result, the object is squeezed as shown in Fig. 4. We therefore reduce the thinning artifacts by a strong regularizer that prefers smooth mesh deformations. Although the regularizer also introduces artifacts by oversmoothing the captured motion, in particular at joint positions as shown for the pipe sequence in Fig. 1, the mesh can be segmented into meaningful parts by spectral clustering based on the captured mesh motion as shown in Fig. 5. The skeleton structure consisting of joints and bones is then estimated based on the mesh segments and mean curvature flow.

As a result, our approach is the first method that creates a fully rigged model of an articulated object consisting of a watertight mesh, embedded skeleton, and skinning weights from depth data. Such models can be used for animation, virtual or augmented reality, or in the context of robot-object manipulation. We perform a quantitative evaluation with five objects of varying size and deformation characteristics and provide a thorough analysis of the parameters.

2 Related Work

Reconstructing articulated objects has attracted a lot of interest during the past decade. Due to the popularity of different image sensors over the years, research focus has gradually shifted from reconstructing 2D skeletons from RGB data [8–11] to 3D skeletons from RGB [12–15] or RGB-D data [4,16,17].

A popular method for extracting 2D skeletons from videos uses a factorization-based approach for motion segmentation. In [8,9] articulated motion is modeled by a set of independent motion subspaces and the joint locations are obtained from the intersections of connected motion segments. A probabilistic graphical model has been proposed in [10]. The skeleton structure is inferred from 2D feature trajectories by maximum likelihood estimation and the joints are located in the center of the motion segments. Recently, [11] combine a fine-to-coarse motion segmentation based on iterative randomized voting with a distance function based on contour-pruned skeletonization. The kinematic model is inferred with a minimum spanning tree approach.

In order to obtain 3D skeletons from RGB videos, structure-from-motion (SfM) approaches can be used. [12] perform simultaneous segmentation and sparse 3D reconstruction of articulated motion with a cost function minimizing the re-projection error of sparse 2D features, while a spatial prior favors smooth segmentation. The method is able to compute the number of joints and recover from local minima, while occlusions are handled by incorporating partial sequences into the optimization. In contrast to [18], it is able to reconstruct complex articulated structures. [15] use ray-space optimization to estimate 3D trajectories from 2D trajectories. The approach, however, assumes that the number of parts is known. In [13,14] markers are attached to the objects to get precise

3D pose estimations of object parts. They use a probabilistic approach with a mixture of parametrized and parameter-free representations based on Gaussian processes. The skeleton structure is inferred by computing the minimum spanning tree over all connected parts.

The recent advances in RGB-D sensors allow to work fully in 3D. An early approach [16] uses sparse KLT and SIFT features and groups consistent 3D trajectories with a greedy approach. The kinematic model is inferred by sequentially fitting a prismatic and a rotational joint with RANSAC. In [4] the 3D trajectories are clustered by density-based spatial clustering. For each cluster, the 3D pose is estimated and the approach [14] is applied to infer the skeleton structure. Recently, [17] presented a method that combines shape reconstruction with the estimation of the skeleton structure. While these approaches operate only with point clouds, our approach generates fully rigged models consisting of a watertight mesh, embedded skeleton, and skinning weights.

3 Mesh Motion

Our approach consists of three steps. We first create a watertight mesh of the object using a depth sensor that is moved around the object while the object is not moving. Creating meshes from static objects can be done with standard software. In our experiments, we use Skanect [19] with optional automatic mesh cleaning using MeshLab [20]. In the second step, we record a sequence where the object is deformed by hand-object interaction and track the mesh to obtain the mesh motion. In the third step, we estimate the skeleton structure and rig the model. The third step will be described in Sect. 4.

3.1 Preprocessing

For tracking, we preprocess each frame of the RGB-D sensor. We first discard points that are far away and only keep points that are within a 3D volume. This is actually not necessary but it avoids unnecessary processing like normal computation for irrelevant points. Since the objects are manipulated by hands, we discard the hands by skin color segmentation on the RGB image using a Gaussian mixtures model (GMM) [21]. The remaining points are then smoothed by a bilateral filter [22] and normals are computed as in [23].

3.2 Mesh Tracking

For mesh tracking, we capitalize on a Laplacian deformation framework similar to [24]. While in [7,25] a Laplacian deformation framework was combined with skeleton-based tracking in the context of a multi-camera setup, we use the Laplacian deformation framework directly for obtaining the mesh motion of an object with unknown skeleton structure. Since we use only one camera and not an expensive multi-camera setup, we observe only a portion of the object and the regularizer will be very important as we will show in the experiments.

For mesh tracking, we align the mesh $\mathcal{M}$ with the preprocessed depth data D by minimizing the objective function

$$E(\mathcal{M}, D) = \mathcal{E}_{smooth}(\mathcal{M}) + \gamma_{def}\Bigg(\mathcal{E}_{model\to data}(\mathcal{M}, D) + \mathcal{E}_{data\to model}(\mathcal{M}, D)\Bigg). \tag{1}$$

with respect to the vertex positions of the mesh $\mathcal{M}$. The objective function consists of a smoothness term $\mathcal{E}_{smooth}$ that preserves geometry by penalizing changes in surface curvature, as well as two data terms $\mathcal{E}_{model\to data}$ and $\mathcal{E}_{data\to model}$ that align the mesh model to the observed data and the data to the model, respectively. The impact of the smoothness term and the data terms is steered by the parameter γ_{def}.

For the data terms, we use the same terms that are used for articulated hand tracking in [26]. For the first term

$$\mathcal{E}_{model\to data}(\mathcal{M}, D) = \sum_i \|\mathbf{V}_i - \mathbf{X}_i\|_2^2 \tag{2}$$

we establish correspondences between the visible vertices $\mathbf{V}_i$ of the mesh $\mathcal{M}$ and the closest points $\mathbf{X}_i$ of the point cloud D and minimize the distance. We discard correspondences for which the angle between the normals of the vertex and the closest point is larger than 45° or the distance between the points is larger than 10 mm.

The second data term

$$\mathcal{E}_{data\to model}(\mathcal{M}, D) = \sum_i \|\mathbf{V}_i \times \mathbf{d}_i - \mathbf{m}_i\|_2^2 \tag{3}$$

minimizes the distance between a vertex $\mathbf{V}_i$ and the projection ray of a depth discontinuity observed in the depth image. To compute the distance, the projection ray of a 2D point is expressed by a Plücker line [27] with direction $\mathbf{d}_i$ and moment $\mathbf{m}_i$. The depth discontinuities are obtained as in [26] by an edge detector applied to the depth data and the correspondence between a depth discontinuity and a vertex are obtained by searching the closest projected vertex for each depth discontinuity.

Due to the partial view of the object, the data terms are highly underconstrained. This is compensated by the smoothness term that penalizes changes of the surface curvature [24]. The term can be written as

$$\mathcal{E}_{smooth}(\mathcal{M}) = \sum_i \|\mathbf{L}\mathbf{V}_i - \mathbf{L}\mathbf{V}_{i,t-1}\|_2^2 \tag{4}$$

where $\mathbf{V}_{i,t-1}$ is the previous vertex position. In order to model the surface curvature, we employ the cotangent Laplacian [24] matrix $\mathbf{L}$ given by

$$L_{ij} = \begin{cases} \sum_{\mathbf{V}_k \in \mathcal{N}_1(\mathbf{V}_i)} w_{ik}, & i = j \\ -w_{ij}, & \mathbf{V}_j \in \mathcal{N}_1(\mathbf{V}_i) \\ 0, & \text{otherwise,} \end{cases} \quad \text{where} \quad w_{ij} = \frac{1}{2|A_i|}(\cot\alpha_{ij} + \cot\beta_{ij}) \tag{5}$$

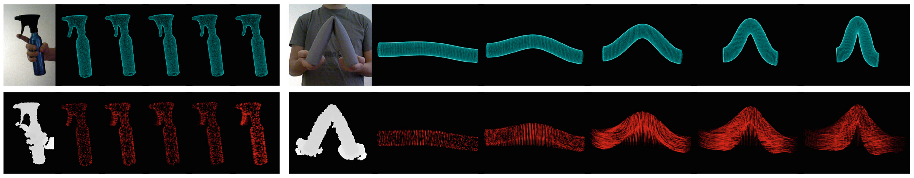


Fig. 1. Tracked mesh with the deformable tracker presented in Sect. 3.2 and the corresponding 3D vertex trajectories. We present images for the sequences "spray" and "pipe 1/2" showing the temporal evolution at 20 %, 40 %, 60 %, 80 % and 100 % of the sequence.

where $\mathcal{N}_1(\mathbf{V}_i)$ denotes the set of one-ring neighbor vertices of vertex $\mathbf{V}_i$. The weight w_{ij} for an edge in the triangular mesh between two vertices $\mathbf{V}_i$ and $\mathbf{V}_j$ depends on the cotangents of the two angles α_{ij} and β_{ij} opposite of the edge (i, j) and the size of the Voronoi cell $|A_i|$ that is efficiently approximated by half of the sum of the triangle areas defined by $\mathcal{N}_1(\mathbf{V}_i)$.

We minimize the least squares problem (1) by solving a large but highly sparse linear system using sparse Cholesky decomposition. For each frame, we use the estimate of the previous frame for initialization and iterate between computing correspondences and optimizing (1) 15 times.

4 Kinematic Model Acquisition

After having estimated the mesh motion as described in Sect. 3, we have for each vertex the trajectory $\mathcal{T}_i$. We use the trajectories together with the shape of the mesh $\mathcal{M}$ to reconstruct the underlying skeleton. To this end, we first segment the trajectories as described in Sect. 4.1 and then infer the skeleton structure, which will be explained in Sect. 4.2.

4.1 Motion Segmentation

In contrast to feature based trajectories, the mesh motion provides trajectories of the same length and a trajectory for each vertex, even if the vertex has never been observed in the sequence due to occlusions. This means that clustering the trajectories also segments the mesh into rigid parts.

Similar to 2D motion segmentation approaches for RGB videos [28], we define an affinity matrix based on the 3D trajectories and use spectral clustering for motion segmentation. The affinity matrix

$$\Phi_{ij} = \exp\left(-\lambda d(\mathcal{T}_i, \mathcal{T}_j)\right) \tag{6}$$

is based on the pairwise distance between two trajectories $\mathcal{T}_i$ and $\mathcal{T}_j$. $\Phi_{ij} = 1$ if the trajectories are the same and close to zero if the trajectories are very dissimilar. As in [28], we use $\lambda = 0.1$.

To measure the distance between two trajectories $\mathcal{T}_i$ and $\mathcal{T}_j$, we measure the distance change of two vertex positions $\mathbf{V}_i$ and $\mathbf{V}_j$ within a fixed time interval. We set the length of the time interval proportional to the observed maximum displacement, i.e.

$$dt = 2 \max_{i,t} \|\mathbf{V}_{i,t} - \mathbf{V}_{i,t-1}\|_2. \tag{7}$$

Since the trajectories are smooth due to the mesh tracking as described in Sect. 3.2, we do not have to deal with outliers and we can take the maximum displacement over all vertices. The object, however, might be deformed only at a certain time interval of the entire sequence. We are therefore only interested in the maximum distance change over all time intervals, i.e.

$$d^v(\mathcal{T}_i, \mathcal{T}_j) = \max_t \left| \|\mathbf{V}_{i,t} - \mathbf{V}_{j,t}\|_2 - \|\mathbf{V}_{i,t-dt} - \mathbf{V}_{j,t-dt}\|_2 \right|. \tag{8}$$

This means that if two vertices belong to the same rigid part, the distance between them should not change much over time. In addition, we take the change of the angle between the vertex normals $\mathbf{N}$ into account. This is measured in the same way as maximum over the intervals

$$d^n(\mathcal{T}_i, \mathcal{T}_j) = \max_t \left| \arccos\left(\mathbf{N}_{i,t}^T \mathbf{N}_{j,t}\right) - \arccos\left(\mathbf{N}_{i,t-dt}^T \mathbf{N}_{j,t-dt}\right) \right|. \tag{9}$$

The two distance measures are combined by

$$d(\mathcal{T}_i, \mathcal{T}_j) = (1 + d^n(\mathcal{T}_i, \mathcal{T}_j))\, d^v(\mathcal{T}_i, \mathcal{T}_j). \tag{10}$$

The distances are measured in mm and the angles in rad. Adding 1 to d^n was necessary since d^n can be close to zero despite of large displacement changes.

Based on (6), we build the normalized Laplacian graph [29]

$$\mathcal{L} = D^{-\frac{1}{2}}(D - \Phi)D^{-\frac{1}{2}} \tag{11}$$

where D is an $n \times n$ diagonal matrix with

$$D_{ii} = \sum_j \Phi_{ij} \tag{12}$$

and perform eigenvalue decomposition of $\mathcal{L}$ to get the eigenvalues $\lambda_1, \ldots, \lambda_n$, $(\lambda_1 \leq \cdots \leq \lambda_n)$, as well as the corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. The number of clusters k is determined by the number of eigenvalues below a threshold λ_{thresh} and the final clustering of the trajectories is then obtained by k-means clustering [29] on the rows of the $n \times k$ matrix $\mathcal{F} = [\mathbf{v}_1 \ \ldots \ \mathbf{v}_k]$.

In practice, we sample uniformly 1000 vertices from the mesh to compute the affinity matrix. This turned out to be sufficient while reducing the time to compute the matrix. For each vertex that has not been sampled, we compute the closest sampled vertex on the mesh and assign it to the same cluster. This results in a motion segmentation of the entire mesh as shown in Fig. 2b.

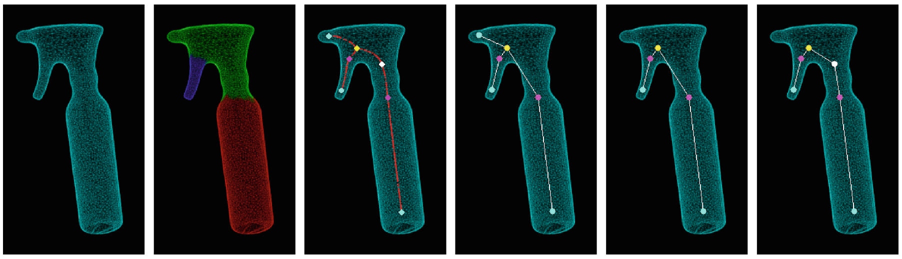


Fig. 2. The steps of our pipeline. (a) *Initial mesh* (b) *Motion segments* (c) *Mean curvature skeleton* where the endpoints are shown with cyan, the junction points with yellow, the virtual point due to collision with white and the motion joints with magenta (d) *Initial skeleton* (e) *Refined skeleton* after removal of redundant bone (f) *Final skeleton* after replacement of the colliding bone with two non-colliding ones and a virtual joint.

Algorithm 1. Overview of the steps of our algorithm.

```
Deformable motion capture
  - Perform deformable tracking of the object                  Sect. 3.2 - Eq. (1)
Motion segmentation of the object
  - Generate dense vertex trajectories from tracking result    Sect. 4.1
  - Sample 1000 trajectories for tractability                  Sect. 4.1
  - Build an affinity matrix of vertex trajectories            Sect. 4.1 - Eq. (6–10)
  - Segment mesh by spectral clustering                        Sect. 4.1 - Eq. (11)
Kinematic model acquisition for the object
  - Infer joints at intersections of mesh segments             Sect. 4.2
  - Infer skeleton topology                                    Sect. 4.2
  - Compute skinning weights                                   Sect. 4.2
```

4.2 Kinematic Topology

Given the segmented mesh, it remains to determine the joint positions and topology of the skeleton. To obtain a bone structure, we first skeletonize the mesh by extracting the mean curvature skeleton (MCS) based on the mean curvature flow [30] that captures effectively the topology of the mesh by iteratively contracting the triangulated surface. The red 3D curve in Fig. 2c shows the mean curvature skeleton for an object. In order to localize the joints, we compute the intersecting boundary of two connected mesh segments using a half-edge representation. For each intersecting pair of segments, we compute the centroid of the boundary vertices and find its closest 3D point on the mean curvature skeleton. In this way, the joints are guaranteed to lie inside the mesh. In order to create the skeleton structure with bones, we first create auxiliary joints without any degree of freedom at the points where the mean curvature skeleton branches or ends as shown in Fig. 2c. After all 3D joints on the skeleton are determined, we follow the mean curvature skeleton and connect the detected joints accordingly to build a hierarchy of bones that defines the topology of a skeleton structure.

Although the number of auxiliary joints usually does not matter, we reduce the number of auxiliary joints and irrelevant bones by removing bones that link an endpoint with another auxiliary joint if they belong to the same motion segment. The corresponding motion segment for each joint can be directly computed from the mean curvature flow [30]. We finally ensure that each bone is inside the mesh. To this end, we detect bones colliding with the mesh with a collision detection approach based on bounding volume hierarchies. We then subdivide each colliding bone in two bones by adding an additional auxiliary joint at the middle of the mean curvature skeleton that connects the endpoints of the colliding bone. The process is repeated until all bones are inside the mesh. In our experiments, however, one iteration was enough. This procedure defines the refined topology of the skeleton that is already embedded in the mesh. The skinning weights are then computed as in [3].

As a result, we obtain a fully rigged model consisting of a watertight mesh, an embedded skeleton structure, and skinning weights. The entire steps of the approach are summarized in Algorithm 1. Results for a few objects are shown in Fig. 5.

5 Experiments

We quantitatively evaluate our approach for five different objects shown in Table 1: the "spray", the "donkey", the "lamp", as well as the "pipe 1/2" and "pipe 3/4" which have a joint at 1/2 and 3/4 of their length, respectively. We acquire a 3D template mesh using the commercial software *skanect* [19] for the first three objects, while for the pipe we use the publicly available template model used in [26]. All objects have the same number of triangles, so the average triangle size varies from 3.7 mm^2 for the "spray", 13.8 for the "donkey", 24.8 for the "lamp" and 4.4 for the "pipe" models. We captured sequences of the objects while deforming them using a Primesense Carmine 1.09 RGB-D sensor. The recorded sequences, calibration data, scanned 3D models, deformable motion data, as well as the resulting models and respective videos for the proposed parameters are available online[1].

We perform deformable tracking (Sect. 3.2) to get 3D dense vertex trajectories as depicted in Fig. 1. Deformable tracking depends on the weight γ_{def} in the objective function (1) that steers the influence of the smoothness and data terms. As depicted in Fig. 4, a very low γ_{def} gives too much weight to the smoothness term and prevents an accurate fitting to the input data, while a big γ_{def} results in over-fitting to the partial visible data and a strong thinning effect can be observed. The thinning gets more intense for an increasing γ_{def}.

Despite of γ_{def}, our approach also depends on the eigenvalue threshold λ_{thr} for spectral clustering. To study the effect of the parameters, we created a test dataset. For each object, we scanned the objects in four different poses. To this end, we fixed the object in a pose with adhesive tape and reconstructed it by moving the camera around the object. The target poses of the objects are shown

[1] http://files.is.tue.mpg.de/dtzionas/Skeleton-Reconstruction.

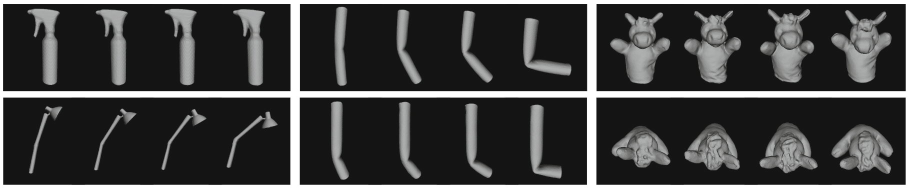


Fig. 3. Each object is scanned in four target poses with increasing difficulty and pose estimation from an initial state is performed for evaluation while spanning the parameter space of $(\gamma_{def}, \lambda_{thresh})$. For the "donkey" object both a front and a top view are presented.

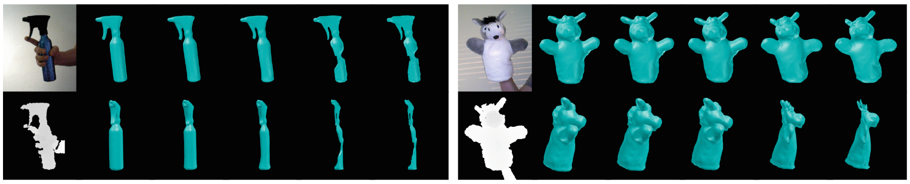


Fig. 4. Deformable tracking for $\gamma_{def} = 0.001,\ 0.005,\ 0.01,\ 0.05,\ 0.1$ (from left to right) that steers the influence of the smoothness and data terms in Eq. (1). We depict the front (top) and side view (bottom) for the last frame of the sequences "spray" and "donkey".

in Fig. 3. To measure the quality of a rigged model for a parameter setting, we align the model $\mathcal{M}(\theta)$ parametrized by the rotations of the joints and the global rigid transformation to the reconstructed object $\mathcal{O}$ from an initial pose. For the alignment, we use only the inferred articulated model, i.e. we estimate the rigid transformation and the rotations of the joints of the inferred skeleton. As data term, we use

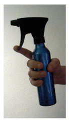
$$\frac{1}{|\mathcal{M}(\theta)| + |\mathcal{O}|} \left(\sum_{\mathbf{V}(\theta) \in \mathcal{M}(\theta)} \|\mathbf{V}(\theta) - \mathbf{V}_{\mathcal{O}}\|_2^2 + \sum_{\mathbf{V}_{\mathcal{O}} \in \mathcal{O}} \|\mathbf{V}_{\mathcal{O}} - \mathbf{V}(\theta)\|_2^2 \right) \tag{13}$$

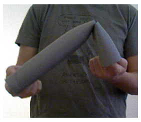
based on the closest vertices from mesh $\mathcal{M}(\theta)$ to $\mathcal{O}$ and vice versa. This measure is also used to measure the 3D error in mm after alignment.

Table 1 summarizes the average 3D vertex error for various parameter settings, with the highlighted values indicating the best qualitative results for each object, while Fig. 5 shows the motion segments and the acquired skeletons for the best configuration. The optimal parameter γ_{def} seems to depend on the triangle size since the smoothness term is influenced by the areas of the Voronoi cells $|A_i|$ (5) and therefore by the areas of the triangles. The objects "Donkey" and "Lamp" have *large triangles* ($>$10 mm^2) and prefer $\gamma_{def} = 0.05$, while the objects with small triangles ($<$10 mm^2) perform better for $\gamma_{def} = 0.005$. Spectral clustering on the other hand works well for $\lambda_{thr} = 0.7$ when *reasonably sized parts* undergo a *pronounced movement*, however, a higher value of $\lambda_{thr} = 0.98$ is better

Table 1. Evaluation of our approach using the target poses shown in Fig. 3. We create a rigged model while spanning the parameter space for the deformable tracking weight γ_{def} and the spectral clustering threshold λ_{thr}. The rigged model is aligned to the target poses by articulated pose estimation. We report the average vertex error in mm.

	γ_{def} \ λ_{thr}	0.40	0.50	0.60	0.70	0.80	0.90	0.95	0.98
Spray	0.001	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	0.005	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.4
	0.01	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.4
	0.05	1.9	1.9	1.9	1.9	1.9	1.9	1.5	1.5
	0.1	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
Pipe 1/2	0.001	10.0	2.4	2.4	2.4	4.5	3.4	3.3	3.6
	0.005	2.4	2.4	2.4	2.4	2.4	4.6	3.8	2.6
	0.01	2.7	2.7	2.7	4.7	3.4	3.7	4.3	4.4
	0.05	2.6	2.6	3.5	2.7	3.6	3.6	3.6	3.6
	0.1	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
Pipe 3/4	0.001	8.3	5.1	5.1	5.1	2.5	3.0	2.8	2.4
	0.005	2.4	2.4	2.4	2.4	3.6	2.5	2.6	2.4
	0.01	2.4	2.4	2.4	2.4	2.8	2.4	2.4	2.4
	0.05	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3
	0.1	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3
Donkey	0.001	6.7	6.7	6.7	6.7	6.7	6.7	6.7	6.7
	0.005	6.7	6.7	6.7	6.7	6.7	6.7	6.7	5.7
	0.01	6.7	6.7	6.7	6.7	5.8	5.8	4.8	4.1
	0.05	4.6	5.1	5.0	4.5	4.4	3.9	3.6	3.6
	0.1	6.3	5.1	5.0	5.1	3.8	4.0	4.0	4.0
Lamp	0.001	12.9	12.9	12.9	12.9	12.9	12.9	11.8	11.8
	0.005	8.2	6.1	6.0	4.7	5.1	4.9	4.6	4.6
	0.01	6.0	6.0	4.6	5.0	5.0	4.7	4.7	4.6
	0.05	11.8	4.7	4.7	4.7	4.7	4.7	5.2	4.8
	0.1	12.6	12.8	5.2	5.3	4.7	4.7	4.6	4.6





for *small parts* undergoing a *small motion* compared to the size of the object like the handle of the "spray". As shown in Fig. 6, a high threshold results in an over-segmentation and increases the number of joints. An over-segmentation is often acceptable as we see for example in Fig. 2b or in Fig. 5 for the "spray" and the "lamp". In general, a slight over-segmentation is not problematic for many applications since joints can be disabled or ignored for instance for animation.

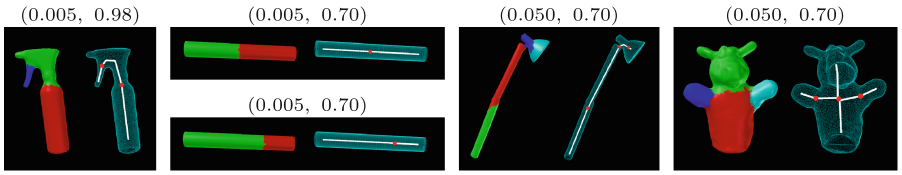


Fig. 5. Results for the best configuration (γ_{def}, λ_{thr}) for each object. The images show the motion segments and the inferred 3D skeleton, where the joints with DoF are depicted with red color. (Color figure online)

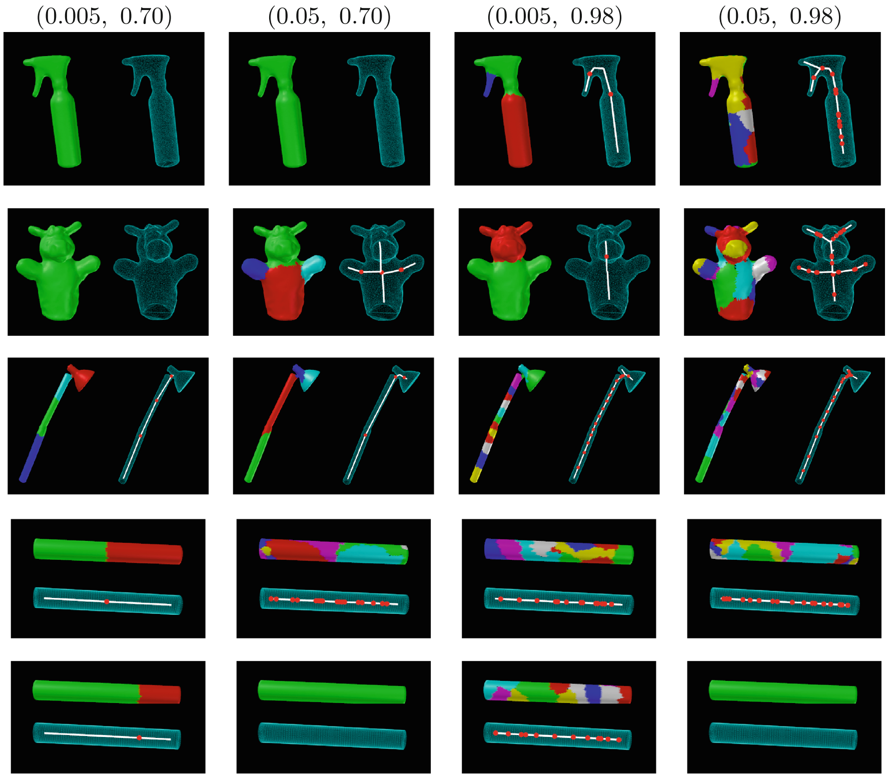


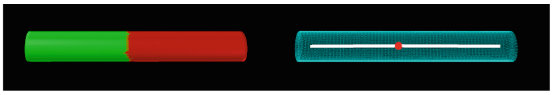
Fig. 6. Results for the four configurations (γ_{def}, λ_{thr}) that arise from the proposed parameters. The images show for each object the motion segments and the inferred 3D skeleton, where the joints with DoF are depicted with red color. (Color figure online)

A slight increase of the degrees of freedom also does not slow down articulated pose estimation, it even yields sometimes a lower alignment error as shown in Table 1.

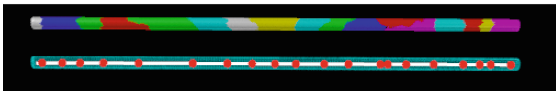
We also evaluated our method on the public sequences *"Bending a Pipe"* and *"Bending a Rope"* of [26], in which the skeleton was manually modeled with 1 and 35 joints, respectively. As input we use the provided mesh of each object and the RGB-D sequences to infer the skeleton. We use the tracked object meshes of [26] as ground-truth and measure the error as in (13), but averaged over all frames. We first evaluate the accuracy of the deformable tracking in Table 2, which performs best with $\gamma_{def} = 0.005$ as in the previous experiments. If we track the sequence with the inferred articulated model using a point-to-plane metric as in [26], the error decreases. While the best spectral clustering threshold λ_{thr} for the pipe is again 0.70, the rope performs best for 0.98 due to the small size of the motion segments and the smaller motion differences of neighboring segments. We also report the error when the affinity matrix is computed only based on d^v without d^n (10). This slightly increases the error for the pipe with

Table 2. Evaluation of our method and resulting kinematic models for the public sequences "Bending a Pipe" and "Bending a Rope" of [26]. We report the average vertex error in mm.

	λ_{thr} / γ_{def}	0.70	0.98	0.70	0.98	
Pipe	0.005	2.6	26.7	2.9	22.1	4.5
Pipe	0.05	12.6	12.6	12.7	12.7	15.9
		articulated with d^n		articulated without d^n		deform.



	λ_{thr} / γ_{def}	0.70	0.98	0.70	0.98	
Rope	0.005	2.5	1.1	2.4	1.1	2.6
Rope	0.05	141.0	141.0	193.8	193.8	nan
		articulated with d^n		articulated without d^n		deform.



optimal parameters. The motion segments and the acquired skeletons for the best configurations are also depicted in Table 2.

6 Conclusion

We presented an approach that generates fully rigged models consisting of a watertight mesh, an embedded skeleton and skinning weights that can be used out of the box for articulated tracking or animation. In that respect we operate fully in 3D capitalizing on deformable tracking, spectral clustering and skeletonization based on mean curvature flow. The thorough evaluation of the parameters provides a valuable intuition about the important factors and opens up possibilities for further generalization in future work. For instance, a regularizer that is adaptive to the areas of the triangles can be used for deformable tracking to compensate seamlessly for the varying triangle sizes across different objects. Furthermore, we have shown in our experiments that the proposed approach generates nicely working rigged models and has prospects for future practical applications.

Acknowledgements. The authors acknowledge financial support by the DFG Emmy Noether program (GA 1927/1-1).

References

1. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: real-time dense surface mapping and tracking. In: International Symposium on Mixed and Augmented Reality (ISMAR) (2011)
2. Sturm, J., Bylow, E., Kahl, F., Cremers, D.: Copyme3d: Scanning and printing persons in 3d. In: German Conference on Pattern Recognition (GCPR) (2013)
3. Baran, I., Popović, J.: Automatic rigging and animation of 3d characters. ACM Trans. Graph. (TOG) **26**(3), 72 (2007)
4. Pillai, S., Walter, M.R., Teller, S.: Learning articulated motions from visual demonstration. In: Robotics: Science and Systems (RSS) (2014)
5. Stoll, C., Gall, J., de Aguiar, E., Thrun, S., Theobalt, C.: Video-based reconstruction of animatable human characters. ACM Trans. Graph. (TOG) **29**(6), 139: 1–139: 10 (2010)
6. De Aguiar, E., Theobalt, C., Thrun, S., Seidel, H.P.: Automatic conversion of mesh animations into skeleton-based animations. Comput. Graph. Forum (CGF) **27**(2), 389–397 (2008)
7. Liu, Y., Gall, J., Stoll, C., Dai, Q., Seidel, H.P., Theobalt, C.: Markerless motion capture of multiple characters using multiview image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) **35**(11), 2720–2735 (2013)
8. Yan, J., Pollefeys, M.: Automatic kinematic chain building from feature trajectories of articulated objects. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2006)
9. Yan, J., Pollefeys, M.: A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) **30**(5), 865–877 (2008)
10. Ross, D.A., Tarlow, D., Zemel, R.S.: Learning articulated structure and motion. Int. J. Comput. Vis. (IJCV) **88**(2), 214–237 (2010)
11. Chang, H.J., Demiris, Y.: Unsupervised learning of complex articulated kinematic structures combining motion and skeleton information. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
12. Fayad, J., Russell, C., Agapito, L.: Automated articulated structure and 3d shape recovery from point correspondences. In: International Conference on Computer Vision (ICCV) (2011)
13. Sturm, J., Pradeep, V., Stachniss, C., Plagemann, C., Konolige, K., Burgard, W.: Learning kinematic models for articulated objects. In: International Joint Conference on Artificial Intelligence (IJCAI) (2009)
14. Sturm, J., Stachniss, C., Burgard, W.: A probabilistic framework for learning kinematic models of articulated objects. J. Artif. Intell. Res. (JAIR) **41**(2), 477–626 (2011)
15. Yücer, K., Wang, O., Sorkine-Hornung, A., Sorkine-Hornung, O.: Reconstruction of articulated objects from a moving camera. In: ICCVW (2015)
16. Katz, D., Kazemi, M., Bagnell, A.J., Stentz, A.: Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects. In: IEEE International Conference on Robotics and Automation (ICRA) (2013)
17. Martín-Martín, R., Höfer, S., Brock, O.: An integrated approach to visual perception of articulated objects. In: IEEE International Conference on Robotics and Automation (ICRA) (2016)

18. Tresadern, P., Reid, I.: Articulated structure from motion by factorization. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
19. Skanect: http://skanect.occipital.com. Accessed 19 Aug 2016
20. MeshLab: http://meshlab.sourceforge.net. Accessed 19 Aug 2016
21. Jones, M.J., Rehg, J.M.: Statistical color models with application to skin detection. Int. J. Comput. Vis. (IJCV) **46**(1), 81–96 (2002)
22. Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. Int. J. Comput. Vis. (IJCV) **81**(1), 24–52 (2009)
23. Holzer, S., Rusu, R.B., Dixon, M., Gedikli, S., Navab, N.: Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2012)
24. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Trans. Vis. Comput. Graph. (TVCG) **14**(1), 213–230 (2008)
25. Gall, J., Stoll, C., De Aguiar, E., Theobalt, C., Rosenhahn, B., Seidel, H.P.: Motion capture using joint skeleton tracking and surface estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
26. Tzionas, D., Ballan, L., Srikantha, A., Aponte, P., Pollefeys, M., Gall, J.: Capturing hands in action using discriminative salient points and physics simulation. Int. J. Comput. Vis. (IJCV) **118**, 172–193 (2016)
27. Pons-Moll, G., Rosenhahn, B.: Model-based pose estimation. In: Moeslund, T.B., Hilton, A., Krüger, V., Sigal, L. (eds.) Visual Analysis of Humans: Looking at People, pp. 139–170. Springer, London (2011). doi:10.1007/978-0-85729-997-0_9
28. Brox, T., Malik, J.: Object segmentation by long term analysis of point trajectories. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6315, pp. 282–295. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15555-0_21
29. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: analysis and an algorithm. In: Advances in Neural Information Processing Systems NIPS (2002)
30. Tagliasacchi, A., Alhashim, I., Olson, M., Zhang, H.: Mean curvature skeletons. Comput. Graph. Forum (CGF) **31**, 1735–1744 (2012)