Temporally Consistent Motion Segmentation from RGB-D Video

P. Bertholet[1] A. E. Ichim [2] and M. Zwicker[1]

[1]Computer Graphics Group, University of Bern, Switzerland
[2]Computer Graphics and Geometry Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland

Abstract
Temporally consistent motion segmentation from RGB-D videos is challenging because of the limitations of current RGB-D sensors. We formulate segmentation as a motion assignment problem, where a motion is a sequence of rigid transformations through all frames of the input. We capture the quality of each potential assignment by defining an appropriate energy function that accounts for occlusions and a sensor specific noise model. To make energy minimization tractable, we work with a discrete set instead of the continuous, high dimensional space of motions, where the discrete motion set provides an upper bound for the original energy. We repeatedly minimize our energy, and in each step extend and refine the motion set to further lower the bound. A quantitative comparison to the current state of the art demonstrates the benefits of our approach in difficult scenarios.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Leveraging motion for object segmentation in videos is a well studied problem. In addition, with RGB-D sensors it has become possible to exploit not only RGB color data, but also depth information to solve the segmentation problem. The goal of our approach is to allow a user, or a robotic device, to move objects in a scene while recording RGB-D video, and to segment objects based on their motion. This scenario has applications, for example, in robotics, where a robotic device could manipulate the scene to enhance scene understanding [MGC*15]. Another application scenario is 3D scene acquisition, where a user would be enabled to physically interact with the scene by moving objects around as the scene is being scanned. The system would then segment and reconstruct individual objects, instead of returning a monolithic block of geometry. KinectFusion-type techniques enable similar functionality, but with the restriction that a full scene reconstruction needs to be available before segmentation can start [IKH*11]. In contrast, we do not require a complete scan of an entirely static scene.

Despite considerable interest recently in computer graphics, computer vision, and robotics, RGB-D motion segmentation remains challenging. Current RGB-D sensors provide limited spatial resolution and suffer from strong noise, which makes it hard to segment and track the motion of small objects. In addition, data from a single sensor suffers from occlusions, which complicates consistent segmentation of objects that cannot be observed in some input frames. We found that current approaches often cannot handle these challenges robustly, and we propose a novel algorithm that leads to significant improvements in difficult scenarios.

Our approach assumes a piecewise rigid motion model, and does not constrain camera movement. Let us define a motion as a sequence of rigid transformations that describes the trajectory of a point through all input RGB-D frames. We formulate segmentation as an energy minimization problem, and define the energy as a functional over all possible motion assignments to the observed scene points. A key idea to make this approach tractable is to work with a discrete set of motions, which provides an upper bound for our original energy. We iteratively expand and refine this motion set to lower our bound in each step. We solve the segmentation problem repeatedly with the motion set, and expand and refine the set to obtain a better segmentation with a lower energy each time.

A crucial component in this approach is our technique to construct suitable motions to add to our motion set, which will allow us to further reduce the segmentation energy in the next step. We achieve this by building on the segments of the current segmentation, and recombining them in a way to avoid getting stuck in bad local minima. We then find new motions by solving a geometric registration problem for the recombined segments. Finally, we compare our approach to state of the art techniques in computer graphics and robotics, and demonstrate significant improvements. In summary, we make the following contributions:

- We describe a novel algorithm for temporally consistent motion segmentation of RGB-D data that is formulated as a motion assignment problem over the continuous, high dimensional space of rigid motions through the input sequence.
- We propose a practical algorithm to solve the problem by working with a discrete set of motions, which provides an upper bound of the original energy. We repeatedly solve the motion

assignment problem by extending and refining the motion set, each time lowering the bound and improving the segmentation.

- We demonstrate significant improvements over the state of the art in several difficult scenarios.

2. Previous Work

Motion Segmentation from RGB Video. Motion segmentation from video is a classical topic in computer vision, and a full review is beyond the scope of this paper. We are inspired by Ochs et al. [OMB14], who observe that motion is exploited most effectively by considering it over larger time windows, for example by tracking feature point trajectories. They point out the advantage of obtaining consistent segmentations over entire video sequences, which is also a goal in our approach. Recent techniques jointly solve for RGB video segmentation and optical flow [SLSJB16, TYB16]. Learning based approaches have also become popular for motion segmentation [FAFM15] and optical flow [DFI*15], and Sevilla-Lara et al. [SLSJB16] leverage semantic segmentation using deep CNNs in their approach. Similar to these techniques, we also obtain both a segmentation and motion estimates. Our approach, however, relies purely on the geometric information in RGB-D data. It may be interesting in the future to combine this with deep learning techniques for RGB images and video.

Motion Segmentation from RGB-D Data. A number of techniques have been quite successful in segmenting objects from pairs of RGB-D frames. Our work is most related to the recent approach of Stückler et al. [SB15] who also use a piecewise rigid motion model and perform energy minimization to recover object segmentation and motion simultaneously. Similar to our approach, they use a coordinate descent strategy for energy minimization and graph cuts to update the segmentation. Earlier work includes the approach by Ven et al. [vdVRT10], who also jointly solve for segmentation and motion by formulating a CRF and using belief propagation. Both techniques, however, are limited to pairs of RGB-D frames. The main difference to our technique is that we solve globally over an entire RGB-D sequence, which allows us to consistently label segments, track partial objects, and accumulate data over time.

Our problem is similar to other techniques that leverage entire RGB-D sequences to segment objects based on their motion, and to fuse partial objects over all frames into more complete 3D reconstructions. The original KinectFusion system [IKH*11] can segment moving objects after a complete scan of a static scene has been obtained. Perera et al. [PBH*15] improve on this by segmenting objects based on incremental motion, whereas KinectFusion requires objects to move completely outside their originally occupied volume in the static scene. As a crucial difference to our approach, both approaches rely on a complete 3D reconstruction of a static version of the scene that needs to be acquired first, before object segmentation can be performed.

The goal of Ma and Sibley's work [MS14] is similar to ours, as they discover, track and reconstruct objects from RGB-D videos based on piecewise rigid motion. A key difference is that they use an incremental approach as they move forward over time to discover new objects, by detecting parts of the scene that cannot be tracked by the dominant camera motion. This means that groups of objects that initially exhibit the same motion (for example one object moving on top of another), but later split and move along different trajectories, cannot be consistently identified and separated over the entire sequence. In contrast, we optimize jointly over segmentation and motion, taking into account entire RGB-D sequences, instead of incremental segmentation followed by tracking. This allows to successfully resolve such challenging scenarios.

Recently, Qing et al. [YLX*16] and Kim et al. [KLAK16] also proposed techniques for simultaneous space-time segmentation and motion estimation for point cloud sequences, addressing the same problem statement as we do. Both techniques focus on articulated objects, rather than entire scenes that may include complex interaction and occlusion between different objects, which our technique can handle. Qing et al. [YLX*16] start with a non-rigid ICP technique [PB11] to establish correspondences between temporally adjacent point cloud frames. These correspondences are extended to local temporal point trajectories, and the key idea is to cluster these trajectories to obtain the final space-time segmentations. This is achieved in three steps: first, they obtain initial, per frame point cloud segmentations by linkage clustering the trajectories [FRP09]. Then they make the segmentation consistent by propagating segmentation boundaries incrementally to all frames, which leads to an oversegmentation. Finally, they consolidate the oversegmentation using a space-time optimization based on graph cuts. The disadvantage of this approach is that it hinges on the correspondences established in the non-rigid ICP step. If this step fails, the approach cannot recover. In contrast, our approach is more robust because it repeatedly re-estimates correspondences. Kim et al. [KLAK16] perform segmentation on a sparse set of trajectories seeded in the first frame via a combination of probabilistic subspace clustering and graphcuts, followed by motion estimation. Their approach, however, assumes absence of occlusions. We include comparisons to these techniques and show that our approach handles complex scenarios more robustly.

RGB-D Scene Flow. Our problem is also related to the problem of obtaining 3D scene flow, that is, frame-to-frame 3D flow vectors, from RGB-D data. For example, Herbst et al. [HRF13] generalize two-frame variational 2D flow to 3D, and apply it for rigid motion segmentation. Quigoro et al. [QBDC14] model the motion as a field of twists and they encourage piecewise rigid body motions. They do not address segmentation, and their method processes pairs of RGB-D frames separately. Sun et al. [SSP15] also address scene flow, but they formulate an energy over several frames in terms of scene segmentation and flow. While they can deal with several moving objects, their segmentation is separating depth layers, not objects. They also show results only for short sequences of less than ten frames. Jaimez et al. [JSS*15] leverage a soft piecewise rigidity assumption and jointly optimize for segmentation and motion to extract high quality scene flow and segmentations for pairs of RGB-D frames. In contrast, our goal is to separate objects only based on their individual motion, and label the segmented objects consistently over time. We perform energy minimization on video segments instead of frame pairs, which also allows us to reason explicitly about occlusion.

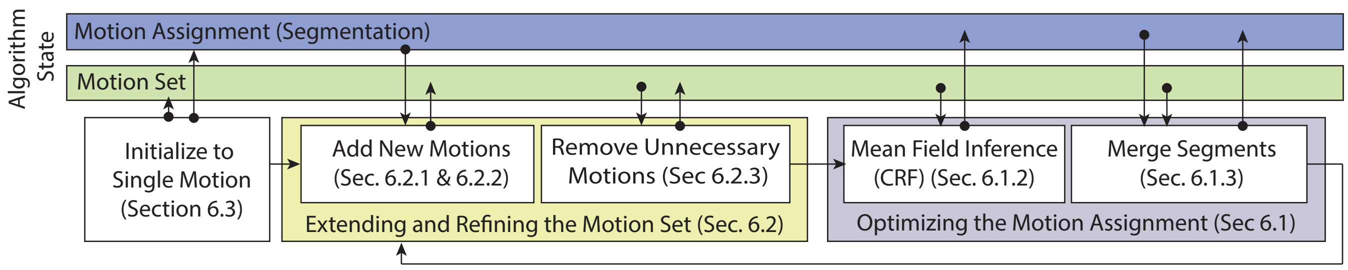

Algorithm State
Motion Assignment (Segmentation)
Motion Set
Initialize to Single Motion (Section 6.3)
Add New Motions (Sec. 6.2.1 & 6.2.2)
Remove Unnecessary Motions (Sec 6.2.3)
Extending and Refining the Motion Set (Sec. 6.2)
Mean Field Inference (CRF) (Sec. 6.1.2)
Merge Segments (Sec. 6.1.3)
Optimizing the Motion Assignment (Sec 6.1)

Figure 1: Our approach minimizes an energy, described in Section 5, that represents the quality of a motion assignment to all scene points in the RGB-D sequence. Our key idea (Section 6) is to repeatedly minimize an upper bound of the original energy by considering only a discrete set of motions. Each time we propose an expanded and refined set of motions that leads to tighter upper bounds of the original energy. Hence our iteration includes two distinct stages: finding a set of motions that allow a tightening of the upper bound (Section 6.2) and finding an assignment that minimizes the upper bound (Section 6.1). Vertical arrows indicate read and write operations into the scene representations.

3. Overview

Given a sequence of RGB-D frames as an input, our goal is to assign an object motion to each RGB-D point, where the motion describes the 3D trajectory of the point through the entire sequence. We assume a piecewise rigid motion model, and define objects as groups of scene points that exhibit the same rigid motion through the sequence. Hence the motion assignments represent a segmentation into objects. We do not assume any a priori knowledge about object geometry or appearance, or the number of objects, and camera motion is unconstrained.

Figure 1 shows an overview of our approach, which is based on repeatedly minimizing a spatio-temporal motion segmentation energy. The original energy is defined as a function of the assignments of a motion to each point in the RGB-D frames, where all motions from the continuous, high dimensional space are considered, and the number of objects (that is, the number of different motions being assigned). To make this practical, a key idea is to work with a discrete set instead of the continuous space of motions, which provides an upper bound to the original energy. In each step, we extend and refine the motion set to further lower the bound and improve the resulting segmentation. To make the energy robust to noise and occlusions, reduce the number of user parameters, and allow generalization to different sensors, we first introduce a parameterized noise model for a depth sensor at hand in Section 4. We then define our motion segmentation energy in Section 5, and describe the energy minimization in detail in Section 6.

Throughout the paper we will use the following notation: We denote the set of RGB-D frames by $f_1,...,f_n$. Any observed 3D point x has associated attributes such as its normal $x.n$, its 3D coordinates $(x.x,x.y,x.z)$, its pixel coordinates $(x.i,x.j)$, and the index of the frame $x.f \in \mathbb{N}$ where x is captured. The set of all observed points is denoted by X. To formulate motion segmentation we introduce the space of all rigid motions through the point cloud sequence and denote it by $\mathbb{M}$. For a sequence with n frames $\mathbb{M}$ is a $(n-1)\cdot 6$ dimensional space, since after choosing an arbitrary frame as reference frame there are $(n-1)$ rigid transformations left to be determined with 6 degrees of freedom each. A motion $\mathcal{M} \in \mathbb{M}$ provides rigid alignments $\mathcal{M}^{i\to j}x$ to map x from any frame f_i to any frame f_j. We assume x's attributes like its normal are transformed too. We define a motion segmentation as a mapping s,

$$s : X \to \mathbb{M}, s(x) = \mathcal{M}_x,$$

which assigns a motion $\mathcal{M}_x$ to every point x. We will propose a variational description for motion segmentation, where we seek the segmentation s that minimizes an energy functional $\mathcal{E}(s) \in \mathbb{R}$.

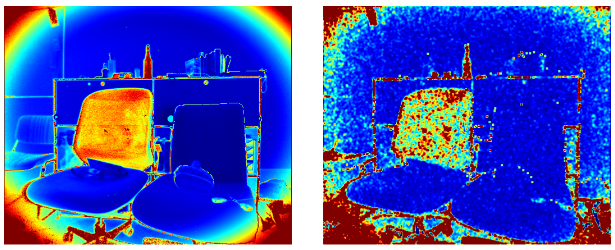
4. Depth Sensors with Calibrated Noise

We propose a sensor dependent, per-pixel isotropic noise model given by the following parameters:

- $\sigma(x)$: the local standard deviation (noise magnitude) of sensed depth as a function of image coordinates $x.i, x.j$ and depth $x.z$
- $\tau(x)$: the local sampling density of the sensor at x, which we define as half the maximal diameter of the area covered by pixel $x.i, x.j$ in the observed scene, under the assumption that x lies on a plane through x with normal $x.n$.
- α: A maximal angle between the surface normal of a point and the viewing direction from the sensor. Surfaces in the scene with a larger angle are deemed unobservable.

To calibrate a noise model for a sensor we manually choose a conservative value for α. The sampling density τ at any position and depth can be computed from the camera model and the normal of the local plane via projection and unprojection. We obtain the standard deviation $\sigma(x)$ via regression of pixel-wise noise estimates to simple, sensor-specific parametric models as follows: given a set of frame pairs from static scenes, we first compute pixel-wise estimates for the standard deviation as the pixel-wise differences between acquired depth values. For the Kinect for Xbox One we propose fitting $a_0 + a_1 z + a_2 z^2 + b_1 max(d - b_2, 0)^2 + c\frac{1}{infrared}$ to the observed noise, where d is the image space distance in pixels to the cameras principal point. The quadratic polynomial in z models that noise increases with depth, the term $b_1 max(d - b_2, 0)^2$ models the typical radial shape of the noise of the Kinect One sensor and the term $c\frac{1}{infrared}$ models the dependence of noise on the infrared spectrum reflectance of objects typical for time of flight sensors. For the Asus Xtion we choose a simpler model $a_0 + a_1 z + a_2 z^2$. See Figure 2 for the extracted parameters, and a qualitative example.

In summary, we model the sensor noise at a point x as an isotropic normal distribution $\mathcal{N}((x.x,x.y,x.z)^T, \hat{\sigma}^2(x))$, where



	a_0	a_1	a_2	b_1	b_2	c
Kinect One	5.244e-03	-8.741e-03	3.060e-03	1.425e-06	174	5.79
ASUS	6.217e-04	-1.643e-04	3.37e-03	-	-	-

Figure 2: Visualization of the noise model for the Kinect One with the fitted model on the left and the measured noise on the right. The measurements were slightly blurred for visualization purposes. The table reports the fitted parameters for both sensors we use.

$\hat{\sigma}(x) = max(\tau(x), \sigma(x))$. Finally, we discard the point if it is observed at a grazing angle, that is $\angle x.n, (eye - x) > \alpha$.

5. 4D Spatio-temporal Motion Segmentation Energy

We propose to seek the motion segmentation $s : X \rightarrow \mathbb{M}$ that minimizes a spatio-temporal energy that measures both if motions are assigned to scene points consistently over time, and if the assigned motions accurately match the observed data. Our energy builds on binary terms $\psi_{x,y}(\mathcal{M}_x, \mathcal{M}_y)$, which depend on the motion assignments $s(x) = \mathcal{M}_x, s(y) = \mathcal{M}_y$ at all pairs of points x, y, and unary terms $\psi_x(\mathcal{M}_x)$, which depend on the motion assignment $s(x) = \mathcal{M}_x$ at each single point x. The energy is the sum over all these terms,

$$\mathcal{E}_{4DMSEG}(s) =$$
$$cost_{label}^{|s(X)|} \left(\sum_{x \in X} \sum_{y \in X} \psi_{x,y}(\mathcal{M}_x, \mathcal{M}_y) + \sum_{x \in X} \psi_x(\mathcal{M}_x) \right), \quad (1)$$

where $|s(X)|$ denotes the number of distinct motions that s assigns to X . The factor $cost_{label} > 1$ encourages s to assign smaller numbers of different motions to avoid oversegmentation. We set it to be just slightly larger than one, $cost_{label} = 1.005$.

Binary Terms. The binary terms represent a registration cost between two points x and y, and they consist of two components, a *correspondence probability* and a *registration error*. First, for each scene point $x \in X$, we will define a probability $P_{x,y}(\mathcal{M}_x)$ for an other point y to correspond to the same physical scene point as x. We consider this probability a function of the motion $s(x) = \mathcal{M}_x$ that s assigns to x. Second, we will define a pairwise registration error $error_{x,y}(\mathcal{M}_x, \mathcal{M}_y)$ between a pair of points x and y, which depends on both the assigned motions at x and y. The binary registration cost is then the registration error weighted by the correspondence probability,

$$\psi_{x,y}(\mathcal{M}_x, \mathcal{M}_y) = \frac{P_{x,y}(\mathcal{M}_x)}{\sum_{y}' P_{x,y'}(\mathcal{M}_x)} error_{x,y}(\mathcal{M}_x, \mathcal{M}_y), \quad (2)$$

where we normalize the weights to sum up to one for each point x. For each x, the sum $\sum_y \psi_{x,y}(\mathcal{M}_x, \mathcal{M}_y)$ now yields a weighted average of registration errors. Hence the pairwise cost only depends on the quality of likely observations, but not on their quantity.

Unary Term. We design the unary term to encourage temporal coherence, by penalizing solutions with scene points x that are unlikely to be observed in other frames,

$$\psi_x(\mathcal{M}_x) = \frac{d}{\sum_y P_{x,y}(\mathcal{M}_x)}, \quad (3)$$

where d is a user defined constant. We now explain the definition of our correspondence probability and registration error in more detail.

Correspondence Probability. We model the probability that a pair of points x and y correspond to the same location in the scene under the motion segmentation s by considering how x is positioned relative to y when mapped to y's frame under its motion $s(x) = \mathcal{M}_x$. The correspondence probability consists of four factors

$$P_{x,y}(\mathcal{M}_x) = \underbrace{1_{y==NN(\mathcal{M}^{\rightarrow y}x)}}_{(i)} \cdot \underbrace{(P_{!occl}(\mathcal{M}^{\rightarrow y}x))}_{(ii)} \cdot$$
$$\underbrace{i_{>\alpha}(\angle(\mathcal{M}^{\rightarrow y}x.n, eye - \mathcal{M}^{\rightarrow y}x))}_{(iii)} \underbrace{e^{-\frac{|x.f - y.f|^2}{(\sigma_{temp}^2)}}}_{(iv)}, \quad (4)$$

where we use the indicator function $1_B = 1$ if B is true, and 0 otherwise, NN means nearest neighbor, and the shortcut notation $\mathcal{M}^{\rightarrow y}x$ indicates the mapping of x under its assigned motion $\mathcal{M}_x$from frame $x.f$ to $y.f$, that is $\mathcal{M}_x^{x.f \rightarrow y.f}x$.

First, (i) is a binary factor indicating whether y is the nearest neighbor of x when x is mapped to y's frame. Next, (ii) is the probability $P_{!occl}(\mathcal{M}^{\rightarrow y}x)$ that the mapped point $\mathcal{M}^{\rightarrow y}x$ is not occluded, which we define using the camera noise model. The observation that might occlude $\mathcal{M}^{\rightarrow y}x$ is the point in the pixel in frame $y.f$ that we hit by projecting $\mathcal{M}^{\rightarrow y}x$ onto the image plane. We denote this point $\pi(\mathcal{M}^{\rightarrow y}x)$, and its noise magnitude is modeled as $\hat{\sigma}(\pi(\mathcal{M}^{\rightarrow y}x))$. The probability that it occludes the mapped point can be expressed directly using the cumulative density function, denoted cdf, of the normal distribution $\mathcal{N}(\pi(\mathcal{M}^{\rightarrow y}x).z, \hat{\sigma}(\pi(\mathcal{M}^{\rightarrow y}x)))$. Hence

$$P_{!occl}(\mathcal{M}^{\rightarrow y}x) = P(\pi(M^{\rightarrow y}x).z > (M^{\rightarrow y}x).z) = 1 - cdf(x.z).$$

We set $P_{!occl}(\mathcal{M}^{\rightarrow y}x)$ to zero if $\mathcal{M}^{\rightarrow y}x$ is outside of the camera frustum or if there was no observation at $\pi(M^{\rightarrow y}x)$.

Third, (iii) is a weight indicating whether x's orientation to the camera would still allow x to be observed by the depth sensor in y's frame. We write this as $i_{>\alpha}(\varphi) = clamp_0^1(cos(\varphi \cdot \pi/(2\alpha)))$, where α is a parameter of the camera noise model (Section 4). Finally, (iv) is a probability depending on the temporal distance between x and y, as in practice correspondences established to temporally closer frames are more reliable. We fixed $\sigma_{temp} = 30$.

Registration Error. The $error_{x,y}(\mathcal{M}_x, \mathcal{M}_y)$ is a binary term that distinguishes whether x and y belong to the same object, that is whether the assigned motions $\mathcal{M}_x$ and $\mathcal{M}_y$ are the same. If $\mathcal{M}_x \neq$

$\mathcal{M}_y$, we assign a maximal penalty. Otherwise, we use a sum of a clamped L_2 error and an incremental error $incr_y(\mathcal{M}_y)$ between mapped and observed 3D positions,

$$error_{x,y}(\mathcal{M}_x, \mathcal{M}_y) = \tag{5}$$
$$\begin{cases} 1 & \mathcal{M}_x \neq \mathcal{M}_y \\ \frac{1}{2}\left(\frac{clamp_0^{3\hat{\sigma}}(||\mathcal{M}_x^{\rightarrow y}x - y||^2)}{3\hat{\sigma}} + incr_y(\mathcal{M}_y)\right) & \text{else.} \end{cases}$$

We choose $3\hat{\sigma}$ for clamping, as under normal distribution 99.73% of data lies in this range and treat other data as outliers. If the quality of the motions is low and exhibits drift, the clamped error saturates quickly when x and y are temporally far apart. The incremental motion penalty avoids the problem with drift, by mapping y to its previous and next frame, and measuring the registration error there,

$$incr_y(\mathcal{M}_y) = \tag{6}$$
$$\sum_{\delta=\pm 1} \frac{P_{!occl}(\mathcal{M}^{\rightarrow\delta}y)\frac{clamp_0^{3\hat{\sigma}}(||NN(\mathcal{M}^{\rightarrow\delta}y) - \mathcal{M}^{\rightarrow\delta}y||^2)}{3\hat{\sigma}}}{\sum_{\delta=\pm 1} P_{!occl}(\mathcal{M}^{\rightarrow\delta}y)},$$

where we also take into account potential occlusion. We use the shortcut notation $\mathcal{M}^{\rightarrow\pm 1}y$ for $\mathcal{M}_y^{y.f\rightarrow y.f\pm 1}y$ (note that *incr* is used only if $\mathcal{M}_x == \mathcal{M}_y$) and $NN(\mathcal{M}^{\rightarrow\pm 1}y)$ for the nearest neighbor of the mapped point $\mathcal{M}^{\rightarrow\pm 1}y$ in the frame $y.f \pm 1$.

Discussion. Let us summarize the key features of our energy. First, it explicitly models nonuniform sensor noise and occlusions in a sound way by using a calibrated noise model for each sensor, making it easy to adapt our method to arbitrary range sensors. It is robust to occlusions by ignoring registration errors of occluded points. Further, the registration error is robust to outliers and missing data, and it enforces temporal consistency. Note that the energy does not include a binary term for spatial smoothness of motion assignments, but our implementation provides some spatial smoothing through subsampling and interpolation, as we describe in Section 6.1.2. Finally, the only user parameters of our energy are σ_{temp} in the correspondence probability, which allows to increase the weight of temporally local evidence, and d, which is used to get rid of the trivial solution of assigning motions which explain an observed point x by the point being visible in only one frame and being out of frustum or invisible else.

6. Energy Minimization

We next describe our approach for finding a motion assignment $\hat{s} : X \rightarrow \mathbb{M}$ that minimizes our energy,

$$\hat{s} = \underset{s:X\rightarrow\mathbb{M}}{\operatorname{argmin}}\, \mathcal{E}_{4DMSEG}(s).$$

Because the space of motions is continuous and high-dimensional it is intractable to solve this directly. Instead, we will operate with a discrete subset of motions $\{\mathcal{M}_k\}, \mathcal{M}_k \in \mathbb{M}, k \in \{1\ldots n\}$, which turns our energy minimization into a discrete labeling problem. Our strategy builds on the observation that for discrete sets $\{\mathcal{M}_k\}$ and $\{\mathcal{M}'_k\}$ we have the upper bounds

$$\begin{aligned} \min_{s:X\rightarrow\mathbb{M}} \mathcal{E}_{4DMSEG}(s) &\leq \min_{s:X\rightarrow\{\mathcal{M}_k\}\cup\{\mathcal{M}'_k\}} \mathcal{E}_{4DMSEG}(s) \\ &\leq \min_{s:X\rightarrow\{\mathcal{M}_k\}} \mathcal{E}_{4DMSEG}(s). \end{aligned} \tag{7}$$

Hence we will solve the discrete labeling problem repeatedly, by iteratively expanding the motion set $\{\mathcal{M}_k\}$ with new motions $\{\mathcal{M}'_k\}$ in each step. This monotonically tightens the upper bound and improves our solution in each step. For efficiency reasons, we will also remove unnecessary motions from the set in each step. Figure 1 illustrates our overall pipeline. We discuss the discrete labeling problem (that is, minimization of the upperbound) given a discrete set of motions in Section 6.1. In Section 6.2 we then describe the refinement of the motion set that allows us to tighten the upper bound in each step.

6.1. Optimizing the Motion Assignment

For a fixed set of n motions $\{\mathcal{M}_k\}, k \in \{1\ldots n\}$ we seek to find an assignment $s : X \rightarrow \{\mathcal{M}_k\}$ that minimizes the upper bound in Equation (7). We first solve for an assignment s that optimizes the unary and pairwise terms of the energy. We simplify the notation and write $s_x = k$ instead of $s(x) = \mathcal{M}_k$, where we represent the motion by its index in the finite set. The assignment problem is then

$$\underset{s:X\rightarrow\{1\ldots n\}}{\operatorname{argmin}} \sum_{x\in X}\sum_{y\in y} \psi_{x,y}(s_x, s_y) + \sum_{x\in X} \psi_x(s_x), \tag{8}$$

which we solve via mean-field inference. Then we greedily merge assignments to reduce the number of assigned motions $|s(X)| \leq n$ to optimize over the factor $1.005^{|s(X)|}$.

6.1.1. Mean-Field Inference for CRFs

Mean-field inference has been a popular choice for various computer vision tasks due to its simplicity, scalability and flexibility, and has been successfully applied to segmentation before [ZJRP*15] [SOD12]. Mean-field inference for our energy in Equation 8 is formulated via the likelihood of its corresponding conditional random field (CRF), $P(\{s_x\}) = \frac{1}{Z}\exp(-\mathcal{E}_{SEG}(\{s_x\}))$, where Z is an adequate normalization factor. Minimizing the energy is equivalent to maximizing the likelihood $P(\{s_x\})$, and the sought assignments $\{s_x\}$ are the MAP parameters of P. The central idea of the mean field is to approximate P by a simpler distribution $Q = \prod_{x\in X} Q_x(s_x)$, where $Q_x(s_x = l)$ can be interpreted as the (marginal) probability for assigning label (object motion) l to point x. The MAP of Q is simply given by the assignments $\operatorname{argmax}_l Q_x(s_x = l)$ for all x.

To approximate the MAP of P, in mean-field inference the approximation quality of Q is measured via the KL divergence to P. This leads to the so called free mean field energy that Q has to minimize,

$$\begin{aligned} F(s,Q) = &\sum_x \sum_{s_x=1}^{n} Q_x(s_x)\psi_x(s_x) \\ &+ \sum_{x,y}\sum_{s_x=1}^{n}\sum_{s_y=1}^{n} Q_x(s_x)Q_x(s_y)\psi_{x,y}(s_x,s_y) \\ &+ \sum_x \sum_{s_x=1}^{n} Q_x(s_x)\log(Q_x(s_x)). \end{aligned} \tag{9}$$

The first two terms are the expected value of the energy in Equation (8) under the distribution Q, while the third term is the negative entropy of Q, acting as a regularisation term on Q. Adding the constraint that the Q_x sum to one (via Laplacian multipliers) yields the following equation that the Q_x at any local maximum need to fulfill,

$$Q_x(s_x = l) = \tag{10}$$
$$\frac{1}{Z_x}\exp(-\psi_x(s_x = l) - \sum_{l'=1}^{n}\sum_{y \neq x} Q_y(s_y = l')\psi_{x,y}(s_x = l, s_y),$$

where the Z_x are normalization factors such that the Q_x sum to one for any x. This is called the update equation and describes Q as its fixpoint. The mean-field CRF inference method is the corresponding fixpoint iteration. To guarantee convergence, the updates in Equation 10 have to be performed sequentially over all points x and potential assignments $s_x = l, l \in 1 \dots n$.

6.1.2. Parallel Mean-Field Inference on the GPU

To efficiently implement the inference scheme we parallelize the update equations and perform inference on the GPU.

Parallel Updates. In the parallel scheme, we update all the probabilities $Q_x(s_x = l)$ in parallel for each x and l (Equation 10). Although naive parallelization does not guarantee convergence to a local minimum, Baqué et al. [BBFF16] showed how using additional damping in each fix point step would guarantee convergence. In our case, however, we did not experience the need for such a damping factor. Note that, as our pairwise terms $\psi_{x,y}$ are independent of s_y when $s_x \neq s_y$, the double sum from Equation 10 simplifies to

$$\sum_{y \neq x} Q_y(s_y = l)\psi_{x,y}(s_x = l, s_y = l)$$
$$+ (1 - Q_y(s_y = l))\psi_{x,y}(s_x = l, s_y \neq l). \tag{11}$$

Acceleration via Subsampling. To further reduce the complexity of the updates (Equation 10) we evaluate them only on 10% randomly sampled points. We then interpolate the result back to the full point cloud to get dense probabilities $Q_x, \forall x \in X$ defined everywhere, before the next update step.

We interpolate the label probabilities in 3D using local weighted averaging based on Euclidean distances between dense interpolation target points and the sparse samples that we used in the mean-field update. At each interpolation target point x we use Gaussian weights with standard deviation $2\hat{\sigma}(x)$. We find neighboring points via linear searches in local 7×7 windows in x's frame $x.f$, which can be efficiently parallelized.

Note that in each parallel update step, we update the sparse points based on the full point set X with interpolated probabilities. That is, the nearest neighbor searches required for the update (see Equation (4) and (6)) access the full point set. As the summation in the update (Equation (11)) only requires the probabilities of a single label l, we evaluate each label l separately (in parallel on the GPU), and only keep the probabilities for one label in GPU memory at a time. To complete the update (Equation 10), we compute the exponentiation and normalization for the sparse points.

Energy Scaling. An important aspect of mean-field inference is that its result depends on the global scale of the energy, as noted for example by Saito et al. [SOD12]. Observe that in Equation (9)

	$\psi_x(s_x = 1)$	$\psi_x(s_x = 2)$
$x = 1$	2	0
$x = 2$	0	1

$$\psi_{1,2}(s_1, s_2) = \begin{cases} 0 & \textit{if } s_1 = s_2; \\ 2 & \textit{otherwise} \end{cases}$$

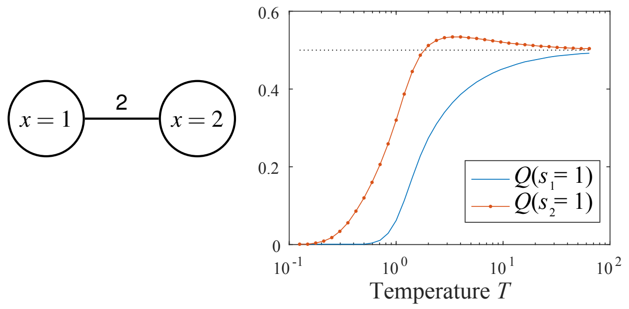


Figure 3: A minimal example to illustrate the influence of energy scaling in mean-field inference. Top row: unary and pairwise energy terms for two nodes $x = 1, x = 2$ and two possible label assignments $s_x = 1, s_x = 2$. Bottom left: the corresponding graphical model. Bottom right: minimizer Q of the free energy (Equation (9)) as a function of the temperature T used to scale the entropy. Q is only shown for the first label, that is $Q(s_1 = 1), Q(s_2 = 1)$. A value below 0.5 means that a MAP assignment selects the second label, above 0.5 it chooses the first label. The optimal labeling (minimizing Equation 8) assigns the second label to both nodes, that is $s_1 = s_2 = 2$.

the relative weight of energy (first two terms) and entropy (third term) is implicitly governed by the scale of the original energy (the unary and binary terms). We demonstrate the effect of a scale factor in a minimal example in Figure 3, where the optimal Q and its MAP result change depending on a relative scale factor T that we apply to the entropy term. The factor T is related to the Gibbs free energy in thermodynamics, where the temperature T weights the entropy term. As Baqué et al. note [BFF16], at high temperatures the Gibbs free energy is convex (because with increased weight of the entropy the pairwise terms loose their importance) and local minima start arising only at lower temperatures.

We obtain best results by gradually decreasing the temperature (increasing the relative scale of the energy), to optimize increasingly non-convex energies. We use an exponential cooling scheme where we increase the scale of our energy by a fixed factor after each mean-field update pass, to overall cover two magnitudes of energy scales. This corresponds to the exponential cooling scheme known from simulated annealing [NA98].

Pseudocode. Our full GPU-based inference algorithm is listed in Algorithm 1 in pseudocode. We first describe the data needed on the GPU (lines 1-11). The dense buffers have *number-of-frames* times *number-of-pixels* entries and provide the dense data needed to compute the CRF update (Equation (10)); the buffer *QDense* provides space to store the dense Q values for a single motion. The arrays from line 8ff encode the sample positions for the sparse samples as well as buffers $QSparse[k]$ to store the probabilities Q for each motion k and all the sparse samples. The inference method proceeds as follows: first the sparse probabilities are uniformly initialized (line 12). Then for a fixed number of iterations the sparse values are

```
1  Struct gpuData contains
2      M[] ;                          // the motion set $\{M_k\}$
3      dense data:
4      Depths[] ;                     // all depth frames
5      Normals[] ;                    // precomp. normals
6      Sigmahats[] ;                  // precomp. $\hat{\sigma}$
7      QDense[] ;     // dense Q for single motion
8      sparse data:
9      QSparsePos ;   // encodes sparse sample pos
10     QSparse[] ;    // the sparse probabilities
11 end
12 gpu::set(QSparse[:],1/numLabels) ;    // initialize Q
13 for it = 0:15 do
14     iterate over motions:
15     for k = 1:length(M) do
16         gpu::interpolate_sparse_to_dense(QSparse[k],
             QDense);
17         compute sums from the update equation Eq. (11).
18         gpu::temporal_sum_dense_to_sparse(M[k], QDense,
             QSparse[k]);
19     end
20     exponentiation and normalization
21     energyScalingFactor = pow(100,it/15)
22     gpu::crf_exp_and_normalization(QSparseOut,QSparse,
         energyScalingFactor);
23 end
```

Algorithm 1: The CRF step and the data that is stored on the GPU. Methods annotated with *gpu* :: are implemented on the GPU.

updated following Equation (10) by first interpolating the sparse values (line 16) and then evaluating the sum from Equation (11) (line 18). Finally, the results are exponentiated while taking the energy scaling scheme into account and normalized to sum to one (line 22).

6.1.3. Merging Segments

After each full iteration we greedily merge pairs of segments as long as it decreases the sum of the unary and binary terms in the motion segmentation energy (Equation (1)), or increases it by less than half a percent. This effectively optimizes over the $cost_{label}^{|s(X)|}$ term in Equation (1), since merging two segments reduces the number of assigned motions $|s(X)|$ by one. Merging segments may indeed decrease the sum of unary an binary terms if CRF inference converged to a non-optimal local minimum featuring over-segmentation. In particular, in scenes with geometrically weakly interconnected regions that belong to the same object, the filter-based CRF inference algorithm sometimes assigns different motions in each region if nearly duplicate motions are present in the motion set. Merging segments resolves such issues.

6.2. Extending and Refining the Motion Set

Our key objective in this step is to add motions to the current set that will actually contribute to further lowering the energy. To facilitate our explanations, let us use the term *label* not only for the index of a motion from our set, but also for all the points assigned to that motion. An intuitive approach would be to extend our motion set by adding the motions that geometrically best align each label across all frames containing any points that belong to the label. These motions would in turn lead to better segmentations, etc.

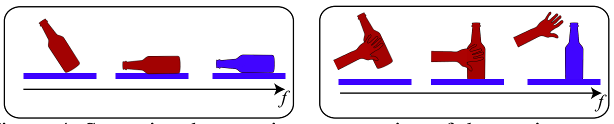


Figure 4: Scenarios that require an extension of the motion set to improve the segmentation. Left: even if we have the correct motion for the red bottle, we cannot improve the segmentation in the third frame. This is because the motion does not track the bottle in the third frame, where it was not segmented correctly. Right: with only the motion for the hand and the static table, we cannot correctly segment the bottle in the third frame. Our approach to extend the motion set can resolve these situations.

This strategy alone, however, easily gets stuck in bad local minima. A trivial example is when all points are attributed to the same label. We can find a single optimal motion for this one label, but adding this motion is not useful to refine the segmentation. In addition, scenes where different objects exhibit different motions only in a part of the sequence, but move along the same trajectories otherwise, are challenging. For example, in Figure 4 (left) the bottle tips over and remains static with the support surface after the fall. Assume that, under the current motion assignments (visualized in blue and red colors), the red label disappears after the bottle falls, which we call *"label death"*. Aligning the red label across all frames where it appears, however, cannot determine the actual motion of the bottle after the red label disappears. Hence, we cannot improve the red label in the next motion assignment step. Analogously, a label may emerge as two objects split and start moving independently (*"label birth"*).

Finally (Figure 4, right), a first object (the bottle) may first share its motion with a second one (the hand), and then with a third one (the support surface). Hence the first object (bottle) may first share its label with the second one (hand), and then switch to the third one (support surface). We call this *"label switch"*. Similarly as above, in these cases the simple strategy can get stuck with a set of motions that includes perfect alignments of the current labels, but we will never add other motions that can further reduce our energy and overcome the problems with label events (death, birth, switch) we just discussed.

We address these issues by introducing new labels, which we assemble from the ones obtained in the previous motion assignment step as described in Section 6.2.1. Then, we find motions that geometrically align these new labels, as discussed in Section 6.2.2, and add them to our set.

6.2.1. Constructing New Labels

We construct the set of new labels in two steps: We first analyze the given labels to detect label events (death, birth, switch as described above), and then assemble the set of new labels based on the detected events.

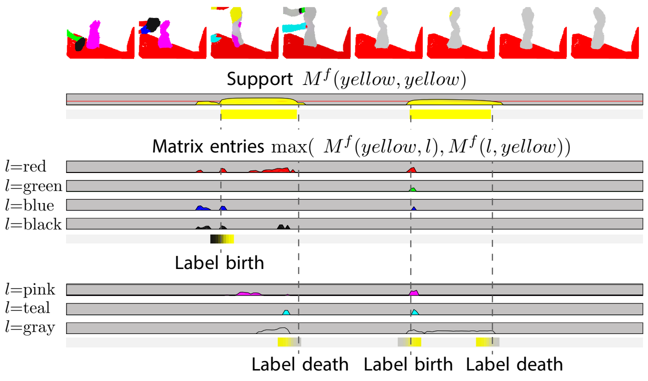


Figure 5: Visualization of the elements of the mass transfer matrices M^f involving the yellow label, and detection of birth and death events. The top plot shows the support, that is the entries $M^f(yellow, yellow)$ for all frames f along with the threshold (red line) to detect birth and death. We visualize the other entries of M^f related to the yellow label by plotting $\max(M^f(yellow,l), M^f(l,yellow))$ in the respective colors of l. We mark the largest label transfers at birth and death (black-yellow and yellow-gray for the first spatio-temporal segment, gray-yellow and yellow-gray for the second), which we use to assemble new labels.

Detecting Label Events. To detect label events including switches, births and deaths we first construct a label mass transfer matrix $M^f \in N \times N$ for each frame f, where N is the number of labels. This matrix stores for each pair of labels how many pixels change from the first to the second label from frame f to $f+1$. To compute M^f we first initialize it with zeros. Then, for each pixel in frame f, we note its label l, compute its closest neighbor in the next frame $f+1$, call its label l', and increment $M^f(l,l')$. In Figure 5 we visualize the elements of M^f related to one label (yellow) over a sequence of frames f.

We detect death of label l' in frame $f+1$ if the number of pixels assigned to l', that is $\sum_{l=1}^{N} M^f(l,l')$, drops from above to below 0.5% of all pixels per frame from f to $f+1$. Similarly, we detect label birth if the number of assigned pixels increases from below to above 0.5% from one frame to the next. Finally, we detect label switch events as the n_{switch} largest temporal local maxima in the temporal stack of the matrices M^f. More precisely, a pair of labels l,l' is a temporal local maximum in frame f if $M^{f-1}(l,l') < M^f(l,l') > M^{f+}(l,l')$, and we select the n_{switch} largest of these. We also apply a temporal box filter to the matrix stack for each matrix entry to avoid spurious local maxima.

Assembling the New Labels. Label events, including deaths, births, and switches, are indications of temporal over-segmentation, that is, 3D points on objects that are erroneously assigned different labels over time. Hence our goal is to construct a new set of labels that do not suffer from temporal over-segmentation, that is, labels that are supported (alive) during the whole sequence of frames. Death and birth events also occur when objects enter or leave the frustum, but in the crucial early stage of our optimization temporal errors are the primary source of such events. When an object truly leaves the frustum, we rely on the robustness of our energy formulation and the segmentation step to discard spurious motions.

First, we complete labels that experience birth or death events by concatenating them with other labels. For a label l with death in some frame f_0, we look up in the ten frames $M^{f \in f_0-9,\dots f_0}$ preceding its death to which label l' the largest label transfer occurred (as illustrated in Figure 5), and then append l' to l (by taking the union of their pixels). We proceed similarly with l', appending another label in case l' dies before the end of the sequence, etc. We proceed in the same way with label births, but moving backward in time. As a result, we obtain a new set of labels that all have full temporal support, and we discard the original labels. Finally, we eliminate duplicate label combinations from this set. This process is illustrated in Figure 6.

Second, we resolve label switches by constructing a new label that combines the label pair involved in each switch. If we have a label switch from l to l' in frame f_0, we construct a new label l'' as the union of l before f_0, and l' after $f0$. We keep all three labels l, l', l'' in our new set.

Finally, we generate new labels by randomly sampling a small number of $n_{outlier}$ points from the 0.5% points incurring the largest cost to Equation (1). We add a label consisting of a short, small tempo-spatial tube around each point. We use a fixed spatial radius of 15 pixels and extend the tubes over 30 frames in both directions using each points' currently assigned motion.

Note that the set of new labels we constructed here is not a segmentation of the data points. Instead, each point may occur in multiple labels. This is not an issue, since we merely use these labels to generate new motions, which we will subsequently use to minimize our segmentation energy.

6.2.2. Finding Motions by Aligning Labels

We find a motion for each label separately by geometrically aligning the points belonging to the label across the sequence of frames. This is a standard 3D registration problem. Since we want to generate motions that are likely to reduce our segmentation energy, we minimize a registration cost that is similar to the binary term in our energy. For efficiency, however, we employ simplified correspondence probabilities and registration errors compared to the original ones (Equation (4) and (5)). We follow an ICP-like [BM92] approach, where we iteratively update the correspondence probabilities and the registration errors.

Modified Correspondence Probability. We simplify the correspondence probability from Equation 4 to

$$P_{x,y}(\mathcal{M}_l) = 1_{y==NN(\mathcal{M}_l^{\to y}x)} \cdot 1_{|y-NN(\mathcal{M}_l^{\to y}x)|<3\hat{\sigma}}, \qquad (12)$$

where we changed the notation for $P_{x,y}$ to indicate that $P_{x,y}$ is a function of the motion M_l of label l. Our simplification amounts to a binary nearest neighbor correspondence, discarding the terms accounting for occlusion, normal orientation relative to the camera, and temporal distance. We ignore the correspondence, however, if the distance between the nearest neighbors is above a threshold $3\hat{\sigma}$.

Modified Registration Error. Let us factor $\mathcal{M}_l^{i\to j}$ into a transformation of frame i to a reference coordinate system, followed by the

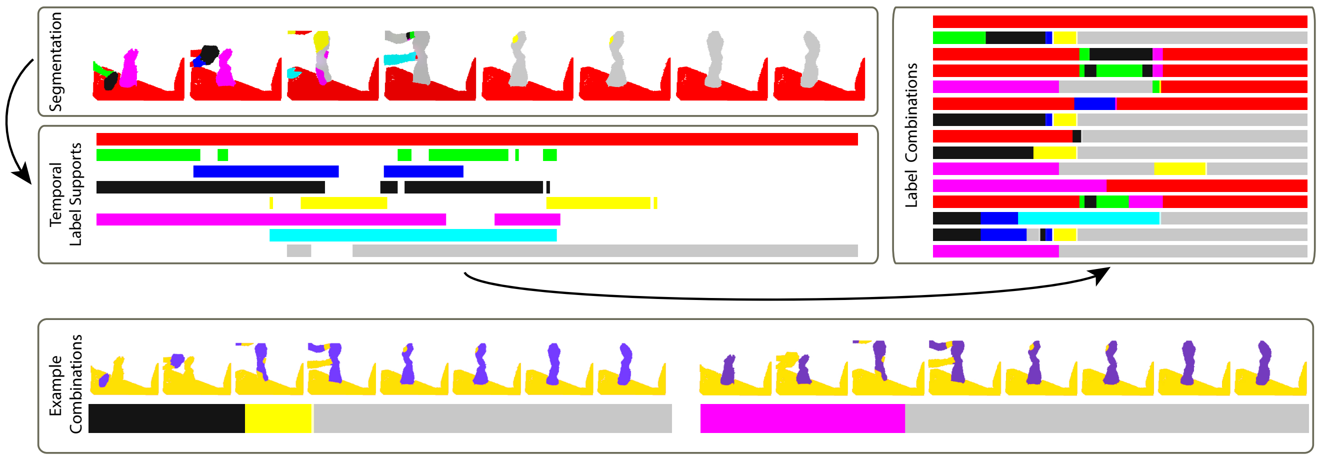


Figure 6: Visualization of the complete label assembling step. The segmentation is shown on the top left. Using the mass transfer matrix, first the temporal label supports are computed. Then the label births and deaths are resolved to get the label combinations shown on the right. Finally, based on these combinations new labels are generated, and we show two selected label combinations at the bottom of the figure. These combined labels will then be used to generate new motions which will be added to the motion set.

transformation to frame j. That is, $\mathcal{M}_l^{i\to j} = T_j T_i^{-1}$, where we omit l to simplify the notation. We now define our modified registration error as

$$error_{x,y}(\mathcal{M}_l) = \left\langle T_{y.f} T_{x.f}^{-1} x - y, y.n \right\rangle^2 + \alpha \| T_{y.f} T_{x.f}^{-1} x - y \|, \quad (13)$$

where we changed the notation for $error_{x,y}$ (Equation (5)) similar as above to indicate that the error depends on the motion M_l of label l. We abandon the incremental error (Equation 6) for simplicity, and clamping the error is not necessary because we include distance-based clamping in the modified correspondence probabilities (Equation 12). Finally, our modified registration error includes a point-to-plane distance and relative weight α for the point-to-point distance, which improves convergence in practice.

ICP Iteration. We solve for the rigid transformations $T_{x.f}$, $T_{y.f}$ by alternating between updating point correspondences (Equation (12)) and minimizing the alignment error (Equation (13)) in a Levenberg-Marquardt algorithm. For faster convergence, we solve for the transformations in a coarse-to-fine fashion by subsampling the point clouds hierarchically and using clouds of increasing density as the iteration proceeds. In addition, we consider the registration error only for point pairs (x,y) occurring in select frame pairs.

Selecting Frame Pairs. The simplest strategy is to include only point pairs from neighboring frames, $\{(x,y)|y.f = x.f+1\}$. This may be sufficient for simple scenes with large objects and slow motion. However, it suffers from drift. We enhance the incremental approach with a loop closure strategy to avoid drift similar to Zollhöfer et al. [ZDI*15]. Using temporally more distant frames mirrors the temporal window in our original motion segmentation energy. The idea is to detect non-neighboring frames that could be aligned directly, and compute the registration error from Equation (13) for points from a sparse set of such pairs. We use the following heuristics to find eligible frame pairs:

- The centroid of the observed portion of the object in frame i lies in the view frustum when mapped to frame j, and vice versa.
- The viewing direction onto the object, approximated by the direction from the camera to the centroid, should be similar in both frames. We tolerate a maximum deviation of 45 degrees.
- The distance of the centroids to the camera is similar in both frames. Currently we tolerate a maximum factor of 2.

The first two criteria are to check that similar parts of the object are visible in both frames, and they are seen from similar directions. The third one ensures that the sampling density does not differ too much. Initializing a set S with the adjacent frame constraints $(i, i+1)$, we greedily extend it with a given number of additional pairs (k,l) from the eligible set. We iteratively select and add new pairs from the eligible pairs such that they are as distant as possible from the already selected ones,

$$S \leftarrow S \cup \underset{eligible\ (k,l)}{\operatorname{argmax}} \left(\min_{(i,j)\in S} |k-i| + |j-l| \right). \quad (14)$$

Overall, for our ICP variant with loop closures, we first solve for alignments only with points from neighboring frame pairs, $\{(x,y)|y.f = x.f+1\}$, taking identity transformations as initial guesses of the alignment between adjacent frames. We then use these alignments to determine and select additional eligible loop closure constraints and do a second ICP iteration with points from the extended set of frame pairs, $\{(x,y)|(y.f, x.f) \in S\}$.

6.2.3. Removing Unnecessary Motions

In addition to constructing new labels (Section 6.2.1) and finding motions for them (Section 6.2.2), we also remove all old motions from the previous iteration step from the motion set, before we solve the motion assignment problem again (refer to Figure 1 for an overview of the process). This is crucial for performance reasons. Removing non-assigned motions from them motion set does not loosen the upper bound from Equation (7). But removing previously assigned motions may loosen the bound. The old motions,

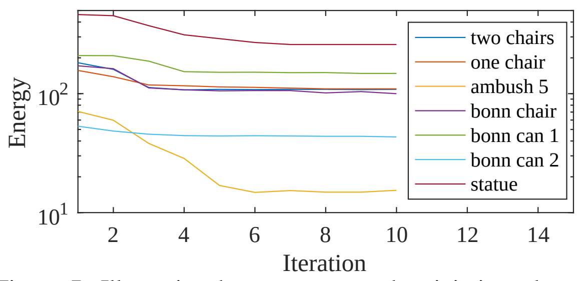


Figure 7: Illustration how our approach minimizes the energy (Equation (1)) throughout the iteration. Note that the process is occasionally non-monotonic. One reason is that the CRF step is not guaranteed to converge to the global minimum. Further, our approach to discard motions from the motion set (Section 6.2.3) may lead to a loosening of the upper bound, yet our procedure robustly minimizes the energy.

however, are typically redundant with new ones that we found for the new labels. In practice our method still continuously reduces the energy (Equation (1)) over the iterations as shown in Figure 7.

6.3. Initialization

We start our minimization (Figure 1) with a single motion, namely the motion obtained by registering all frames with the registration method described in Section 6.2.

7. Implementation and Results

7.1. Parameters, Implementation, and Runtimes

We use the same parameters in all our experiments but choose sensor dependent noise models. We fix the three free parameters in the energy definition (Section 5) to $cost_{label} = 1.005$ (remove labels if the energy is increased by less than half a percent), $d = 0.1$ (bias in unary term) and $\sigma_{temp} = 30$ (spatial window for correspondence probability). To construct new labels (Section 6.2.1), we set $n_{switch} = n_{outlier} = 5$ and limit the number of motions to 32. When aligning the motions (Section 6.2.2) we set the balance between the point-to-point and point-to-plane term to $\alpha = 0.1$ (Eq. (13)). We apply at most 100 ICP iterations or stop upon convergence. In the motion assignment step (Section 6.1) we set the number of CRF iterations to 15 and interpolate the update equation evaluated on 10% of the points. We always perform 10 main iterations. We discuss the influence of the parameters and the noise model in Section 7.1.1.

Overall the complexity of our method is linear in the number of frames and quadratic in the number of motions. In theory, the update of the mean-field inference (Equation 10) is quadratic in the number of frames, as for each point in each frame points from all other frames contribute to the update. In practice, due to the exponential decay of the correspondence probabilities with respect to the temporal distance (Eq. (4)), we evaluate the update equation on a fixed temporal window around each frame. The quadratic complexity in the number of motions stems from the greedy merging of segments (Section 6.1.3), where we consider all pairs.

OURS	Motion Set	Assignment	EG16	IROS16
220	128	92	418	268

Table 1: The runtimes of our method, as well as the ones by Qing et al. [YLX*16] (EG16) and Kim et al. [KLAK16] (IROS16), in seconds per frame, averaged over all sequences in our quantitative evaluation. We break down our times into the refinement of the motion set (Section 6.2) and the motion assignment (Section 6.1).

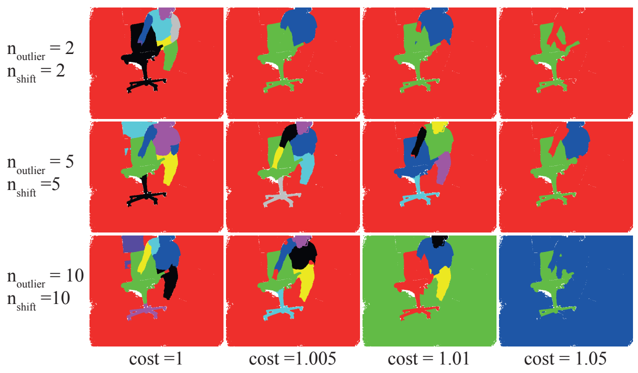


Figure 8: Influence of label costs $cost_{label}$ and hypothesis generation $n_{outlier}, n_{shifts}$ on the method's final result after 10 iterations. Increasing $cost_{label}$ leads to a coarser segmentation while increasing n_{switch}, $n_{outliers}$ will allow the method to find segments more robustly and in less iterations as more hypotheses are tested. On the other hand the quality of the inference step slightly decreases as the number of hypotheses increases. With $cost_{label} = 1$ the method works well but results in oversegmentations when geometry is ill connected, a value slightly larger than one corrects this.

We implemented our approach in C++ and run all steps on the CPU, except for CRF inference (Section 6.1.1) and the evaluation of the energy in the segment merging step (Section 6.1.3), which are implemented on the GPU. We report runtimes of our approach and the global methods by Qing et al. [YLX*16] and Kim et al. [KLAK16] in Table 1. Our parallel CRF inference is quite fast - the pairwise terms in the update Equation 11 are evaluated and summed up at about 2700fps on our NVIDIA Titan GPU, including nearest neighbor search. Using temporal windows of 60 frames, (in average) 15 motions, 15 CRF iterations and 10 overall iterations this results in an average computation time of 45s. Including the merge step, which takes about the same time, we get the reported 92s to optimize motion assignments. The rest of our code is not optimized for speed and leaves a lot of room for improvement. For example, registering a single label (Section 6.2.2) runs only at 0.5 fps (which is the current bottleneck). We make our code and data available at: http://www.cgg.unibe.ch/publications/rgb-d-motion-segmentation/project-page.

7.1.1. Influence of Parameters, Noise Model

In general the method is relatively robust to the parameter choices. The parameters $cost_{label}$ and n_{switch}, $n_{outliers}$ control the label generation and removal; their joint influence is visualized in Figure 8. Our parameter choice is a trade-off between under- and oversegmentation and the stability of our method.

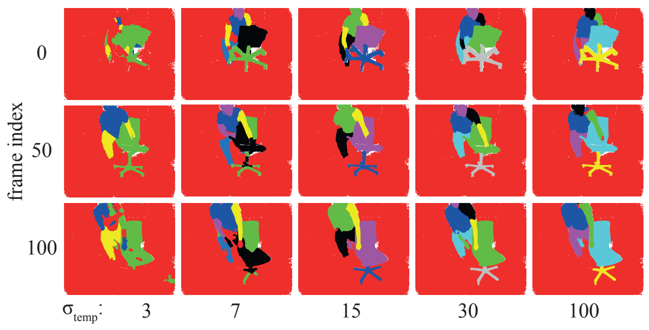


Figure 9: Results for varrying σ_{temp}. Increasing σ_{temp} yields better temporal consistency and improves the results, as more frames are taken into account.

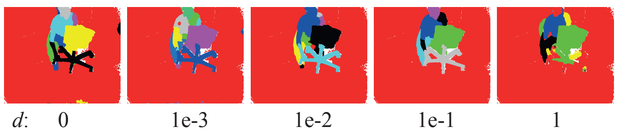


Figure 10: Setting the bias d (dataterm) to zero leads to artifacts in difficult regions (close to the boundary, behind the person's head), as motions are favored which explain the points as being occluded as often as possible. These artifacts are fixed with a small bias. Large values will break the method, as free-space violation becomes cheaper than occlusion.

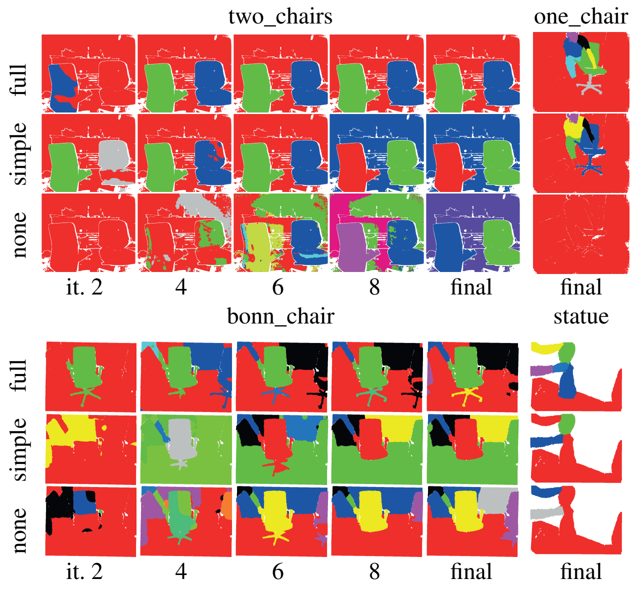


Figure 11: Results with our proposed noise model $\hat{\sigma}$ (full), with a simplified noise model $max(\overline{\sigma}, \tau)$ (simple) and without any adaptive noise model $\overline{\sigma}$ (none). On two_chairs and bonn_chair the intermediate results at iterations 2, 4, 6 and 8 are shown alongside the final result. Using the full model improves the convergence rate and the final results.

Increasing the parameter σ_{temp} allows assignments to take more frames into account and leads to increased temporal consistency, at the expense of computation time. We did not observe benefits of using $\sigma_{temp} > 30$; we qualitatively show the influence of the parameter in Figure 9. The exact choice of d (bias) does not matter much. Setting it to zero still yields sensible results, with a preference to explaining observations as being visible only once and hidden else on difficult regions. This behavior is fixed when setting d between $1e-3$ to $1e-1$. When set higher the method degrades, as free-space violation becomes cheaper than potential occlusion, see Figure 10. The parameters related to registration (α, number of iterations, exact choice of registration energy) were chosen in an ad-hoc manner to produce relieable, robust registration results. Note that motion estimation could, in principle, be replaced with any robust registration method.

The noise and sensor model is central for our method to perform robustly over a diverse set of scenes and sensors and effectively encapsulates sensor specific parameters. To evaluate the influence of the noise model we compare our method to two base-lines, using either a simplified adaptive or no adaptive noise model. We obtain the first by replacing $\hat{\sigma} = \max(\tau, \sigma)$ by $max(\overline{\sigma}, \tau)$, where $\overline{\sigma}$ is the mean value of σ, computed once per sensor. For the second, we replace the complete noise model $\hat{\sigma}$ by the constant $\overline{\sigma}$. We qualitatively compare the models in Figure 11 and quantitatively demonstrate the benefit of the full model in Section 7.2. Note that in average our simplified baselines also outperform our competitors.

7.2. Quantitative Evaluation

We compare our method to the two recent techniques by Qing et al. [YLX*16] (abbreviated EG16) and Kim et al. [KLAK16] (IROS16) using seven sequences described in Table 2. The scenes of varying complexity are recorded with three different sensors. They feature up to twelve near-rigid and rigid objects with rotational and translational motion, and a moving camera. Rotations are typically harder to handle as they result in challenging self-occlusions. We recorded one_chair, two_chairs, and statue ourselves; bonn_chair, bonn_can1, and bonn_can2 are from data shared by Stückler and Behnke [SB15], and ambush5 is from the Sintel dataset [BWSB12].

We run our method and EG16 on temporally subsampled data at about 10fps. For IROS16 we use all frames as we observed a performance drop with subsampling, while EG16 exhibits a large performance gain from subsampling. The only exception is the statue scene, where we needed to run IROS16 with subsampling as their code would not complete within 24 hours otherwise.

For a quantitative evaluation we manually annotated all sequences with ground-truth segmentations. Our goal is to evaluate the segmentation of independently moving *rigid* objects and object parts, and how well this generalizes to *near-rigid* parts such as arms and legs. Hence we also classify each ground-truth label as either rigid or near-rigid. We define each ground-truth label using a ternary mask including a set of *positive*, *negative*, and *optional* positive pixels. We show an example ground-truth segmentation for the one_chair sequence in Figure 12. While Stückler and Behnke [SB15] already provide ground-truth segmentations on a

Sequence	Frames			Labels			Sensor	
	num	used	gt	num	rigid	type	type	motion
one_chair	100	(50)	8	12	(3)	R+T	Kin1	static
two_chairs	60	(60)	5	3	(3)	R+T	Kin1	static
statue	660	(220)	16	5	(3)	R+T	Asus	static
ambush5	20	(20)	5	9	(2)	T	Synt	moving
bonn_chair	180	(60)	6	4	(3)	T	Asus	moving
bonn_can1	180	(60)	5	6	(3)	T	Asus	moving
bonn_can2	60	(20)	4	3	(2)	T	Asus	moving

Table 2: Summary of the sequences used for evaluation. The *Frames* column lists the number of frames (*num*), the number of frames used in our method after temporally subsampling to about 10fps (*used*), and the number of frames with ground-truth annotations (*gt*). The *Labels* column lists the number of different rigid and near-rigid segments (*num*), the number of rigid segments (*rigid*), and their dominant type of motion (*type*), which is rotational (*R*) or translational (*T*). Last, we list the acquisition sensor, which are the Kinect for Xbox One (*Kin1*), the Asus Xtion Pro (*Asus*), or a synthetic sensor (*Synt*) for the synthetically rendered ambush5 scene, and whether the sensor is static or moving.

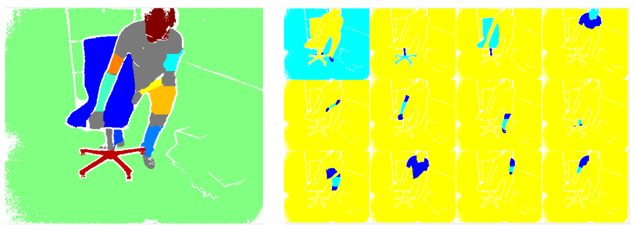


Figure 12: Visualization of ternary ground-truth masks for our evaluation. On the left we show the *positive* masks for all labels in different colors, and we draw pixels that do not occur in any *positive* mask in dark gray. On the right we show the ternary masks for each label, where *positive* pixels are in teal, *negative* pixels in yellow, and *optional* pixels in blue.

few frames, we use our annotations instead for consistency. The ternary masks with optional positive pixels for each label allow us to obtain a more fine-grained evaluation, compared to masking ambiguous pixels as "do not care" for all labels.

Given a ternary ground-truth mask m with values $positive, optional$, and $negative$, and a binary mask b representing a single label found by one of the methods (OURS, EG16, or IROS16), we compute true positives $tp = |b\&(m = true|m = optional)|$, true negatives $tn = |!b\&(m = false|m = optional)|$, false positives $|b\&(m = false)|$, and false negatives $|!b\&(m = true)|$, where the Boolean operations are applied to each pixel, and $|.|$ counts the number of true pixels. From this we compute the accuracy $tp/(tp+fp+fn)$, precision $tp/(tp+fp)$, and F-score as usual.

To evaluate the segmentation of a frame, we first need to determine the correspondences between the found labels and the ground-truth labels. Similar to Mahmood et al. [MDSL17], we choose the correspondences that maximize the F-score. We then divide the sum of the per label scores by the number of ground truth labels. To compare the performance of each method when only rigid objects,

	EG16			IROS16			OURS		
	A	P	F	A	P	F	A	P	F
one_chair	0.19	0.26	0.23	0.22	0.30	0.26	**0.58**	**0.6**	**0.61**
	0.56	0.79	0.67	0.45	0.61	0.54	**0.93**	**0.94**	**0.96**
two_chairs	0.93	**1.00**	0.96	0.28	0.52	0.33	**0.96**	0.98	**0.98**
	0.93	**1.00**	0.96	0.28	0.52	0.33	**0.96**	0.98	**0.98**
statue	0.70	0.81	0.78	0.37	0.41	0.41	**0.88**	**0.89**	**0.90**
	0.74	0.86	0.83	0.61	0.67	0.67	**0.94**	**0.95**	**0.97**
ambush5	0.42	0.53	0.52	0.28	0.3	0.32	**0.92**	**0.94**	**0.95**
	0.6	0.83	0.75	0.57	0.59	0.64	**0.96**	**0.98**	**0.98**
bonn_chair	0.53	0.75	0.62	0.56	0.69	0.63	**0.83**	**0.92**	**0.9**
	0.45	0.68	0.53	0.49	0.65	0.55	**0.80**	**0.91**	**0.88**
bonn_can1	0.21	0.35	0.28	0.51	0.61	0.57	**0.96**	**0.97**	**0.98**
	0.42	0.7	0.56	0.82	0.90	0.90	**0.95**	**0.97**	**0.98**
bonn_can2	0.40	0.66	0.52	**0.89**	0.94	**0.94**	0.83	**0.96**	0.90
	0.20	0.59	0.33	**0.92**	**0.98**	**0.96**	0.8	**0.98**	0.88
Average	0.48	0.62	0.56	0.44	0.54	0.49	0.85	0.89	0.89
	0.56	0.78	0.66	0.59	0.70	0.66	0.91	0.96	0.95

Table 3: Acccuracy (A), Precision (P) and F-Score (F) for the methods by Qing et al. [YLX*16] (EG16), Kim et al. [KLAK16] (IROS16) and our method (OURS). For each scene we report the metrics evaluated on the complete scene featuring both rigid and near-rigid parts, like arms, heads, etc. (first row), as well as the metrics evaluated only on the rigid objects (second row).

or rigid and near-rigid objects are considered, we evaluate every scene twice: once using all the ternary masks including rigid and near-rigid objects, and once using only the absolutely rigid parts.

We summarize the quantitative results in Table 3, where we report accuracy, precision, and F-score, both in the purely rigid case, and including rigid and near-rigid objects. Stückler and Behnke [SB15] and Kim et al. [KLAK16] also report accuracy, while precision and F-score are more commonly otherwise. In summary, this quantitative evaluation shows that we outperform both comparison methods by a large margin.

We evaluate the effectiveness of our proposed noise model in the same way in Table 4, by comparing our full method against a baseline with either a simplified adaptive noise model or without any adaptive noise model, as described in Section 7.1.1. In summary, the full method generalizes best to all scenes and sensors and clearly outperforms the two baselines. Note that in average our simplified baselines also outperform our competitors.

7.3. Discussion

The results of all methods and all sequences are visualized in Figures 15 to 18. Our method excels on the bonn_can1 (Figure 15), two_chairs (Figure 17), ambush5 (Figure 16) and statue sequence (Figure 18), both on the rigid and the near-rigid portion of the scenes. These scenes feature highly non-uniform noise (two_chairs) complex interactions (bonn_can1, statue, ambush5) and up to nine near-rigid objects (ambush5). On the one_chair sequence we consistently segment all rigid parts and find a reasonable under-segmentation of the man featured in the sequence (Figure 17). A limitation of our method can be observed on the bonn_can2 and the bonn_chair sequence (Figures 15 and 16). Our

	NONE			SIMPLE			FULL		
	A	P	F	A	P	F	A	P	F
one_chair	0.07	0.07	0.07	0.42	0.43	0.45	**0.58**	**0.6**	**0.61**
	0.27	0.27	0.30	0.58	0.58	0.62	**0.93**	**0.94**	**0.96**
two_chairs	**0.98**	**1.00**	**0.99**	**0.98**	**1.00**	**0.99**	0.96	0.98	0.98
	0.98	**1.00**	**0.99**	**0.98**	**1.00**	**0.99**	0.96	0.98	0.98
statue	0.54	0.55	0.57	0.55	0.56	0.59	**0.88**	**0.89**	**0.90**
	0.32	0.32	0.37	0.51	0.51	0.56	**0.94**	**0.95**	**0.97**
ambush5	0.26	0.33	0.35	**0.92**	**0.94**	**0.95**	**0.92**	**0.94**	**0.95**
	0.69	0.70	0.81	**0.96**	**0.98**	**0.98**	**0.96**	**0.98**	**0.98**
bonn_chair	0.55	0.73	0.62	0.65	0.73	0.69	**0.83**	**0.92**	**0.9**
	0.51	0.65	0.57	0.55	0.65	0.60	**0.80**	**0.91**	**0.88**
bonn_can1	0.77	0.78	0.80	0.77	0.78	0.80	**0.96**	**0.97**	**0.98**
	0.91	0.92	0.95	0.89	0.90	0.94	**0.95**	**0.97**	**0.98**
bonn_can2	**0.85**	0.94	**0.92**	**0.85**	0.95	**0.92**	0.83	**0.96**	0.90
	0.86	0.97	**0.92**	0.85	0.98	**0.92**	0.8	**0.98**	0.88
Average	0.57	0.63	0.62	0.74	0.77	0.77	0.85	0.89	0.89
	0.65	0.69	0.70	0.76	0.80	0.80	0.91	0.96	0.95

Table 4: Acccuracy (A), Precision (P) and F-Score (F) using our method without an adaptive noise model (NONE), with a simplified noise model (SIMPLE) and with our full noise model (FULL), reported as in Table 3.

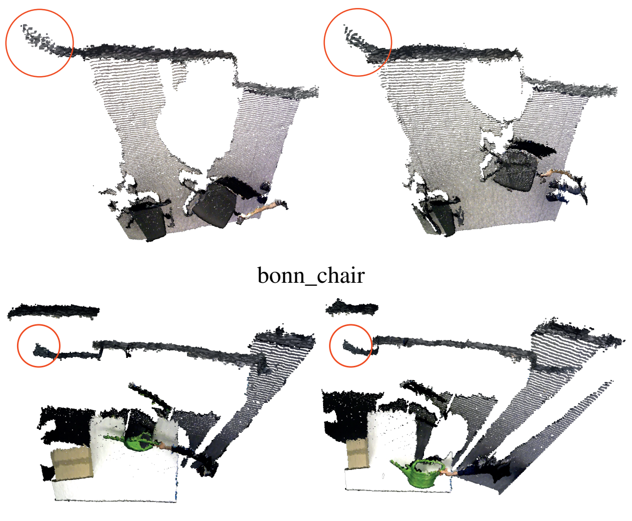


Figure 13: Top view on selected frames from the bonn_chair and the bonn_can2 sequence. Aberrations in the observed depth are very pronounced in the bonn_chair sequence, and also present in the bonn_can sequence. Together with camera motion they cause our method to over-segment.

method robustly finds all rigid parts but over-segments the background. This is due to strong aberrations in the recorded depth data: as the camera moves our method explains the aberrations as an additional moving object. We document the aberrations in Figure 13. Leveraging the assigned motions, our approach also allows us to accumulate the point clouds and perform 3D reconstruction, as shown in Figure 14.

The IROS16 method works very well on scenes with few objects and non-rotational motions, such as the bonn_can1 and bonn_can2 scene (Figure 15). IROS16 quickly fails in the presence of occlusions because it assumes the availability of complete point trajectories throughout the sequence. Hence it does not perform well on the ambush5, one_chair, two_chair and statue scene. Also IROS16 does not generalize well to the near-rigid scene parts. The near-rigid motions and the non-uniformity of noise pose a challenge to the employed probabilistic subspace clustering. The brittleness of the method in the presence of occlusions was noted by Kim et al. in the original paper [KLAK16], where the method was demonstrated to work well on scenes with up to three (near)-rigid objects.

The EG16 method by Qing et al. [YLX*16] handles scenes with many motions and little occlusion well, such as the synthetic ambush5 scene (Figure 16) and the statue scene (Figure 18). More pronounced occlusion leads to the erroneous establishment of correspondences, resulting in over-segmentations. At the same time EG16 heavily relies on spatial smoothness to reduce over-segmentation in their intermedidate single-frame results. When occlusions and self-occlusions lead to objects appearing separated in individual frames, this often results in spatial over-segmentation in the final results (see Figures 15, 16 and 17). Note that the method works well on the two_chair scene (Figure 17) despite self-occlusions and highly non-uniform noise, as the single objects stay spatially connected throughout the sequence.

In summary, we clearly benefit from our explicit occlusion model, and our energy minimization strategy is able to find dominant rigid and near-rigid motions robustly. A further reason for the performance gap between EG16, IROS16, and our method is that EG16 and IROS16 rely on very sparse sampling, as both methods have quadratic complexity in the number of samples per frame. Both methods compute a segmentation for about 2-5% of the points only, while we solve for a dense segmentation, even though we approximate a sub-step of CRF inference on 10% of the points.

7.4. Limitations

While our technique finds segmentations and rigid motions more robustly than previous methods and it generalizes well to near-rigid parts, it still has a number of limitations.

- Computation time is on the order of three to four minutes per frame. We share this limitation with the global approaches of our two competitors. Hence, all three methods cannot handle full videos with thousands of frames. We believe that a variant of our formulation could be a building block for large sequences, for example by relying on key frames and by using alternative, faster approaches to augment the motion set.
- Our formulation is not resilient to aberrations in depth data. This could be reduced by additionally relying on semantic features rather than on geometry alone.
- We share certain limitations with ICP, which may not be able to align objects when they undergo large motions between frames. Also, objects that leave the camera frustum or are completely occluded for several frames cannot be handled correctly. We would need additional strategies to augment the motion set in both cases.

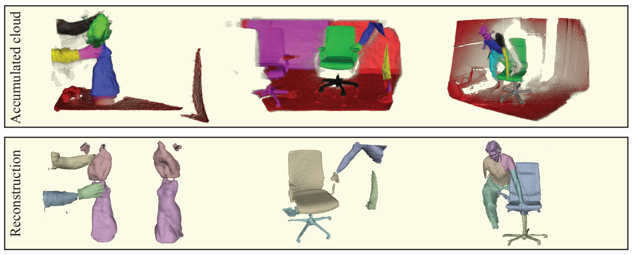


Figure 14: From left to right: accumulated point clouds and selected reconstructions for the statue, bonn_chair, and our one_chair sequence. The accumulated point clouds were created by mapping all observed data points to the central frame.

- Our method will fail for extremely non-rigid objects like cloth, water, or similar.

8. Conclusions

We presented a novel method for temporally consistent motion segmentation from RGB-D videos. Our approach is based almost entirely on geometric information, which is advantageous in scenes with little texture or strong appearance changes. We demonstrated successful results on scenes with complex motion, where object parts sometimes move in parallel over parts of the sequences, and their motion trajectories may split or merge at any time. Even in these challenging scenarios we obtain consistent labelings over the entire sequences, thanks to a global energy minimization over all input frames. Our approach includes two key technical contributions: first, a novel formulation of a motion segmentation energy which incorporates occlusion and sensor noise in a principled way, second a minimization framework for the energy, which allows the principled use of powerful heuristics to find motions.

References

[BBFF16] Baqué P., Bagautdinov T., Fleuret F., Fua P.: Principled parallel mean-field inference for discrete random fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 5848–5857. 6

[BFF16] Baqué P., Fleuret F., Fua P.: Multi-modal mean-fields via cardinality-based clamping. *arXiv preprint arXiv:1611.07941* (2016). 6

[BM92] Besl P. J., McKay N. D.: A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (Feb 1992), 239–256. doi:10.1109/34.121791. 8

[BWSB12] Butler D. J., Wulff J., Stanley G. B., Black M. J.: A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV)* (Oct. 2012), A. Fitzgibbon et al. (Eds.), (Ed.), Part IV, LNCS 7577, Springer-Verlag, pp. 611–625. 11, 15

[DFI*15] Dosovitskiy A., Fischery P., Ilg E., HÃďusser P., Hazirbas C., Golkov V., v. d. Smagt P., Cremers D., Brox T.: Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec 2015), pp. 2758–2766. doi:10.1109/ICCV.2015.316. 2

[FAFM15] Fragkiadaki K., Arbelaez P., Felsen P., Malik J.: Learning to segment moving objects in videos. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on* (June 2015), pp. 4083–4090. doi:10.1109/CVPR.2015.7299035. 2

[FRP09] Fradet M., Robert P., PÃľrez P.: Clustering point trajectories with various life-spans. In *Visual Media Production, 2009. CVMP '09. Conference for* (Nov 2009), pp. 7–14. doi:10.1109/CVMP.2009.24. 2

[HRF13] Herbst E., Ren X., Fox D.: Rgb-d flow: Dense 3-d motion estimation using color and depth. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (May 2013), pp. 2276–2282. doi:10.1109/ICRA.2013.6630885. 2

[IKH*11] Izadi S., Kim D., Hilliges O., Molyneaux D., Newcombe R., Kohli P., Shotton J., Hodges S., Freeman D., Davison A., Fitzgibbon A.: Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2011), UIST '11, ACM, pp. 559–568. URL: http://doi.acm.org/10.1145/2047196.2047270, doi:10.1145/2047196.2047270. 1, 2

[JSS*15] Jaimez M., Souiai M., Stuckler J., Gonzalez-Jimenez J., Cremers D.: Motion cooperation: Smooth piece-wise rigid scene flow from rgb-d images. In *3D Vision (3DV), 2015 International Conference on* (2015), IEEE, pp. 64–72. 2

[KLAK16] Kim Y., Lim H., Ahn S. C., Kim A.: Simultaneous segmentation, estimation and analysis of articulated motion from dense point cloud sequence. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Oct 2016), pp. 1085–1092. doi:10.1109/IROS.2016.7759184. 2, 10, 11, 12, 13, 15

[MDSL17] Mahmood M. H., Díez Y., Salvi J., Lladó X.: A collection of challenging motion segmentation benchmark datasets. *Pattern Recognition 61* (2017), 1–14. 12

[MGC*15] Ma L., Ghafarianzadeh M., Coleman D., Correll N., Sibley G.: Simultaneous localization, mapping, and manipulation for unsupervised object discovery. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on* (May 2015), pp. 1344–1351. doi:10.1109/ICRA.2015.7139365. 1

[MS14] Ma L., Sibley G.: Unsupervised dense object discovery, detection, tracking and reconstruction. In *Computer Vision âĂŞ ECCV 2014*, Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), vol. 8690 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 80–95. URL: http://dx.doi.org/10.1007/978-3-319-10605-2_6, doi:10.1007/978-3-319-10605-2_6. 2

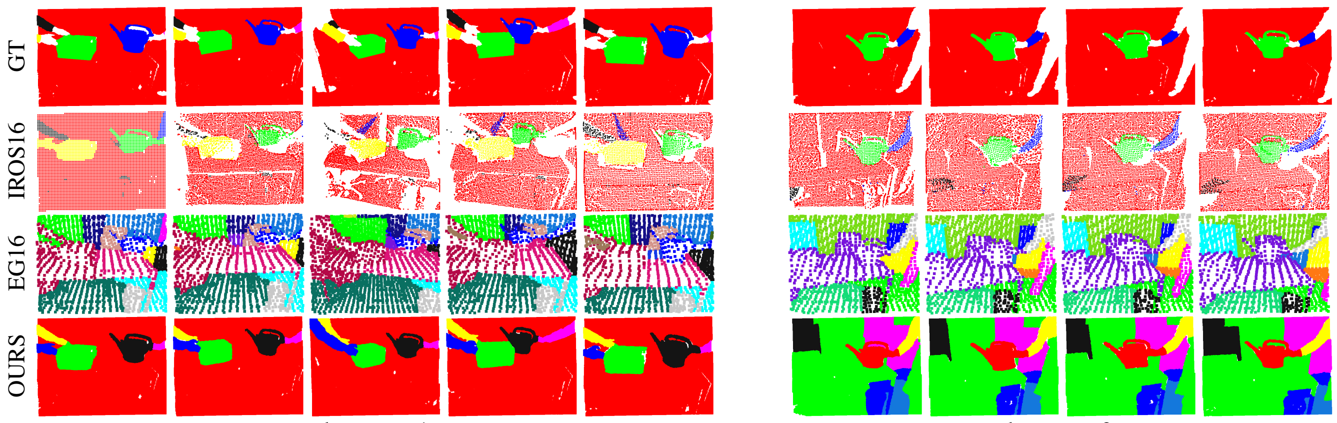
[NA98] Nourani Y., Andresen B.: A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General 31*, 41 (1998), 8373. 6

[OMB14] Ochs P., Malik J., Brox T.: Segmentation of moving objects by long term video analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 36*, 6 (June 2014), 1187–1200. doi:10.1109/TPAMI.2013.242. 2

[PB11] Papazov C., Burschka D.: Deformable 3d shape registration based on local similarity transforms. *Computer Graphics Forum 30*, 5 (2011), 1493–1502. URL: http://dx.doi.org/10.1111/j.1467-8659.2011.02023.x, doi:10.1111/j.1467-8659.2011.02023.x. 2

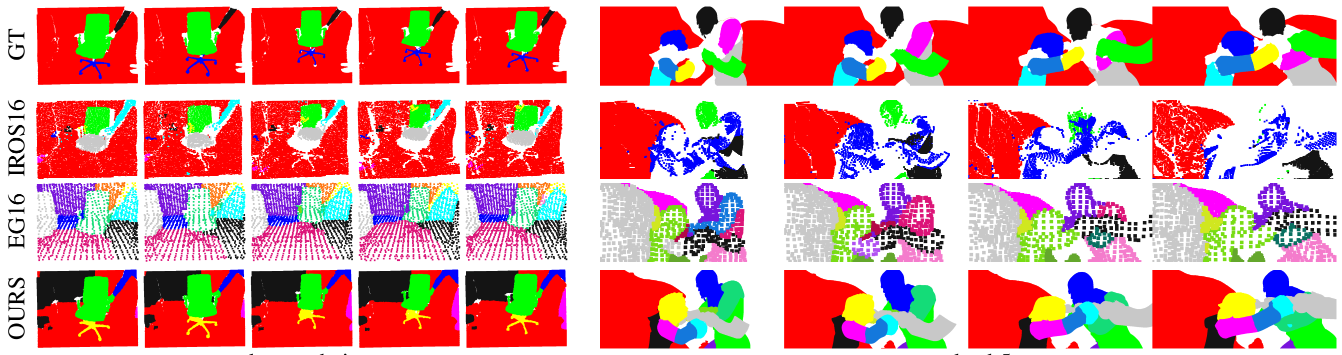
[PBH*15] Perera S., Barnes N., He X., Izadi S., Kohli P., Glocker B.: Motion segmentation of truncated signed distance function based volumetric surfaces. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on* (Jan 2015), pp. 1046–1053. doi:10.1109/WACV.2015.144. 2

[QBDC14] Quiroga J., Brox T., Devernay F., Crowley J.: Dense semi-rigid scene flow estimation from rgbd images. In *Computer Vision âĂŞ ECCV 2014*, Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), vol. 8695 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 567–582. URL: http://dx.doi.org/10.1007/978-3-319-10584-0_37, doi:10.1007/978-3-319-10584-0_37. 2



bonn_can1 bonn_can2

Figure 15: Results on a sub-sequence of the bonn_can1 and bonn_can2 scene of Stückler and Behnke [SB15]. From top to bottom: visualization of the ground-truth, results obtained by IROS16 [KLAK16], EG16 [YLX*16], and our method. The colors in the ground-truth indicate the *positive* pixels in the ternary masks of the ground-truth labels. White pixels are not marked as *positive* in any ground-truth label.



bonn_chair ambush5

Figure 16: Results on the bonn_chair of Stückler and Behnke [SB15] and the ambush5 sequences from the Sintel dataset [BWSB12]. *bonn_chair*: due to aberrations in the recorded depth the sequence is over-segmented. Our method is the only to correctly segment the swiveling base of the chair. *ambush5*: our method works robustly despite the presence of non rigid motion. We successfully segment most dominant objects such as the main body parts (torso, arms, head) and the halberd. Note how, despite the heavy occlusion by the halberd, the left arm of the tall man is correctly segmented.

[SB15] Stückler J., Behnke S.: Efficient dense rigid-body motion segmentation and estimation in rgb-d video. *International Journal of Computer Vision 113*, 3 (2015), 233–245. URL: http://dx.doi.org/10.1007/s11263-014-0796-3, doi:10.1007/s11263-014-0796-3. 2, 11, 12, 15

[SLSJB16] Sevilla-Lara L., Sun D., Jampani V., Black M. J.: Optical flow with semantic segmentation and localized layers. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 3889–3898. doi:10.1109/CVPR.2016.422. 2

[SOD12] Saito M., Okatani T., Deguchi K.: Application of the mean field methods to mrf optimization in computer vision. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 1680–1687. 5, 6

[SSP15] Sun D., Sudderth E. B., Pfister H.: Layered rgbd scene flow estimation. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on* (June 2015), pp. 548–556. doi:10.1109/CVPR.2015.7298653. 2

[TYB16] Tsai Y. H., Yang M. H., Black M. J.: Video segmentation via object flow. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 3899–3908. doi:10.1109/CVPR.2016.423. 2

[vdVRT10] van de Ven J., Ramos F., Tipaldi G.: An integrated probabilistic model for scan-matching, moving object detection and motion estimation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (May 2010), pp. 887–894. doi:10.1109/ROBOT.2010.5509586. 2

[YLX*16] Yuan Q., Li G., Xu K., Chen X., Huang H.: Space-time co-segmentation of articulated point cloud sequences. *Computer Graphics Forum 35*, 2 (2016), 419–429. URL: http://dx.doi.org/10.1111/cgf.12843, doi:10.1111/cgf.12843. 2, 10, 11, 12, 13, 15

[ZDI*15] Zollhöfer M., Dai A., Innmann M., Wu C., Stamminger M., Theobalt C., Niessner M.: Shading-based refinement on volumetric signed distance functions. *ACM Trans. Graph. 34*, 4 (July 2015), 96:1–96:14. URL: http://doi.acm.org/10.1145/2766887, doi:10.1145/2766887. 9

[ZJRP*15] Zheng S., Jayasumana S., Romera-Paredes B., Vineet V., Su Z., Du D., Huang C., Torr P. H.: Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1529–1537. 5

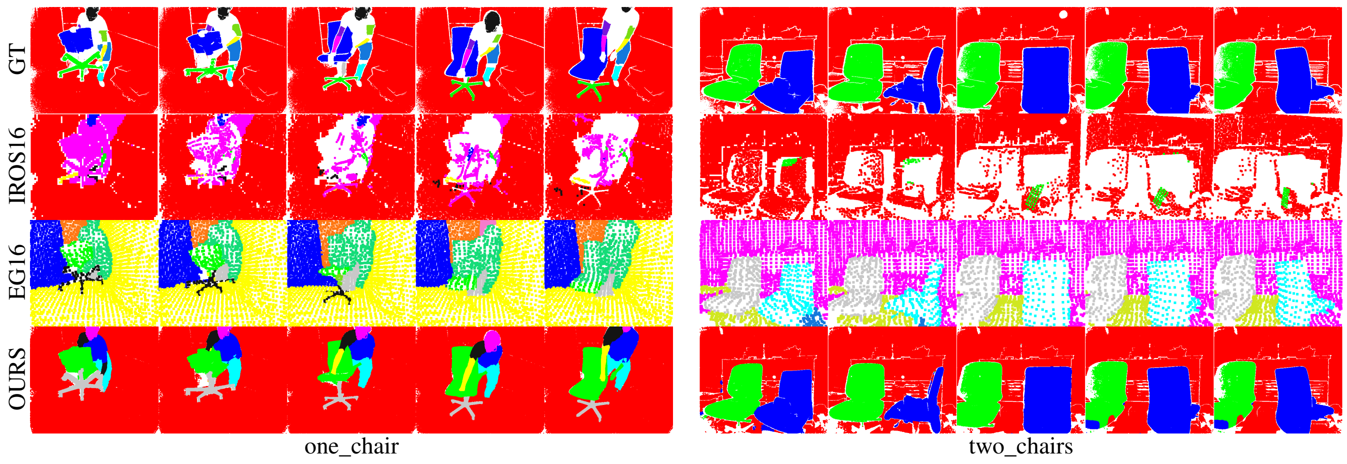


Figure 17: Results on the one_chair and two_chairs sequences. *one_chair:* we clearly outperform both IROS16 and EG16. Note how we correctly segment the swiveling base of the chair. In addition, we extract the most dominant near-rigid parts (legs, right shoulder, right arm, head, torso), but we miss some parts like the left upper and lower arm. *two_chairs:* EG16 segments the two seats slightly better than our method but over-segments the floor from the wall.

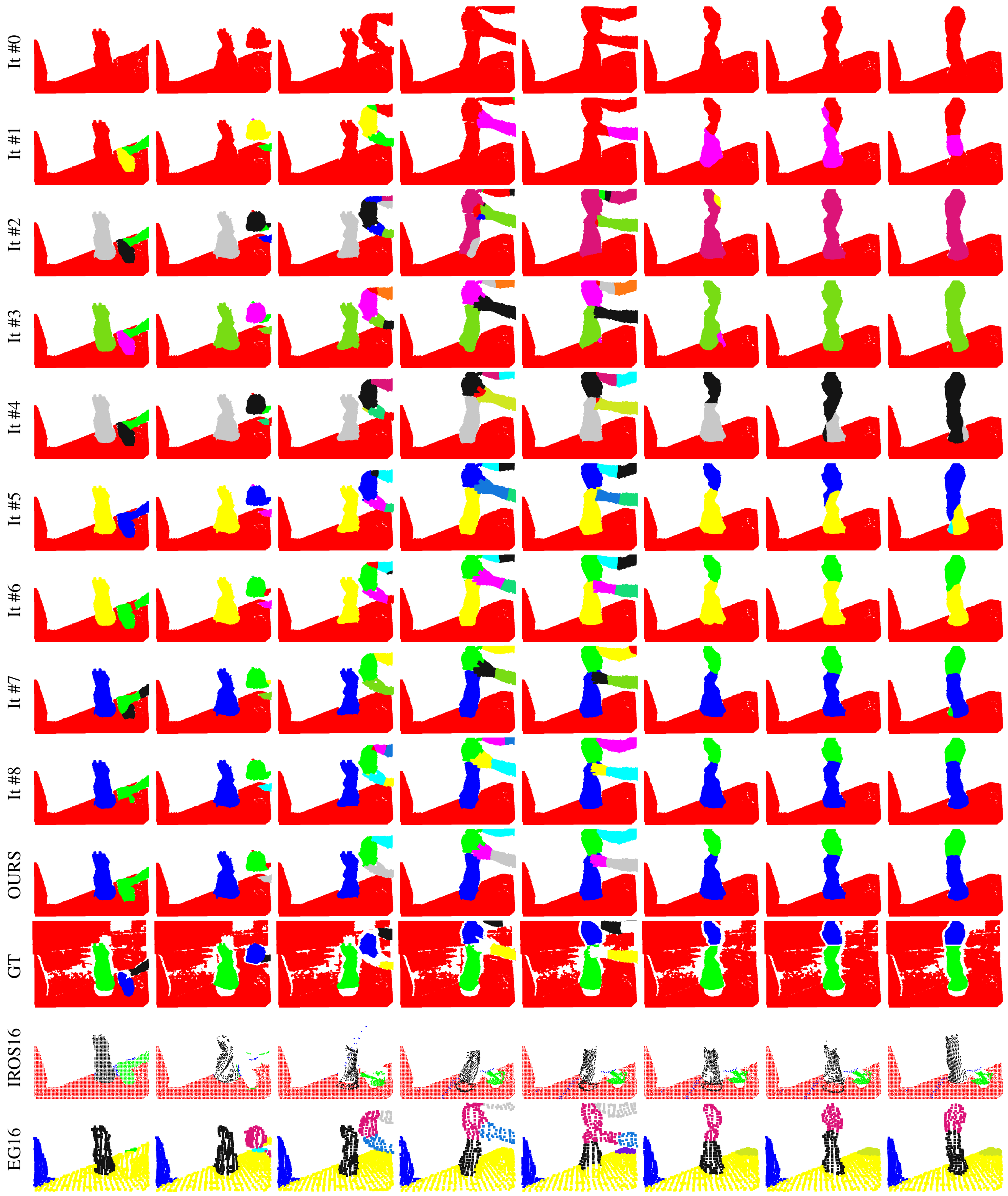


Figure 18: Results on the statue sequence. This figure shows our segmentation results throughout the iterations of our optimization. The temporal consistency and spatial correctness improve steadily and the method is robust to the presence of nonrigid objects (arms, hands).