Reconstruction of Articulated Objects from a Moving Camera

Kaan Yücer[1,2] Oliver Wang[2] Alexander Sorkine-Hornung[2] Olga Sorkine-Hornung[1]

[1]ETH Zurich [2]Disney Research

Abstract

Many scenes that we would like to reconstruct contain articulated objects, and are often captured by only a single, non-fixed camera. Existing techniques for reconstructing articulated objects either require templates, which can be challenging to acquire, or have difficulties with perspective effects and missing data. In this paper, we present a novel reconstruction pipeline that first treats each feature point tracked on the object independently and incrementally imposes constraints. We make use of the idea that the unknown 3D trajectory of a point tracked in 2D should lie on a manifold that is described by the camera rays going through the tracked 2D positions. We compute an initial reconstruction by solving for latent 3D trajectories that maximize temporal smoothness on these manifolds. We then leverage these 3D estimates to automatically segment an object into piecewise rigid parts, and compute a refined shape and motion using sparse bundle adjustment. Finally, we apply kinematic constraints on automatically computed joint positions to enforce connectivity between different rigid parts, which further reduces ambiguous motion and increases reconstruction accuracy. Each step of our pipeline enforces temporal smoothness, and together results in a high quality articulated object reconstruction. We show the usefulness of our approach in both synthetic and real datasets and compare against other non-rigid reconstruction techniques.

1. Introduction

Reconstructing a static 3D scene from image sequences has been an important research question for several decades. In particular, structure from motion (SfM) [17] techniques have been successfully used in a wide area of different applications such as localization, navigation, and image based modeling, up to reconstructing entire cities from unstructured image collections [2].

Deforming object reconstruction, on the other hand, is a widely studied, but still unsolved problem. Many deforming objects, such as humans, animals and most human-made machines, move in an articulated way, *i.e*. they can be approximated by a set of piecewise rigid parts connected by joints. This has spawned a lot of interest in the task of articulated structure from motion (A-SfM).

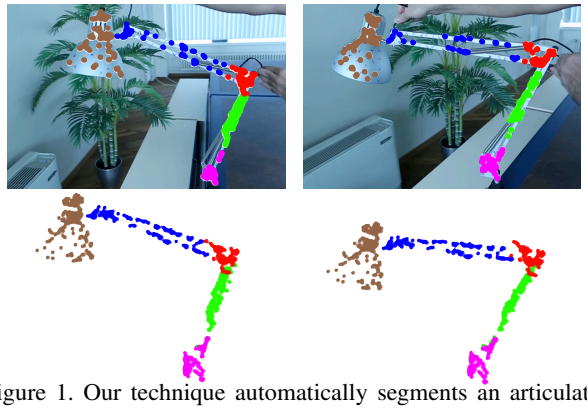

Figure 1. Our technique automatically segments an articulated object into its piecewise rigid parts (top), which are used to infer a 3D articulated model and its motion at each frame (bottom).

One general solution to A-SfM is to use shape templates and deform them to match the observed images [15]. However, these methods require detailed template and motion models and cannot be easily generalized. A more generic approach is to track 2D points over a video, segment them into articulated parts and apply factorization methods [14, 38]. In order to segment the input trajectories, such methods generally minimize an energy function, usually exploiting point connectivity in 2D and modeling the object shape in 2.5D or 3D. Estimating shape models using 2D track connectivity is cumbersome due to inherent ambiguities, and if good models cannot be estimated from point tracks, the result quality degrades. Moreover, projective cameras cannot be modeled easily, and partial point tracks require special attention.

Most proposed techniques assume that the cameras are not calibrated and optimize directly for the objects shape and motion. However, most real world scenes contain enough static rigid components in the background to calibrate the cameras using SfM techniques [1]. In this work, we make use of calibrated cameras and propose a pipeline that initially treats every 2D point track independently and incrementally imposes rigidity and articulation constraints. Camera calibrations help us formulate the problem using perspective cameras whereas previous work uses orthographic cameras.

In the first step, we show how 2D point tracks can be directly used to infer 3D information by a trajectory smoothness assumption without the need for a dual 2D-3D representation. By transforming the problem domain, we can model both object shape and point connectivity in a unified 3D framework. In a second step, we automatically segment the 3D trajectories into piecewise rigid components using a metric that depends on their relative positions along the 3D trajectories. After segmentation, shapes and motions of these piecewise rigid components can be computed using nonlinear optimization. We then automatically estimate a skeleton for the object and the corresponding joint positions between the segments, and finally constrain the optimization using joint positions to ensure that connected components remain connected after reconstruction (See Figure 1). This incremental technique has the advantage of gradually adding reliable information to approach the actual solution. This way, we can avoid computing a fully unconstrained optimization of all parameters, which is prone to local minima.

As input, our system takes a video of a dynamic articulated object. We require the object to have enough texture for computing point correspondences between frames. We assume that the background contains enough static parts for SfM to compute camera calibrations, which enables working with perspective camera models. Also, we require the camera to move at a speed that is similar in magnitude to object motion to compute its 3D shape and motion over time. These requirements will be explained in Section 3.

Our specific contributions over previous works are (i) a 3D trajectory estimation from 2D tracks based on a smooth manifold assumption that does not restrict motion to more limited subspaces, (ii) an automatic segmentation of the trajectories into piecewise rigid components without the need for ambiguous 3D motion models, and (iii) a new nonlinear optimization of kinematic constraints that connects the piecewise rigid components into a single articulated model which is consistent with the observations in the input images.

2. Related Work

Template-based reconstruction. If a shape template of the non-rigid object is available a priori, a full 3D reconstruction can be computed by deforming the template using motion priors to match the current observation. This has been demonstrated to work for estimating shapes of faces [7], sheet-like inextensible surfaces [27] and poses of human bodies [6, 15]. In our approach, we address the reconstruction of generic articulated objects without the requirement of shape priors.

Non-rigid structure from motion (NRSfM). A more generic approach is to directly reconstruct objects using tracked points over a video or a photo collection. The main idea is that the object motion is limited to a low dimensional subspace. Factorization techniques have been used to recover the shape and motion of a single rigid object [33], and of deformable non-rigid objects [8, 9] by assuming that the object shape is a linear combination of some basis shapes.

This factorization problem is under-constrained and the non-rigid shape bases and coefficients cannot be recovered uniquely. Different approaches have been proposed to resolve this ambiguity, e.g., using semi-definite programming without the need for priors [12], using basis constraints [37], 3D shape prior models followed by iterative non-linear optimization [10,11], or Gaussian priors on the object shape [34]. Hartley and Vidal [18] show that the NRSfM problem has a closed form solution for perspective cameras. However, all these methods use global shape models and quickly become unstable in unconstrained, real-world acquisition scenarios.

We do not use 2D point tracks to compute the final shape directly, but make use of calibrated cameras to lift the problem into 3D and solve it in this higher dimensional space. Hence, we do not require any object shape or motion priors.

Other techniques [13,29,36] do not use global shape models and assume that the object consists of piecewise local 3D patches. However, they only reconstruct sheet-like deformable shapes and do not consider articulation constraints.

Articulated object reconstruction. Techniques that specifically reconstruct articulated objects usually first segment the object into its piecewise rigid components and then apply factorization to compute the actual 3D shape. In order to segment the object, 3D motion models are computed from the 2D point tracks and the tracks are assigned to these models.

Seminal works in that area [35, 39] couple articulated parts via joints and apply hierarchical factorization on this structure with articulation constraints. Segmentation is done via a RANSAC approach [35]. Paladini et al. [22] improve these methods by applying iterative factorization. Yan and Pollefeys [38] first compute a set of linear motion subspaces from 2D tracks locally and apply spectral clustering using an affinity matrix, which describes how well a track fits into a model. However, this method is sensitive to noise in the point tracks. Fayad et al. [14] segment the point tracks into "overlapping models", where a point track can belong to multiple models, to understand the boundaries between articulated parts better. This work has been extended with additional energy functionals [30] to make the segmentation process more robust. Since the neighborhoods of point tracks are estimated in 2D, computing good 3D shape models is difficult and often leads to misclassifications during segmentation. These methods also require long point tracks.

In contrast, we first lift the problem to 3D and compute 3D trajectories from 2D point tracks using a trajectory smoothness assumption. These 3D trajectories can then be segmented easily by simple distance metrics, which helps us avoid the difficulty of computing good prior models from 2D point tracks locally. Moreover, we do not require long point tracks and do not need to deal with missing data explicitly.

Trajectory triangulation. Trajectory triangulation is a common method for reconstructing 3D trajectories from 2D tracking data. First works in this area computed 3D trajectories assuming the points moved on lines and cones [5], on planes [31] and on polynomial 3D curves [20]. However, these methods do not generalize to arbitrary object motions. Akther et al. [4] assume that the point trajectories are a linear combination of discrete cosine transform bases, but they require full visibility of point tracks over all frames. The same bases were then used to describe point trajectories independently [25], however this method required very fast and random camera motions. This work was later extended [24] for reconstructing smooth trajectories for articulated object parts. However, an articulated trajectory can be computed only if the 3D trajectory of the part it is connected is known a priori. Also, manual labeling of point tracks is needed.

In our work, we do not constrain the point trajectories to lie in a specific, and hence limited, motion subspace. Instead, we directly optimize for a smooth 3D trajectory that explains the 2D observations. These trajectories are then used to compute the full articulated shape and motion.

3. Method

Our approach consists of the following steps. As input, we require an image sequence of a moving, piecewise rigid articulated object, with sufficiently many static components in the background for calibration. We first compute camera calibrations for each input view based on the static background elements using [1], and standard 2D point tracks on the image sequence [32]. We then lift the 2D tracks to 3D using a ray-space optimization, resulting in an animated 3D point cloud with trajectories lying on smooth manifolds of camera rays (Section 3.1). The 3D point clouds are then automatically segmented into piecewise rigid components using a weighted graph clustering approach based on distance variations between the 3D trajectories, and the shapes and motions of these rigid components are computed using non-linear energy minimization (Section 3.2). Lastly, we introduce articulation constraints, connecting all rigid parts via virtual joints computed from the data, and apply bundle adjustment to the full articulated object (Section 3.3).

More formally, given a video stream represented by a set of images $\mathbf{I} = \{I_1, ..., I_F\}$ and corresponding calibration matrices $\mathbf{P}_f$, we compute a set of 2D feature points $\mathbf{W} = \bigcup W_f^t$ tracked over I_f with $f \in \{1, ..., F\}$ and $t \in \{1, ..., T\}$, where T is the number of point tracks. The 3D point cloud representing the articulated object over all frames is denoted as $\mathbf{S} = \bigcup S_f^t$, with each 3D point S_f^t corresponding to a 2D feature W_f^t. We incorporate visibility information using the indicator variable V_f^t, which equals 1 if a track t is visible on frame f and 0 otherwise. Hence, we only consider W_f^t with $V_f^t = 1$ and recover only the 3D points in frames where their 2D tracks are visible.

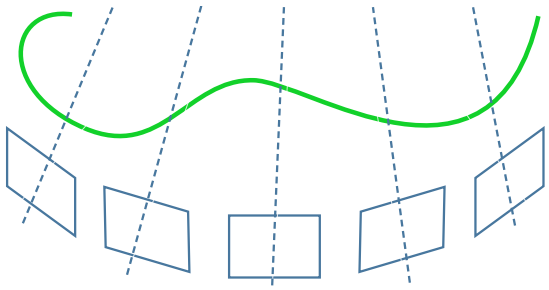

Figure 2. Illustration of our ray space optimization.

3.1. Ray-Space Optimization

The 3D trajectory of a point tracked in 2D is located on the respective camera rays in 3D (see Figure 2). Formally, given a point W_f^t and the camera center C_f of view f, the 3D point S_f^t lies on the following line parametrized by μ_f^t:

$$S_f^t = C_f + \mu_f^t D_f^t, \tag{1}$$

where D_f^t is the direction vector that goes from C_f through W_f^t. All $S_f^t \in \mathbf{S}^t$ for a point track $\mathbf{W}^t$ lie on the manifold described by these rays. There are multiple valid 3D paths that project to the same point track, rendering the problem highly under-constrained. However, since real world objects usually move on smooth trajectories, we make use a temporal coherence assumption between the 3D tracked positions in neighboring frames and optimize the following energy:

$$E_{rs}(\mathbf{S}^t) = \sum_{f=1}^{F-1} \|S_f^t - S_{f+1}^t\|^2. \tag{2}$$

Plugging in Eq. (1) into the above energy results in the following function, which only depends on $\boldsymbol{\mu}^t = \bigcup_{f=1}^{F} \mu_f^t$:

$$E_{rs}(\boldsymbol{\mu}^t) = \sum_{f=1}^{F-1} \|(C_f + \mu_f^t D_f^t) - (C_{f+1} + \mu_{f+1}^t D_{f+1}^t)\|^2 \tag{3}$$

As described in above, for each track $\mathbf{W}^t$, we optimize the above energy only for f where $V_f^t = 1$, *i.e.* only for the frames where the track is visible.

If a point is static, this energy function corresponds to standard triangulation. However, if a point is moving faster than the camera, this energy results in point trajectories which are close to camera centers. In order to alleviate this, we introduce weights ω_f^t that approximate the amount of motion for each tracked point. As the point and camera motion are intertwined, we use the distance from the epipolar line as a means to approximate the point motion. To that end, we compute the epipolar line corresponding to W_f^t in I_{f+1} and measure the distance d_f^t between this line and W_{f+1}^t. These distances are normalized between 0.1 and 1, and ω_f^t is computed as the reciprocal of d_f^t. Thus, our energy becomes:

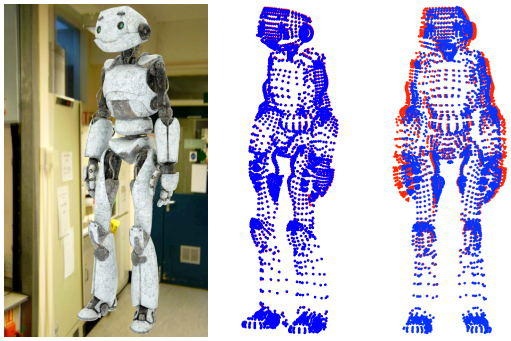

Figure 3. Left: An input image from the Robot sequence. Right: Renderings of the computed point cloud (in blue) and the ground truth data (in red) from two different view points. Note that the reprojection error is 0 due to the ray space optimization.

$$E_{rs}(\boldsymbol{\mu}^t) = \sum_{f=1}^{F-1} \omega_f^t \|(C_f + \mu_f^t D_f^t) - (C_{f+1} + \mu_{f+1}^t D_{f+1}^t)\|^2, \tag{4}$$

which now applies more weight to static parts of the point tracks, thereby keeping the dynamic trajectories in the proximity of the static points. This estimate cannot detect movement along the epipolar axis, in which case it simplifies to Eq. (3). However, we found that due to the soft nature of the weighting scheme, it improves the accuracy of our reconstructions in most practical scenarios.

The energy function above has some nice properties: First of all, it is linear, and each point track is independent, making it very efficient to minimize. Moreover, it does not force the point tracks to lie on any motion subspace, and hence can describe a diverse set of possible movements. By using this optimization framework, we can get very efficient and robust models of 3D shape and motion of the objects, as can be seen in Figure 3 and the supplemental video.

3.2. Piecewise Rigidity Constraints

The ray space optimization results in a valid representation of the actual 3D object. However, point tracks and camera calibrations can be imprecise, causing inaccurate reconstructions, and excessive object motion can lead to distortions (see Section 4). Moreover, at this point we do not have any higher-level information about the articulated structure of the object, but only have individually reconstructed 3D trajectories. To fix these issues, we introduce piecewise rigidity constraints, which improve the results by enforcing that the 3D trajectories belong to a piecewise rigid object. The challenge in computing piecewise rigid reconstructions is often in how to identify the parts that are rigid. Fortunately, our initial reconstruction provides strong cues to segment individual 3D trajectories into rigid components.

Segmentation. We first define a distance metric between 3D trajectories. The underlying intuition is that two points belonging to the same rigid component move similarly, and should be close to each other. For each pair of 3D trajectories, we compute their average distance $d(t, t')$ over the frames where they are both visible, as well as the variance $v(t, t')$ of this distance. Our final distance metric is then a weighted sum: $e(t, t') = d(t, t') + \alpha v(t, t')$.

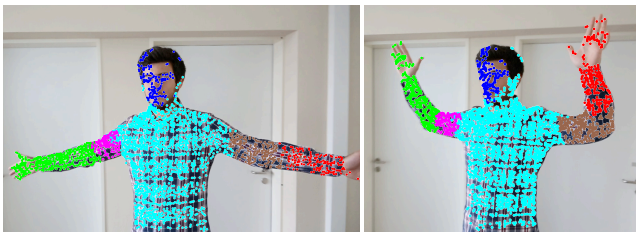

Figure 4. Our 3D trajectories are used to automatically segment the object into its piecewise rigid parts. 6 different segments are identified in the above example.

Using this metric, we build a graph structure where nodes represent trajectories and the edge weights are the affinities between trajectories. Affinities are computed from $e(t, t')$ using a simple normal distribution function $N(0, \sigma)$ where σ is set to half the standard deviation of all distances. We add edges for the 10 closest neighbors of each node in order to maintain a sparse graph for computational efficiency. We then apply recursive two-way spectral clustering on this graph [38], repeatedly subdividing all clusters into two parts until two criteria are reached: 1) all clusters have a lower number of trajectories than T/k (where T is the number of tracks and k is the number of expected clusters) and 2) all clusters have a $\max(e(t, t'))$ lower than a threshold. k is an input, but does not have to be exact. It needs to be close to the actual cluster number for a reliable segmentation. Finally, too small clusters (*i.e.* $T/100$) are attached to the closest clusters. See Figure 4 for segmentation results.

Piecewise rigid reconstruction. The segmentation step results in N objects whose motion can be defined by a rigid transformation. This means that for each piecewise rigid object, we can compute its shape $\boldsymbol{\Omega}^n$ and a transformation at each frame f, consisting of a rotation R_f^n and a translation T_f^n. We describe the point cloud $\mathbf{S}_f$ at each frame as a combination of the objects:

$$\mathbf{S}_f = \bigcup_{n=1}^{N} R_f^n \boldsymbol{\Omega}^n + T_f^n. \tag{5}$$

We are interested in the 3D shape of the objects $\boldsymbol{\Omega}^n$ and their transformations $(R_f^n | T_f^n)$, such that when the 3D points $\mathbf{S}_f$ are projected back to the images I_f, the reprojection error is minimized. This way, we can find the best set of articulated objects that describes the motion of the point tracks observed in the frames. For the point tracks W_f^t and their corresponding 3D position S_f^t, we seek to minimize the following reprojection error:

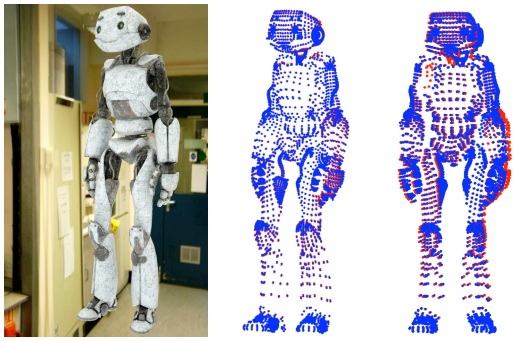

Figure 5. Results of the piecewise rigid reconstruction for the same frame as in Figure 3. The overall reconstruction matches the ground truth more closely, but errors in the chest and arm are still visible.

$$\min_{\mathbf{S}} \sum_{t=1}^{T} \sum_{f=1}^{F} \|W_f^t - \mathbf{P}_f(S_f^t)\|^2. \tag{6}$$

Combining this with Eq. (5) yields:

$$\min_{\mathbf{\Omega},\mathbf{R},\mathbf{T}} \sum_{f=1}^{F} \sum_{n=1}^{N} \|\mathbf{W}_f^n - \mathbf{P}_f(R_f^n \mathbf{\Omega}^n + T_f^n)\|^2, \tag{7}$$

where $\mathbf{W}_f^n$ is the set of the feature points corresponding to the 3D points in $\mathbf{\Omega}^n$, or, more formally, $\mathbf{W}_f^n = \bigcup_{t=1}^{T} W_f^t$ such that $S_f^t \in \mathbf{\Omega}^n$.

This results in a non-linear optimization problem, which we minimize using the CERES framework [3]. Similar to prior steps, this energy is only optimized for (t, f) with $V_f^t = 1$. This step enforces spatial constraints on the point tracks and results in partial objects that move rigidly between different frames (see Figure 5 and the result video), providing robustness to trajectories starting and ending at arbitrary frames. We additionally include a temporal smoothness constraint to force the object parts to have coherent trajectories:

$$\begin{aligned}\min_{\mathbf{\Omega},\mathbf{R},\mathbf{T}} & \sum_{f=1}^{F} \sum_{n=1}^{N} \|\mathbf{W}_f^n - \mathbf{P}_f(R_f^n \mathbf{\Omega}^n + T_f^n)\|^2 \\ & + \lambda \sum_{f=1}^{F-1} \sum_{n=1}^{N} \|(R_f^n \mathbf{\Omega}^n + T_f^n) - (R_{f+1}^n \mathbf{\Omega}^n + T_{f+1}^n)\|^2,\end{aligned} \tag{8}$$

where λ is the temporal smoothness parameter. At each iteration, points with high reprojection errors are marked as outliers and filtered out, similar to standard SfM techniques.

3.3. Kinematic Constraints

The piecewise rigidity constraints generate object parts that move rigidly between frames while adhering to the original point tracks projected back to the input images. However, there is no guarantee yet that parts connected via joints are actually behaving as a kinematic chain in the reconstruction. In order to alleviate this, we introduce articulation constraints. First, we compute a connectivity graph to estimate the underlying skeleton structure, and then optimize the individual components' rigid motions by estimating virtual joints that constrain the motion of connected components.

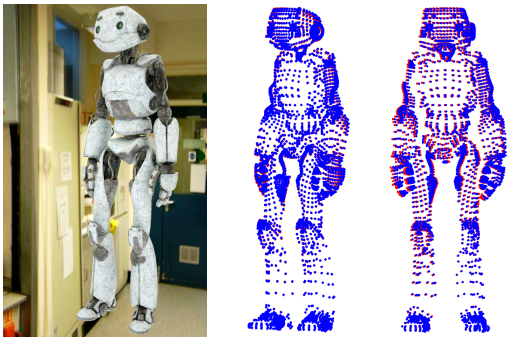

Figure 6. The final results for the same frame as in Figure 3. The errors visible in the preceding steps have been decreased, and the remaining error is more evenly distributed.

Skeleton estimation. We compute a graph where the nodes represent the piecewise rigid components $\mathbf{\Omega}^n$ and the edges describe a distance metric D. Intuitively, we expect the distance between two connected components $\mathbf{\Omega}^n$ and $\mathbf{\Omega}^m$ to be low in general. Therefore we use the mean of the minimum distances between them:

$$D(n,m) = \frac{1}{|\mathbf{F}|} \sum_{f \in \mathbf{F}} \min \|S_f^n - S_f^m\|^2, \tag{9}$$

where S_f^n and S_f^m are the 3D points belonging to $\mathbf{\Omega}^n$ and $\mathbf{\Omega}^m$, respectively. The skeleton can then be estimated as the minimum spanning tree of that graph.

Virtual joint estimation. A joint J can move arbitrarily in 3D space, but remains fixed in the local coordinate systems of two rigid components $\mathbf{\Omega}^n$ and $\mathbf{\Omega}^m$ that are linked via that joint. Additionally, when transforming the respective local joint coordinates J_n and J_m to world coordinates they should coincide:

$$R_f^n J^n + T_f^n = R_f^m J^m + T_f^m. \quad \forall f \in \mathbf{F}'. \tag{10}$$

We can solve for J_n and J_m using linear least squares [19]. In case the two object motions are very similar, the least squares solution will not be robust. We alleviate this problem by constraining J_n and J_m to lie inside the bounding box of $\mathbf{\Omega}^n$ and $\mathbf{\Omega}^m$ scaled by a factor of 1.5. We solve this using a standard constrained linear least squares solver.

Joint aware parametrization. The joints can now be used to restrict the motion of the piecewise-rigid parts. Since we assume no prior knowledge about the object, we treat all estimated virtual joints as universal joints, i.e., the position of one rigid part is constrained with respect to the other, but their rotations are independent. We express the world positions of the objects $\mathbf{\Omega}^n$ in a hierarchical manner:

$$w(n, f, \mathbf{\Omega}^n) = \begin{cases} R_f^n \mathbf{\Omega}^n + T_f^n & \text{if n is root} \\ R_f^n \mathbf{\Omega}^n + w(p(n), f, J_n) & \text{otherwise} \end{cases} \tag{11}$$

where $p(n)$ is the parent of $\mathbf{\Omega}^n$ in the minimum spanning tree. We plug this function into Eq. (7):

$$\min_{\mathbf{\Omega},\mathbf{R},\mathbf{T}} \sum_{f=1}^{F} \sum_{n=1}^{N} \|\mathbf{W}_f^n - \mathbf{P}_f(w(n, f, \mathbf{\Omega}^n))\|^2, \tag{12}$$

and solve it again using CERES. Eq. (12) combines rigidity and kinematics. We also apply temporal smoothness constraints similar to Eq. (8), leading to a global bundle adjustment with all constraints. Outlier rejection is applied as in Section 3.2. Our final result can be seen in Figure 6.

4. Results and Experiments

In the following section we evaluate our method quantitatively on one ground truth synthetic dataset and on several motion capture sequences. We also add qualitative evaluation on several real-world datasets, with various comparisons to state-of-the-art approaches. The result of running the three steps of our algorithm on different datasets are shown in Figure 7. Please also refer to the supplemental video for a more detailed demonstration of the moving 3D trajectories.

Parameters. Our algorithm requires three parameters. The weight α for the segmentation distance metric is set to 0.1, giving more weight to proximal points. The expected number of segments k is chosen close to the actual values: 6 for *Body* and *Person* datasets, 4 for *Lamp* and 14 for the *Robot* and *Skin* data sets. The temporal smoothness weight λ is set to 5. See our supplemental material for experiments with λ.

Comparisons. We compared our method against the following non-rigid reconstruction methods: trajectory space (TRS) [4], metric projection (MP) [23] and column space fitting (CSF) [16] on a synthetic ground truth dataset (See Figure 6). We used the normalized reconstruction error [14] defined as:

$$E_R = \|\mathbf{S} - \mathbf{S}^*\|_F / \|\mathbf{S}^*\|_F, \tag{13}$$

where $\mathbf{S}$ is the reconstructed 3D point cloud, $\mathbf{S}^*$ is the ground truth, and $\|\mathbf{S}\|_F$ corresponds to the Frobenius norm. The results are shown in Table 1.

The above approaches are based on factorization and make use of a low-rank assumption, and not all of them support occluded points. For this reason, we compare only using point tracks that are fully visible in the first 100 frames.

	TRS	MP	CSF	RayS	Rigid	Final
E_R	0.1302	0.0654	0.0712	0.0389	0.0184	0.0092

Table 1. Comparison of the reconstruction error (E_R) to three state-of-the-art methods using the full tracks in the first 100 frames (i.e., occlusion-free tracks) for the synthetic *Robot* dataset.

As factorization based methods do not exploit calibrated cameras but estimate them in their optimizations, we aligned their reconstructions to the ground truth using procrustes analysis independently at each frame, and chose parameters that resulted in the lowest reconstruction error.

These results indicate that perspective effects and articulated motion are challenging for these approaches, since their assumptions about an underlying low-dimensional shape subspace are violated. It should be noted that, unlike previous methods, our approach requires camera calibration matrices. However, this requirement is often satisfied in real-world capture scenarios with static background content, so that off-the-shelf-tools can compute sufficiently accurate calibrations. Exploiting this data enables more robust and accurate results, especially using projective cameras. An additional visual comparison to the method using trajectory-based parameterization [4] is shown in Figure 8.

We also perform a quantitative comparison of our method to two articulated SfM techniques [14, 38]. We use the 'skin' dataset from [26], a motion capture dataset with ground-truth 3D positions, which was also used in [14] to compare to [38]. We rotate a virtual perspective camera (using 5^o per-frame separation, similar to [28]) around the object, project the 3D vertex positions into the virtual cameras, and then use our method to recover the 3D coordinates. It is worth noting that, our method works directly on perspective images, while prior works [14, 38] employ a simpler and less realistic orthographic camera model for the reconstruction.

The dataset contains both full tracks, visible in all frames, and partial tracks, which are visible only for a subset of the frames. Since our method can handle both full and partial tracks, we run our algorithm using all available tracks. Qualitative results can be seen in Figure 9. We again use Eq. (13) to compute the error. Our mean reconstruction error is 5.51% over all 467 tracks, compared to 7.13% for [14] over the same set of tracks. Since [38] can only handle full tracks, only these were used in the reconstruction, and the final error computed over 219 full tracks was 6.15% [14]. Note that for [38], each segment was aligned with the ground truth independently (implying prior knowledge of the object structure), which results in an unrealistically low error value. Our method exhibits a lower reconstruction error and also reconstructs 3D positions for both partial and full tracks.

In order to evaluate the first step (RayS) of our algorithm, we made qualitative and quantitative comparisons with existing non-rigid reconstruction methods. First, we used the point tracks and camera calibrations from [24] and reconstructed 3D point trajectories. As shown in Figure 10, our method yields more visually pleasing results, with faithful reconstructions of the limbs. Moreover, we performed quantitative comparisons using the mocap data of [4], in which objects undergo non-rigid, but non-articulated motions. We used the humanoid objects for our comparison, and again

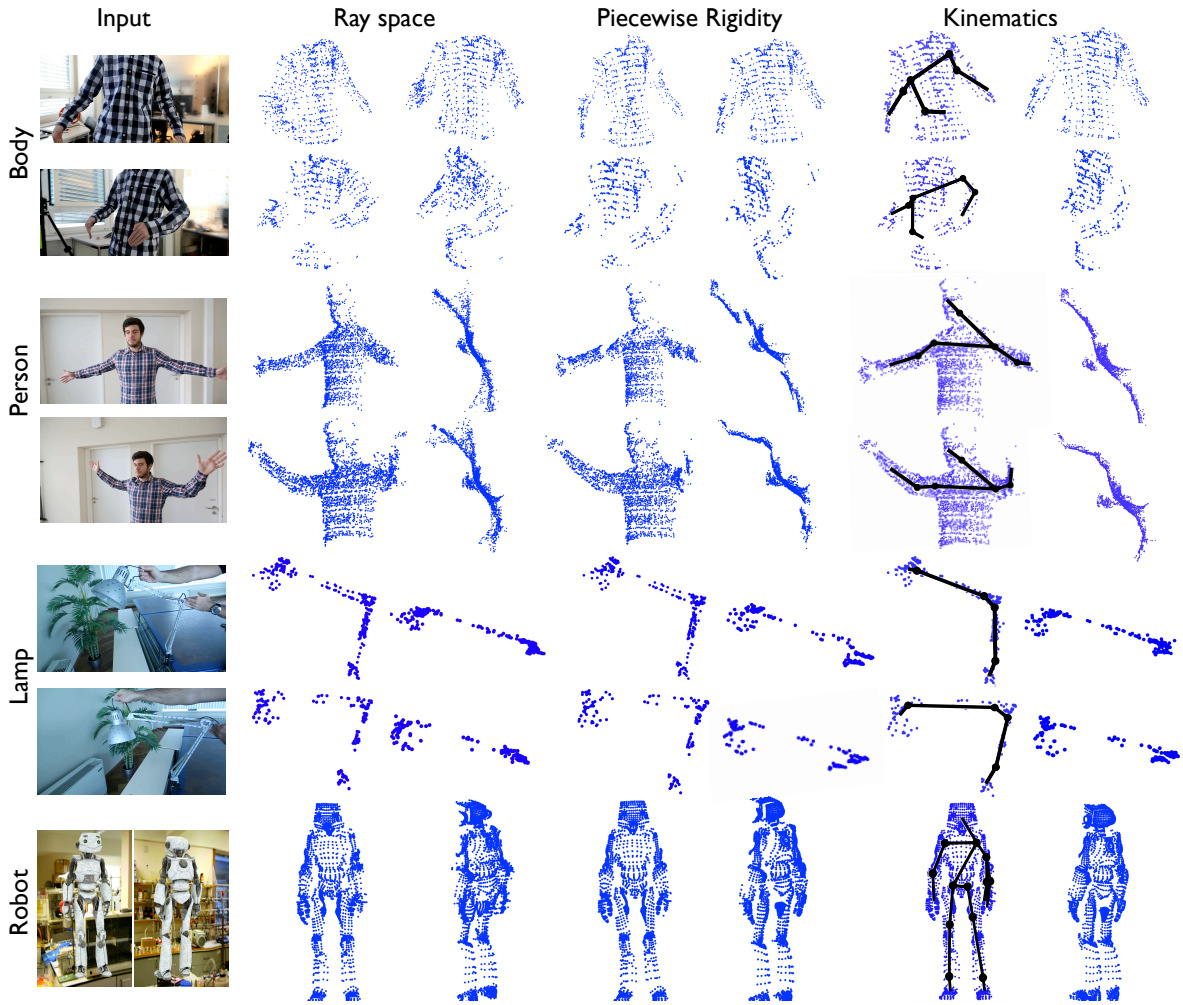

Figure 7. Results of our algorithm on one synthetic and three real datasets. For each dataset, we show reconstructions from two different viewing angles and at two points in time. Note that the amount of noise is reduced and the articulated motion becomes more realistic as we proceed through our pipeline. Kinematic constraints make it possible to keep piecewise rigid pieces together, such that the resulting 3D reconstructions are physically plausible. We show the results of the last step with and without the skeletons for a better visualization.

rotated a perspective camera around the object to project the 3D vertex positions. We varied the per-frame camera separation for our algorithm between 3^o and 8^o, simulating different camera speeds. In Table 2 we compare our error to three state-of-the-art techniques for NRSfM: CSF [16], SPM [12] and EM-PND [21]. In order to be consistent with the comparisons, we changed the Frobenius norm in Eq. (13) to the L2-norm, as is used in [21]. It can be seen that our errors decrease steadily as the per-camera separation and the camera speed increase, which leads to more robust triangulation of trajectories. Eventually, our results surpass other methods. Due to the low number of points in these datasets, we only applied our RayS step. Given more points, the result quality could be further improved using articulation constraints. Since our method does not rely on specific camera paths, any camera motion could be used for this comparison. We chose rotating cameras since they enable easy simulation of different camera speeds with similar viewing angles.

Timing. Most steps of our method are efficient, with the nonlinear optimization of rigid components being the most computationally expensive. The time required by our algorithm depends on the number of frames and point tracks. We present timings on a desktop with 3.2 GHz Intel processor and 32 GB RAM in Table 3. Note that our Matlab and CERES implementations use only a single core. All steps can be greatly optimized using CPU & GPU parallelization.

Limitations and Future Work. The quality of our initial ray-space optimization depends on the relative speed of the point tracks vs. the camera motion. It requires the majority of observed motion to be camera translation as opposed to object motion. In the extreme case of a static camera, all rays will converge at the center, yielding a trivial solution. This situation is unavoidable, as 3D pose is ambiguous without prior information. Despite this, we found that our RayS step computes accurate trajectories for initialization.

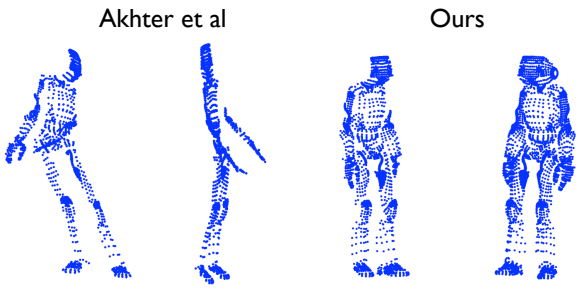

Figure 8. Comparison of our result to prior work [4] on our synthetic dataset. Our algorithm reconstructs the shape more faithfully due to the articulation constraints and perspective camera models.

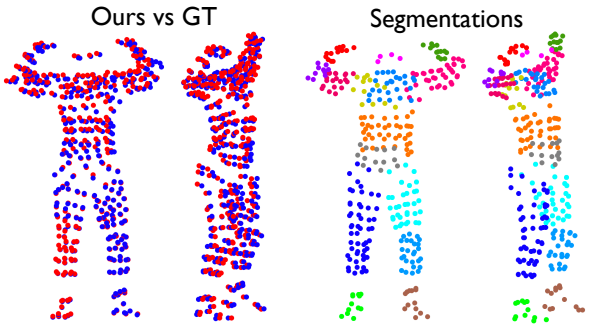

Figure 9. Our results on the 'skin' dataset from [26]. On the left, we show our reconstructed point cloud (blue) vs the ground truth point cloud (red) for a frame from two different viewpoints. On the right, we show how our segmentations from the same viewpoints. See the accompanying video for additional visualizations.

When dealing with real-world data, the availability of point tracks is very important, and on untextured objects, we cannot create 3D reconstructions. However, our RaySS step can work with a low number of tracks, as it treats them individually. Furthermore, lost point tracks due to occlusions are a common issue with free camera movement. Our approach is robust to new tracks and occlusions, but considers only contiguous tracks. Re-detecting and connecting tracks after occlusions is an interesting future work.

Compared to most existing techniques, our method requires camera poses to be computed a priori, which can be computed in many cases from static backgrounds using SfM. Excessive motion in the background can degrade SfMs performance. However, these poses can be used to generate better reconstructions using realistic perspective camera models. Extending our technique to be used without camera poses is an interesting area for future work.

5. Conclusion

In this work, we have presented an automatic method for 3D reconstruction of articulated motion from video taken by a single moving camera. Our approach consists of three main stages; a novel, efficient ray-space optimization, piecewise rigid reconstruction, and finally a fully articulated reconstruction. Our ray-space optimization both serves as a robust initial solution for the object shape and helps with automatically segmenting the object into respective rigid components. By consecutively adding articulation constraints, we can reconstruct the full shape and motion of the 3D object. Our technique is able to deal with large camera motions, perspective cameras, sparse point tracks and self-occlusions. We hope that the ideas presented in this paper provide a new perspective on the problem of articulated object reconstruction.

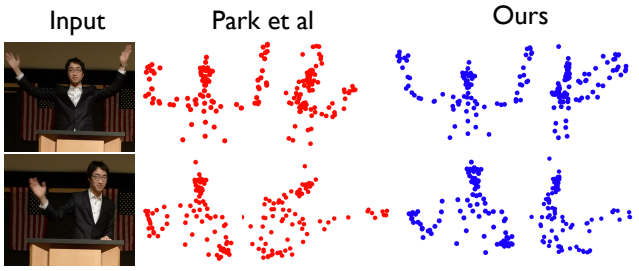

Figure 10. Comparison to [24], with two viewpoints on 3D points for both frames. There is less distortion on the arms in our result.

	CSF	SPM	EM-PND	RaySS - 3^o	RaySS - 5^o	RaySS - 8^o
walking	0.0708	0.0861	0.0465	0.1038	0.0636	0.0363
yoga	0.0226	0.0224	0.014	0.0378	0.0167	0.0115
stretch	0.0219	0.0288	0.0156	0.0335	0.0221	0.0117
pickup	0.0607	0.0356	0.0372	0.0333	0.0248	0.0141
drink	0.0123	0.0216	0.0037	0.0063	0.0037	0.0020
dance	0.1349	0.1454	0.1834	0.1697	0.1020	0.0693

Table 2. Comparison of the reconstruction error of RaySS versus other non-rigid reconstruction techniques for different per-frame separations. The error values for the other methods are taken from [21]. Green cells are marked as the ones with the lowest error, and blue cells with the second lowest error. The effect of the camera speed can be seen in these examples by comparing our approach with different per-frame separation. Ours is the only approach using a perspective projection of the ground truth points.

Method	Time
Ray-space optimization (Section 3.1)	10 sec
Segmentation (Section 3.2)	3 min
Piecewise Reconstruction (CERES) (Section 3.2)	35 min
Joint Computation (Section 3.3)	30 sec
Articulated Reconstruction (CERES) (Section 3.3)	75 min

Table 3. Runtimes of our different steps for the *Person* dataset with 300 frames, 6 articulated parts and 5000 point tracks.

Acknowledgements

We are grateful to Joël Bohnes for helping with the implementation and Maurizio Nitti for generating the Robot data set. We thank Ijaz Akhter, Lourdes Agapito, João Fayad, Yuchao Dai and Jingyu Yan for sharing their code and data with us. Lastly, we thank the authors of [4, 16, 23, 24, 26] for making their code and data sets available online.

References

[1] VisualSFM : A Visual Structure from Motion System. http://ccwu.me/vsfm/. [Online; accessed 09-Nov-2014].

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, 2011.

[3] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org.

[4] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade. Nonrigid structure from motion in trajectory space. In *NIPS*, pages 41–48, 2008.

[5] S. Avidan and A. Shashua. Trajectory triangulation: 3d reconstruction of moving points from a monocular image sequence. *IEEE PAMI*, 22(4):348–357, 2000.

[6] A. O. Balan and M. J. Black. An adaptive appearance model approach for model-based articulated object tracking. In *CVPR*, 2006.

[7] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, pages 187–194, 1999.

[8] M. Brand. Morphable 3d models from video. In *CVPR*, 2001.

[9] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR*, 2000.

[10] A. D. Bue. A factorization approach to structure from motion with shape priors. In *CVPR*, 2008.

[11] A. D. Bue, F. Smeraldi, and L. de Agapito. Non-rigid structure from motion using ranklet-based tracking and non-linear optimization. *Image Vision Comput.*, 25(3):297–310, 2007.

[12] Y. Dai, H. Li, and M. He. A simple prior-free method for non-rigid structure-from-motion factorization. In *CVPR*, 2012.

[13] J. Fayad, L. de Agapito, and A. Del Bue. Piecewise quadratic reconstruction of non-rigid surfaces from monocular sequences. In *ECCV*, pages 297–310, 2010.

[14] J. Fayad, C. Russell, and L. de Agapito. Automated articulated structure and 3d shape recovery from point correspondences. In *ICCV*, pages 431–438, 2011.

[15] A. Fossati, M. Dimitrijevic, V. Lepetit, and P. Fua. From canonical poses to 3d motion capture using a single camera. *IEEE PAMI*, 32(7):1165–1181, 2010.

[16] P. F. U. Gotardo and A. M. Martínez. Computing smooth time trajectories for camera and deformable shape in structure from motion with occlusion. *IEEE PAMI*, 2011.

[17] A. Hartley and A. Zisserman. *Multiple view geometry in computer vision (2. ed.)*. Cambridge University Press, 2006.

[18] R. Hartley and R. Vidal. Perspective nonrigid shape and motion recovery. In *ECCV*, 2008.

[19] A. Hornung, S. Sar-Dessai, and L. Kobbelt. Self-calibrating optical motion tracking for articulated bodies. In *VR*, 2005.

[20] J. Y. Kaminski and M. Teicher. A general framework for trajectory triangulation. *Journal of Mathematical Imaging and Vision*, 21(1-2):27–41, 2004.

[21] M. Lee, J. Cho, C. Choi, and S. Oh. Procrustean normal distribution for non-rigid structure from motion. In *CVPR*, pages 1280–1287, 2013.

[22] M. Paladini, A. D. Bue, M. Stosic, M. Dodig, J. M. F. Xavier, and L. de Agapito. Factorization for non-rigid and articulated structure using metric projections. In *CVPR*, 2009.

[23] M. Paladini, A. D. Bue, J. M. F. Xavier, L. de Agapito, M. Stosic, and M. Dodig. Optimal metric projections for deformable and articulated structure-from-motion. *IJCV*, 2012.

[24] H. S. Park and Y. Sheikh. 3d reconstruction of a smooth articulated trajectory from a monocular image sequence. In *ICCV*, 2011.

[25] H. S. Park, T. Shiratori, I. Matthews, and Y. Sheikh. 3d reconstruction of a moving point from a series of 2d projections. In *ECCV*, pages 158–171, 2010.

[26] S. I. Park and J. K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Trans. Graph.*, 25(3):881–889, 2006.

[27] M. Perriollat, R. I. Hartley, and A. Bartoli. Monocular template-based reconstruction of inextensible surfaces. *IJCV*, 95(2):124–137, 2011.

[28] A. Rehan, A. Zaheer, I. Akhter, A. Saeed, M. H. Usmani, B. Mahmood, and S. Khan. Nrsfm using local rigidity. In *WACV*, pages 69–74, 2014.

[29] C. Russell, J. Fayad, and L. de Agapito. Energy based multiple model fitting for non-rigid structure from motion. In *CVPR*, pages 3009–3016, 2011.

[30] C. Russell, R. Yu, and L. de Agapito. Video pop-up: Monocular 3d reconstruction of dynamic scenes. In *ECCV*, pages 583–598, 2014.

[31] A. Shashua and L. Wolf. Homography tensors: On algebraic entities that represent three views of static or moving planar points. In *ECCV*, 2000.

[32] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.

[33] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *IJCV*, 9(2):137–154, 1992.

[34] L. Torresani, A. Hertzmann, and C. Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE PAMI*, 30(5):878–892, 2008.

[35] P. A. Tresadern and I. D. Reid. Articulated structure from motion by factorization. In *CVPR*, 2005.

[36] A. Varol, M. Salzmann, E. Tola, and P. Fua. Template-free monocular reconstruction of deformable surfaces. In *ICCV*, pages 1811–1818, 2009.

[37] J. Xiao, J. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. *IJCV*, 67(2):233–246, 2006.

[38] J. Yan and M. Pollefeys. A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *IEEE PAMI*, 30(5):865–877, 2008.

[39] H. Zhou and T. S. Huang. Recovering articulated motion with a hierarchical factorization method. In *Gesture Workshop*, pages 140–151, 2003.