NON-RIGID MULTI-BODY TRACKING IN RGBD STREAMS

K. X. Dai[1], H. Guo[1]*, P. Mordohai[2], F. Marinello[3], A. Pezzuolo[3], Q. L. Feng[1], Q. D. Niu[1]

[1] College of Land Science and Technology, China Agricultural University,
Beijing 100083, China - (dxdkx, guohaolys, fengql, niuquandi)@cau.edu.cn
[2] Department of Computer Science, Stevens Institute of Technology, NJ 07030, USA - mordohai@cs.stevens.edu
[3] Department of Land, Environment, Agriculture and Forestry, University of Padova, Legnaro(PD) 35020,
Italy - (francesco.marinello, andrea.pezzuolo)@unipd.it

KEY WORDS: Non-rigid, Multi-body tracking, RGBD, Point clouds

ABSTRACT:

To efficiently collect training data for an off-the-shelf object detector, we consider the problem of segmenting and tracking non-rigid objects from RGBD sequences by introducing the spatio-temporal matrix with very few assumptions - no prior object model and no stationary sensor. Spatial temporal matrix is able to encode not only spatial associations between multiple objects, but also component-level spatio temporal associations that allow the correction of falsely segmented objects in the presence of various types of interaction among multiple objects. Extensive experiments over complex human/animal body motions with occlusions and body part motions demonstrate that our approach substantially improves tracking robustness and segmentation accuracy.

1. INTRODUCTION

The emergence of modern, affordable and accurate RGBD sensors as one of several standard visual sensor systems for intelligent robots has promoted the development of point cloud processing technology and related applications such as 3d reconstruction (Zollhöfer et al., 2018), object pose (Choi and Christensen, 2016) and tracking of human/animals behavior (Xue et al., 2016, Matthews et al., 2017). The present paper introduces the possibility of propagating non-rigid object segmentation through time. We do not assume the presence of a prior class model (i.e. as the Kinect models human joint angles), as that would preclude the system from segmenting and tracking arbitrary objects of interest. Similarly, we do not assume the sensor is stationary or that a pre-built static environment map is available. A complete solution to this task would be useful in robotics, computer vision, and graphics. Much work has been put into object detection methods using RGBD sensors. While off-the-shelf object detection algorithms (Kart, 2018) are often fast and effective at runtime, training data collection remains a challenge. Using our algorithm for the collection of training data avoids this problem.

2. RELATED WORKS

While there is much previous work on tracking in general, our problem is the estimation of detailed segmentation masks rather than bounding boxes, while the lack of simplifying assumptions restrict the directly-related previous work substantially. Here we only review previous work based on RGBD input data. As compared to generic, short-term tracking on RGB, RGBD tracking is a relatively unexplored area. This can be partly attributed to the lack of datasets with ground truth until recently. (Song and Xiao, 2013) captured and annotated a dataset consisting of 100 videos with an online evaluation system and their benchmark is still the largest available. They also provided multiple baseline algorithms under two main categories: depth as an additional cue and point cloud tracking. Among the ten proposed variations the one with RGBD histogram of oriented gradients (HOG) features and boosted by optical and occlusion detector achieved the best performance. (Meshgi et al., 2016) proposed an occlusion-aware particle filter based tracker that can handle persistent occlusions in a probabilistic manner. Starting with the seminal work of (Bolme et al., 2010), Discriminative Correlation Filter (DCF) based trackers have gained momentum due to their performance, fast model update (training) and mathematical elegance. Success of the DCF approach naturally caught the attention in the RGBD community as well. (Camplani et al., 2015) proposed Discriminative Correlation Filter based RGBD tracker. All these works addresses the online, model-free tracking task, where the goal is to track arbitrary objects in a bounding box rather than provide a segmentation mask (Kart, 2018). Rigid object tracking using depth information, such as the open source method in PCL (Rusu and Cousins, 2011), addresses a similar but simpler problem, as it is not designed to work with deformable objects.

The most similar, recent work to ours is that of (Teichman et al., 2013), which uses online learning to segment consecutive RGBD frames. However, this work assumes a single initial segmentation is provided by human labeling. Older previous similar work on this problem include HoughTrack (Godec et al., 2011), which uses Hough Forests and GrabCut to segment various challenging 2D image sequences. The goal of our algorithm is to collect training examples of objects in unstructured environments from RGBD sequences. Our method is different from (Teichman et al., 2013) in multiple, fundamental aspects. First of all, the authors introduce the use of a large number of features instead of only geometric cues computed to avoid redundant computational complexity. Secondly, their online learning scheme consists of a simple initial segmentation which requires a human in the loop whereas we propose to integrate camera pose estimation with background subtraction to determine initial segmentation automatically.

*Corresponding author

3. APPROACH

In this section, the algorithm for segmenting and tracking non-rigid objects is described. In our approach, since no prior object model and no stationary sensor are assumed, we do foreground seed detection by integration of global motion estimation with occlusion detection first, and then use the seeds and scene flow information to construct a novel matrix that is able to guide the segmentation and tracking procedure for further analysis. We describe these steps in the following sections.

3.1 Foreground Seed Detection

The input to this stage is in the form of 3D point cloud sequences captured by a comsumer depth camera and the desired output is a set of seeds in the first frame. A seed is a connected component, above a minimum size, whose position has changed over time. We do not assume the sensor to be fixed and the objects to be stationary. In other words sensor and objects can move independently. The first step in moving object detection is a comparison of two frames. We begin by downsampling the input point clouds using an octree with leaf size d_r. At most one point per leaf is retained and all subsequent steps are applied on the downsampled point clouds denoted by $D = \{D_t\}$ where $D_t = \{p_i^t\}$. Where t is the frame index and i is the point index. Typically, $2r$ is used as the leaf size, where r is the resolution of the point cloud defined as the average distance between neighboring points. However, directly comparing frames will fail when the sensor moves during data acquisition. Moreover, noise from the sensor and occlusion lead to false seeds. Throughout the paper we do not distinguish between data captured by a stationary or mobile sensor. In other words, we always estimate sensor motion. Figure 1 shows a visualization of input point clouds that are passed to the following stages of processing.

3.1.1 Global Motion Estimation To directly compare two frames, the point cloud of each frame must be represented in a common coordinate system for the sequences from moving sensor. The transformation $T^{(t+1)_t}$ between two successive frames D_t and D_{t+1} has to be estimated.

After downsampling, un-oriented normals are estimated for the point clouds using the technique of (Mitra et al., 2004). Then, Fast Point Feature Histogram (FPFH) descriptors are computed according to (Rusu and Cousins, 2011). The local descriptors are used to establish correspondences, which may contain several errors initially. Errors are reduced by requiring that the correspondences be *reciprocal*, i.e. if the most similar point to p_i^t is p_j^{t+1}, we require that the most similar point to p_j^{t+1} is p_i^t. Otherwise, the correspondence is rejected. We, then, use RANSAC to estimate a rigid transformation that is supported by the largest number of points. For more details on both *rigid transformation estimation* and *FPFH* see our previous research (Guo et al., 2016). Figure 2 shows a visualization of camera pose estimation of two frames that is used to eliminate the false seeds caused by global motion in Section 3.1.2.

3.1.2 Seed Detection and Refinement In this section, we find the potential seeds of objects that have moved between the reference frame D_t and a subsequent frame. In order to identify all points in the moved object, we first transform $D_{t+\delta}$ named target frame onto the coordinate system of the reference frame D_t using $T_{(t+\delta)t}$. Thus, we obtain $D_{t+\delta}^T$.

$$D_{t+\delta}^T = (\prod_{i=t}^{i=t+\delta-1} T_{(i+1)i}) \cdot D_{t+\delta}$$

Where $\delta = \{1,2,3,\cdots\}$ is the number of frames we have to skip to ensure existence of seed when objects are not moving. For the first reference frame, we simply increase δ from one until we find the seed or finish the point cloud sequence. For the rest of the reference frames, we begin with the target frame of the last reference frame. In the registration result of two frames with a moving sensor, parts of the point cloud D_t could be outside of field of view of frame $D_{t+\delta}^T$. Thus, this outer region is recognized as changed space during frame comparison, leading to false seed. In order to filter out this outlier region, we use the view frustum of the sensor. In our paper, the Xtion Pro Live sensor has a vertical field of view of 45°, and a horizontal field of view of 58°. Let C_{inside} denote the points from frame D_t inside of field of view of frame $D_{t+\delta}$. Frustum culling method was employed to filter points outside the field of view of frame $D_{t+\delta}^T$ to obtain C_{inside}. Figure 3 shows a visualization of C_{inside} of frame D_t. To compute efficiently, the comparison is performed using a kd-tree from PCL (Rusu and Cousins, 2011). We begin by building a *kd-tree* over the points of frame $D_{t+\delta}$. For each point that belongs to C_{inside}, a point p_k is added to the set $C_{changed}$ under the condition that it is sufficiently far from a point p_j in $D_{t+\delta}^T$. The condition is implemented by testing whether $d_e(p_i - p_j) \geq 2 \cdot r$.

$$C_{changed} = \{p_k | d_e(p_i - p_j) \geq 2 \cdot r, p_i \in C_{inside}, p_j \in D_{t+\delta}^T\}$$

Where $d_e(p_i, p_j) = \|p_i - p_j\|$ denotes the Euclidean distance between point p_i and p_j , $C_{changed}$ represents the potential candidate moving object. However, as shown in the left part of Figure 4, there are point sets belonging to the static scene in the moving object candidates $C_{changed}$, even though the registration is accurate. This is due to occlusion, that is, these points are visible from view point of frame D_t, but are occluded by objects in frame $D_{t+\delta}$ after objects have moved. In addition, sensor noise and some registration errors generate outliers leading to false seeds. In order to remove the false seed from $C_{changed}$, for each point in $C_{changed}$, we first attempt to model the ray from the viewpoint of frame $D_{t+\delta}$ to this point. If occlusion is detected by finding the nearest point on frame $D_{t+\delta}$ along each ray, then this point is added to point set $C_{occluded}$. Otherwise, this point is added to point set C_{free}. We then cluster points of C_{free} that are moving candidates without occlusion effect using region growing to obtain a set of clusters.

$$\Gamma_t = (\gamma_1, \cdots, \gamma_\Phi)$$

Each of these clusters is grown from a point p_i^t, an unclustered point, in C_{free}. A new point p_j^t is added to the cluster if it is the single nearest neighbor of a point p_m^t which is already in the cluster. To avoid false seeds owing to sensor noise and some registration errors, we drop clusters with less than a minimum number of points. The right part of Figure 4 shows a visualization of the seeds that are passed to the next stages of processing.

3.2 Clustering and Initial Correspondences

There are a large number of possible cues that could inform a segmentation and tracking algorithm. 3D structure, scene flow, depth discontinuities, etc., all provide potentially useful information. Therefore, for the reference frame D_t and target frame $D_{t+\delta}^T$ from the previous stage, we first remove the ground and perform a 3D clustering of remaining points to estimate the 3D scene structure of the frame. Then dense correspondences between D_t and $D_{t+\delta}^T$ are computed as the solution of scene flow. Finally, information of 3D scene structure of the frame and scene flow is encoded with a novel matrix form that is able to guide the segmentation and tracking procedure. We now describe each stage

Figure 1. Left to right: frame D_t from moving sensor; frame $D_{t+\delta}$ from moving sensor; frame D_t from stationary sensor; frame $D_{t+\delta}$ from stationary sensor. Note that all point clouds are raw data which have not been downsampled yet.

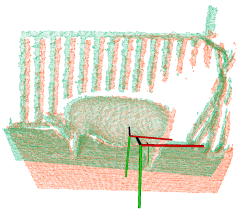


Figure 2. Rigid transformation estimation for the data from moving sensor of Fig. 1

of this section in more detail.

3.2.1 Segmentation Both indoor and outdoor scenes are usually arranged with respect to the gravity direction which is typically orthogonal to the ground. Without loss of generality, we make the assumption that objects move on a ground plane, and our algorithm estimates and removes the ground plane from the D_t. We compute ground plane coefficients using RANSAC and remove all the inliers denoted by C_p^t within a threshold distance. Inliers here are points with distance from the ground plane less than $2r$. Typical values for the parameters of RANSAC are: 10,000 iterations and $\varepsilon = 3r$, where r is the resolution (average distance between neighboring points) of the original point cloud before downsampling. The same process is repeated in frame $D_{t+\delta}^T$ to detect the ground plane cluster $C_p^{t+\delta}$. Once this operation has been performed, the different structures are no longer connected through the ground, so they could be clustered by labeling neighboring 3D points on the basis of their Euclidean distance. Objects that are in contact can be handled as long as they appear separated in some of the frames. See Section 3.3. We then cluster points of $D_t - C_p^t$ using region growing to obtain a set of clusters.

$$C^t = \{C_1^t, \cdots, C_{M_1}^t\}$$

Each of these clusters is grown from a point p_i^t, an unclustered point, in $D_t - C_p^t$. A new point p_j^t is added to the cluster if it is the single nearest neighbor of a point p_m^t which is already in the cluster. In order to avoid the clusters formed due to clutter, clusters with less than a minimum number of points (500) are discarded. The same procedure is applied on the point clouds $D_{t+\delta}^T - C_p^{t+\delta}$ to obtain clusters $C^{t+\delta} = \{C_1^{t+\delta}, \cdots, C_{M_2}^{t+\delta}\}$. Figure 5 shows a visualization of clusters C^t and $C^{t+\delta}$ that are passed to the next stages of processing.

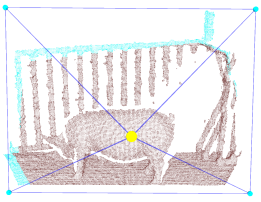


Figure 3. Frustum culling for the data of Fig. 2. Note that C_{inside} is darkgray color. The cyan colored points are filtered by frustum culling with the pose estimated in Section 3.1.1.

Figure 4. Left: Potential candidate of moving object $C_{changed}$ obtained from comparing two frames from the stationary sensor of Fig. 1. Right: Seeds after removing occlusion and sensor noise effects.

3.2.2 Correspondence and *Spatial Temporal Matrix* The output to this stage is one matrix named Spatial Temporal Matrix which integrates information of spatial connection relationship of geometric structure of the scene with temporal information of correspondences. This matrix will be used to guide the tracking part in Section 3.3.

Based on the FPFH descriptor from Section 3.1.1, we begin by finding the nearest and most similar point in frame $D_{t+\delta}^T$ for every point on frame D_t using the Euclidean distance in the 33-D descriptor space $d_f(p_i^t, p_j^{t+\delta}) = ||f(p_i^t) - f(p_j^{t+\delta})||$. Here $f(p_i)$ denotes the FPFH descriptor of point p_i. The resulting set of correspondences is denoted by $K = \{(p_k^t, p_k^{t+\delta})\}$ with $p_k^t \in D_t$ and $p_k^{t+\delta} \in D_{t+\delta}^T$. For query point p_i^t, a new correspondence $(p_i^t, p_i^{t+\delta})$ is added to the set of correspondences K under two conditions: if point $p_i^{t+\delta}$ is the single nearest neighbor of query point; and if the feature distance $d_f(p_i^t, p_j^{t+\delta})$ is less than a threshold (400). Correspondences are sought between frame D_t and frame $D_{t+\delta}^T$ in both directions, frame D_t to frame $D_{t+\delta}^T$ and frame $D_{t+\delta}^T$ to frame D_t. Reciprocal correspondences that have been selected in both directions are retained, while all other correspondences are discarded. Figure 6 shows correspondences found in this stage.

We define the Spatial Temporal Matrix A based on intuition that a correspondence implies that the label of the query point at time

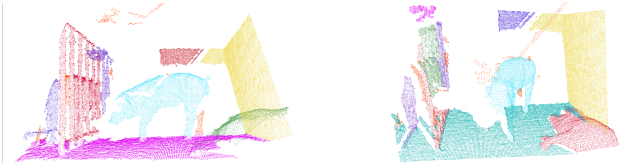


Figure 5. Clustering results of two frames from the stationary sensor of Fig. 1.

Figure 6. Correspondences found for the data from the stationary sensor of Fig. 1

t is likely to propagate to that of the matching point at time $t+\delta$. The matrix A consists of M_1+1 rows and M_2+1 columns. Here M_1 is the number of clusters of frame D_t excluding the ground plane cluster extracted in Section 3.2.1. So is the case with M_2. Given set of correspondences $(p_k^t, p_k^{t+\delta})$ with $p_k^t \in C_i^t$ and $p_k^{t+\delta} \in C_j^{t+\delta}$. Each correspondence between points of two clusters provides evidence that the clusters themselves may be in correspondence. We can statistically infer that cluster $C_j^{t+\delta}$ is likely be the same physical object of cluster C_i^t. Thus, we can treat each correspondence as a vote for this kind of corresponding relationship. Therefore, elements a_{ij} of matrix A are set equal to the number of votes linking cluster C_i^t and $C_j^{t+\delta}$. Then we can conclude that cluster $C_j^{t+\delta}$ matches cluster C_i^t only if a_{ij} almost equals to the number of correspondences between those two clusters. Figure 7 shows the A matrix for the data from the stationary sensor of Figure 1. More discussions will be provided in Section 3.3.

A	8605	0	0	0	0	0	0	0	0	5
B	0	1437	0	0	0	0	0	0	0	13
C	0	0	0	1927	1205	582	0	0	0	0
D	0	0	2754	0	0	0	0	0	0	1
E	337	0	0	0	0	0	1035	0	0	0
F	0	0	0	0	0	538	0	0	0	0
G	0	0	0	0	0	0	0	823	0	3
L	0	0	0	0	0	0	0	0	467	0
M	81	49	54	0	10	0	0	2	0	10946
	A	B	C	D	E	F	G	L	M	N

Figure 7. Matrix A for the data from stationary sensor of Fig. 1

3.3 Multi-body Tracking

In this section we describe the novel part of our approach, which jointly performs segmentation as well as tracking of multiple non-rigid objects. Towards this goal, we propose criteria based on matrix A from last section that is able to guide the tracking and segmentation process. In the remainder of this section, we start by deducing the criteria from the distribution of correspondences in every cluster and showing how to track the object using it. We then introduce our segmentation procedures which operate only when objects are touching each other after moving.

3.3.1 Criteria and Tracking From Section 3.2.2, we learn that the track of cluster C_i^t in frame $D_{t+\delta}$ can be found by using the $i-$th row of matrix A. Base on the observation that each isolated moving object is very likely corresponding to one cluster. Thus we can track the objects by using the tracking relationships discovered with matrix A. The situations that multiple objects are in contact with each other after moving will be solved in Section 3.3.2. An oversimplification of the problem would be to consider all moving objects in 3d scene. Specifically, we only care about the rows of matrix corresponding to moving objects. Therefore we recognize the clusters which contain moving objects in frame D_t using seeds Γ_t obtained in Section 3.1.2. Let $C_{candidate}$ denote the set of clusters which contain moving objects. The set representation of $C_{candidate}$ is shown as follows.

$$C_{candidate} = \{C_i^t | C_i^t \in C_t, \gamma_i \in \Gamma_t, C_i^t \cap \gamma_i \neq \varnothing\}$$

For each seed γ_i in Γ_t, we choose any point p_k^t in γ_i. A new cluster C_i^t from C_t is added to the set under the conditions that it contains point p_k^t and that it does not belong to the set. Note that because the seeds and clusters are compact, we can obtain unique cluster by using any point in one seed. Here we use first by index in every seed.

The normalized value regardless of the number of correspondences in one cluster is useful to determine a common threshold value to cut off the wrong corresponding relationship between two clusters which is caused by a few of wrong correspondences obtained in Section 3.2.2. For that reason, each entry of matrix A is divided by the maximum entry in its column, the resulting matrix denoted by A^c. However, small moving objects are likely to be ignored when they move close to very large objects. Therefore the same procedure is applied on the matrix rows, the resulting matrix denoted by A^r. Figure 8 shows A^c for matrix A from Figure 7. The larger the entry of matrix A^c or A^r is, the more it indicates that two clusters are corresponding. For every cluster C_i^t in $C_{candidate}$, given its corresponding $i-$th row of matrix A^r denoted by row vector A_{i*}^r, we add j to set $R_i = \{r_k^i\}$ if A_{ij}^r is greater than a threshold ω (0.15). Thus we record the corresponding clusters in target frame $D_{t+\delta}$ by using indexes of the columns of the matrix for cluster C_i^t. The same applies to clusters in reference frame D_t by using indexes of the rows of the matrix. We can conclude following criteria:

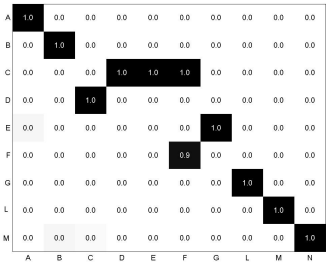


Figure 8. Matrix A^c for matrix A from Figure 7

1. We infer that cluster $C_{r_k^i}^{t+\delta}$ is the track of cluster C_i^t from the conditions: if the cardinality of set $|R_i| = 1$; given the r_k^i-th column of matrix A^c denoted by column vector $A_{*r_k^i}^c$, we add h to set $O_{r_k^i} = \{o_d^{r_k^i}\}$ if $A_{hr_k^i}^c$ is greater than a threshold ω (0.15) with $d = \{1, 2, \cdots\}$, and if the cardinality of set $|O_{r_k^i}| = 1$. Obviously here we have $\max(k) = 1, \max(d) = 1$. Note that this is the isolated object tracking case in physical world. It can be multiple objects tracking as long as they are not contacted with each other when multiple seeds are detected, that is to say multiple rows with different i get involved.

2. The same conditions from criterion 1 except the cardinality of set $\max(d) = |O_{r_k^i}| > 1$, we infer that cluster $C_{r_k^i}^{t+\delta}$ is the track of $\max(d)$ clusters $C_{o_d^{r_k^i}}^t$. Thus cluster $C_{r_k^i}^{t+\delta}$ can be split into $\max(d)$ small regions. The implementation of segmentation will be presented in Section 3.3.2. This case might be the result of one of the following situations: multiple objects

moving close to each other, objects moving close to background such as a wall or table, one moving object occluded by object in the middle at time t. Clusters $C^t_{o_d^{r_k^i}}$ need to be merged into one cluster if the situation belongs to the third case.

3. The same conditions from criterion 1 except the cardinality of set $\max(k) = |R_i| > 1$, we infer that $\max(k)$ clusters $C^{t+\delta}_{r_k^i}$ are the track of cluster C^t_i. This case might be the result of one of the following situations: multiple objects moving away from each other, object moving away from background, one moving object exposed completely after occlusion in the middle. Clusters $C^{t+\delta}_{r_k^i}$ need to be merged into one cluster if situation belongs to the third case. This criterion is similar to criterion 2 if the reference frame and target frame switch position at the beginning of Section 3.1.2.

4. We can determine whether object is leaving the view of camera. More specifically, we infer that one object is just moving out of view of camera if the cardinality of set $\max(k) = |R_i| = 0$. Similarly, we can detect an entrance of one object if the reference frame and target frame switch positions at the beginning of Section 3.1.2.

Based on the above criteria, we can handle most of the issues in tracking of multiple object. Tracking multiple isolated objects is straightforward by applying criterion 1. Specifically, criterion 1 is applied on each cluster C^t_i in $C_{candidate}$ to obtain the cluster of its track in time $t+\delta$. Based on Section 3.2.1, we know that isolated objects correspond to clusters. Thus, we implement the multiple $T^{(t+1)_t}$ objects tracking. Figure 9 shows the tracking result for the data from the stationary sensor of Figure 1. We have to apply criterion 2 and 3 when objects move together or other cases which lead to changes of spatial connectivity of compact cluster. However, note that our algorithm is only using the reference frame and target frame and as a result we can not determine when to split or merge since we do not know whether the cluster is one object or not. We may solve this problem by using information more than two frames in the future. Since we know the cluster corresponding relationship, merging is pretty simple. In contrary to merging, splitting is more difficult. Therefore, splitting (segmentation) based on criterion 2 will be presented in following section.

Figure 9. Isolated object tracking result for the data from stationary sensor of Fig. 1

3.3.2 Splitting From time t to $t+\delta$, more objects could be merged into the same cluster because they are too close or they contact with each other after moving or, for the same reason, one object could be clustered together with the background, such as a wall or a table. Based on criterion 2 in last section, we know the clusters $C^t_{o_d^{r_k^i}}$ at time t and its corresponding merging cluster $C^{t+\delta}_{r_k^i}$ at time $t+\delta$. The aim of this subsection is to split cluster $C^{t+\delta}_{r_k^i}$ into $\max(d) = |O_{r_k^i}|$ clusters which are the track of clusters $C^t_{o_d^{r_k^i}}$. In Figure 10 we report an example adhering to criterion 2. In particular, it appears that one object makes contact with a wall after moving with $\max(d) = 2$ and $\max(k) = 1$.

Given C^t_i in $C_{candidate}$ with $\max(d) > 1$ and $\max(k) = 1$, for every cluster $C^t_{o_d^{r_k^i}}$ we start by aligning it with cluster $C^{t+\delta}_{r_k^i}$ in order to obtain the dense correspondences between them. These correspondences are employed to split cluster $C^{t+\delta}_{r_k^i}$. More specifically, we estimate the rigid transformation between cluster $C^t_{o_d^{r_k^i}}$ and $C^{t+\delta}_{r_k^i}$ by using the method mentioned in Section 3.1.1. Meanwhile, we align those two clusters. Then we obtain the correspondences by using the same strategy described in Section 3.2.2. The resulting set of correspondences is denoted by $K^{kernel}_{kd} = \{(s_m, t_m)\}$ with $s_m \in C^t_{o_d^{r_k^i}}$ and $t_m \in C^{t+\delta}_{r_k^i}$. This gives us a set of reliable correspondences that we refer to as *kernel correspondences*. These correspondences can be seen at the left part of Figure 11. Note that those correspondences are not dense enough to create a segmentation mask for rigid object or background, much less to make a segmentation mask for non rigid object.

To overcome this problem, kernel correspondences are propagated to all unmatched points from their neighbors in order to obtain dense correspondences which can be used for segmentation. In

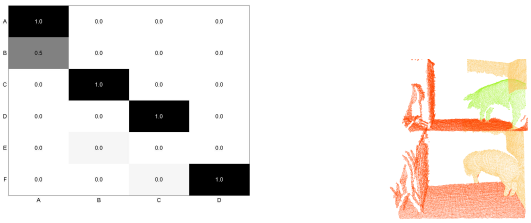


Figure 10. An example adhering to criterion 2. Left: matrix A^c for the data in the right. Right top: clusters $C^t_{o_1^{r_k^i}}$ and $C^t_{o_2^{r_k^i}}$ at time t, $o_1^{r_k^i}$ and $o_2^{r_k^i}$ record the indexes of row A,B in matrix A^c. Right: corresponding merging cluster $C^{t+\delta}_{r_1^i}$ at time $t+\delta$.

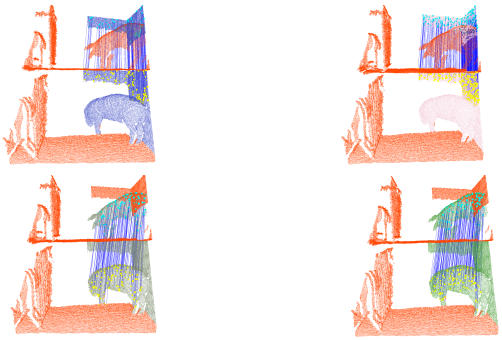


Figure 11. Left: Kernel correspondences for clusters $C^t_{o_1^{r_k^i}}$ and $C^t_{o_2^{r_k^i}}$ from Figure 10. Right: 50% dense correspondences for clusters $C^t_{o_1^{r_k^i}}$ and $C^t_{o_2^{r_k^i}}$ from Figure 10.

particular, given a cluster $C^t_{o^{r^i_k}_d}$, its corresponding cluster $C^{t+\delta}_{r^i_k}$, and a sample point $s_j \in C^t_{o^{r^i_k}_d}$ that does not have a corresponding point yet, we seek nearest point s_v with a correspondence (s_v, t_v). Let r denote geodesic distance between s_j and s_v. Note that we seek neighbors in the same point cloud here, we use geodesic distances in this search, approximated by graph distances in the 20-nearest neighbor graph. We then assign to s_j a corresponding point that is most consistent with the kernel correspondences according to minimum geodesic distance.

$$K^{dence}_{kd} = \{(s_j,t_j) | t_j = \underset{t \in G_{2r}(t_v, C^{t+\delta}_{r^i_k})}{\arg\min} e_{K^{kernel}_{kd}}(s_j,t),\ s_j \in C^t_{o^{r^i_k}_d}\}$$

where $G_{2r}(t_v, C^{t+\delta}_{r^i_k})$ is the geodesic neighborhood of radius $2r$ in cluster $C^{t+\delta}_{r^i_k}$), the consistency error $e_{K^{kernel}_{kd}}$ is defined as:

$$e_{K^{kernel}_{kd}}(s,t) = \sum_{(s_m,t_m) \in K_{kd}} \left[d_g(s,s_m) - d_g(t,t_m) \right]^2$$

where $d_g()$ denotes the geodesic distance between two points. We thus obtain the final set of correspondences K_{kd} comprising the kernel K^{kernel}_{kd} and the correspondences from the propagation K^{dense}_{kd}. See the right part of Figure 11 for a visualization of a sampling of the dense correspondences.

In order to segment all points in $C^{t+\delta}_{r^i_k}$ that are the track of cluster $C^t_{o^{r^i_k}_d}$, dilation is applied to all points t_m in K_{kd} on cluster $C^{t+\delta}_{r^i_k}$. Dilation adds geodesic neighborhood of radius $\eta(0.08)$ of point t_m to points set $C^{dilation}_{kd}$ as following shows.

$$C^{dilation}_{kd} = \{p_i | (s_m,t_m) \in K_{kd}, p_i \in C^{t+\delta}_{r^i_k}, d_g(p_i,t_m) \le \eta\}$$

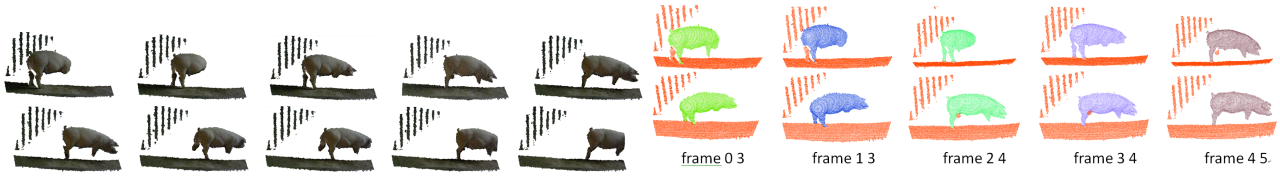
We then cluster points $C^{t+\delta}_{r^i_k} - C^{dilation}_{kd}$ using region growing to obtain a set of clusters. Then, the largest among those clusters, in terms of number of points, is denoted by $C^{largest}_{kd}$. Then we remove all points in $C^{largest}_{kd}$ from $C^{t+\delta}_{r^i_k}$, the remaining points form one compact cluster denoted by C^{track}_{kd}. This cluster is the track of cluster $C^t_{o^{r^i_k}_d}$ that we seek for. Then we assign $C^{largest}_{kd}$ to $C^{t+\delta}_{r^i_k}$ and set $d = d+1$. This procedure is repeatedly applied until the $\max(d)$-th cluster is processed. Figure 12 shows the splitting segmentation result for the data from Figure 11.

4. EXPERIMENTAL RESULTS

We present qualitative and quantitative results on three indoor data sets collected with a Xtion Pro Live RGBD camera. The first data set, named live pigs, has been collected inside of a small pig farm with planar ground, multiple arbitrarily moving pigs with white hair, fences and other static facilities. The second data set, multiple people, has been collected in office with planar ground, two people walking around, other static facilities. The third data set, single live dog, has been collected in office with plane ground, one arbitrarily moving dog with black hair, other static objects. We present qualitative and quantitative results on those three diverse datasets. All datasets are processed with constant parameter values. Specifically, the number of points in one frame is $307,200$, the octree leaf size for downsampling d_r is set equal to $2r$, where $r = 0.005m$ is the resolution of the point cloud, the radius of the neighborhood for SPFH computation is $10r$, k used in computing FPFH is set to 30, the minimum number of points for a seed in Γ_t is 200, the Xtion Pro Live sensor has a vertical field of view of $45°$, an horizontal field of view of $58°$, the minimum number of points for a cluster in C^t is 500, the threshold on normalized value is $\omega = 0.15$, while the threshold on FPFH distance is set to 400 and the geodesic neighborhood of radius is $\eta = 0.08$.

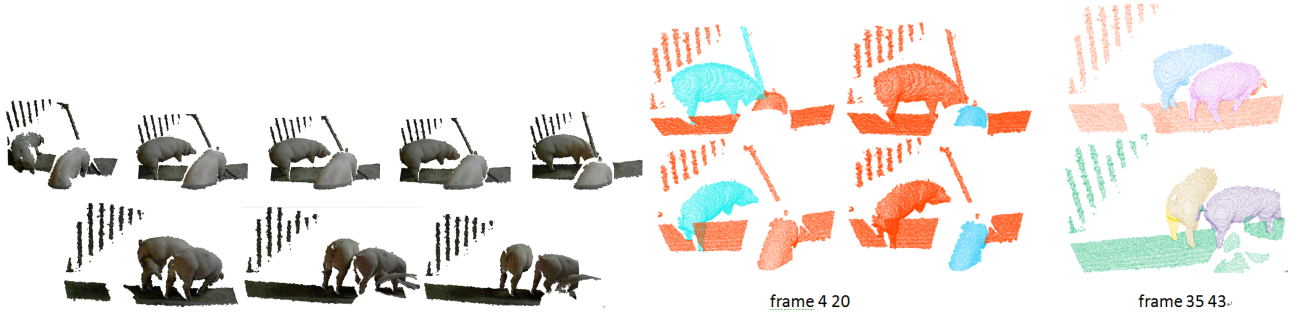
Figure 12. An example of splitting segmentation for data from Figure 11. Left: $d = 1$. Right: $d = 2$.

Results on live pigs. The first dataset we validate our method on is live pigs, consisting of 7 sequences with a total of about 350 frames. Each sequence was chosen for testing different scenarios. Our results demonstrate that our method can be effective even in sequences which include significant non-rigid object transformations and a lack of visually distinguishing appearance. As seen in the illustration in Figure 13, a single pig can be tracked efficiently. In addition to tracking multiple isolated pigs, the method works for multiple instances making contact for a short time. See Figure 14 for examples. In particular, the interface between two contacted pigs in sequence 2 is maintained correctly. Note that the stick appears in the tracking results between frames 4 and 20 because we ignore all small clusters with less than 500 points when constructing the matrix A. As Figure 14 shows, it is worth to notice that yellow part was partitioned off accurately from the track in the results of frame 35 and 43, was owing to occlusion. As is typical in tracking tasks, stability versus permissiveness is a tradeoff. Since we detect the seeds by using only space changes between two frames. In multiple tracking cases, it is not well defined whether static objects relative to moving objects should be seeds or not; this is a common cause of errors in the current implementation. An example of this occurs in sequence 3 and can be seen in Figure 15, when one of the pigs stands still the others moved around, and the system can not determine that motionless pigs should be assigned to foreground. The error is then propagated. Similarly, one pig makes contact with static objects over time leading to the pig and objects to be tracked together. Fortunately, it is likely that we can resolve these limitations by knowing which objects belong to foreground before processing the current frame, through integrating tracking results of previous frames. The rest of two sequences from the first dataset was taken by a hand-held sensor. The tests on those two sequences are similar to the other sequences. Except that global motion effects appeared. It is worth to notice that our global motion estimation can be effective in noisy sequences. We present quantitative results for the first dataset in Table 1. Individual frames are evaluated with two very intuitive metrics. In particular, the ratio of misses and false positives in the sequence denoted as FN and FP respectively, computed over the total number of objects presented in all frames. Ground truth was generated by counting manually. Specifically, we count all moving objects for which no hypothesis was output as misses such as the motionless pigs in multiple pigs case, and also all tracker hypothesis for which no real object exists as false positives such as objects that were in contact with pigs for some time.



(a) Input of sequence 4 from the live pigs dataset. (b) Output of sequence 4 from the live pigs dataset.

Figure 13. Single pig tracking for sequences 4 from the first dataset.



(a) Input of sequence 2 from the live pigs dataset. (b) Output of sequence 2 from the live pigs dataset.

Figure 14. Multiple pig tracking and split segmentation for sequence 2 from the first dataset.

Datasets	FN	FP
live pigs	30.65%	4.36%

Table 1. Miss rate and false positive rate results for the live pigs dataset

Results on multiple people. Here we present results on the multiple people dataset, consisting of 2 sequences with a total of about 150 frames with different conditions such as two people moving separately, two people making contact with each other, people contacted with facilities. Figure 16 shows results for the case of two people making contact with each other, while Table 2 shows quantitative results on the multiple people dataset.

Datasets	FN	FP
multiple people	25.66%	20.24%

Table 2. Miss rate and false positive rate results for the multiple people dataset.

Results on single live dog. The final dataset collected for this paper is one sequence with one random walking dog above a plane ground without other facilities around. Table 3 shows quantitative results on the single live dog dataset.

Datasets	FN	FP
single live dog	5.74%	13.19%

Table 3. Miss rate and false positive rate results for the single live dog dataset.

Timing Results. Running times vary depending on the way of setting up the camera, the size of moving objects and also segmentation time. Each segmentation requires the computation of geodesic distance for two corresponding clusters to find the dense correspondences which are used for segmentation. Timing results are reported on an Intel Core i7-2670QM CPU at 2.20 GHZ. The Point Cloud Library (PCL) is used for many of the supporting tasks, like FPFH descriptor computation, but other parts of the code are not optimized.

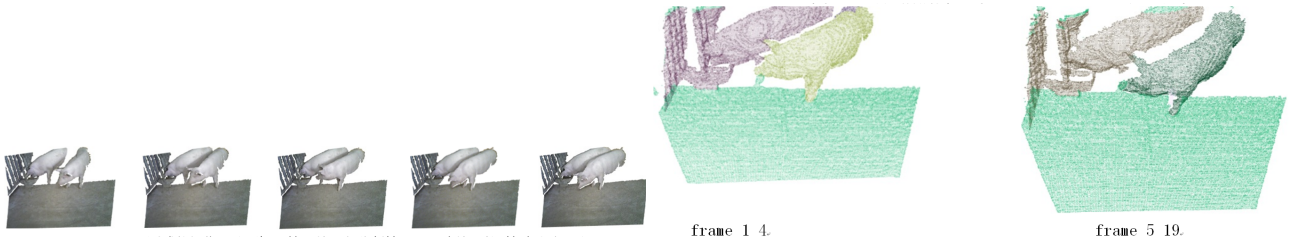
Execution times per task for each case are shown in Table 4. The computation of normals and FPFH descriptors depends on the local density of the point cloud since they are done in r-neighborhoods. Correspondence computation and matrix construction primarily depend on the number of input points, while segmentation depend on the number of parts to be split and the number of points in each part.

Case	Init.	Seg.	Corr.	Track/Split	Total
Tracking	13.95	0.91	56.41	0.01	71.28
Split	13.95	0.91	56.41	935.6	1006.87

Table 4. Timing results each step for each frame in seconds.The four main stages of processing are: FPFH computation and initial alignment; seed detection and intial segmenation; correspondences finding and matrix construction; tracking or segmenation.

5. CONCLUSIONS

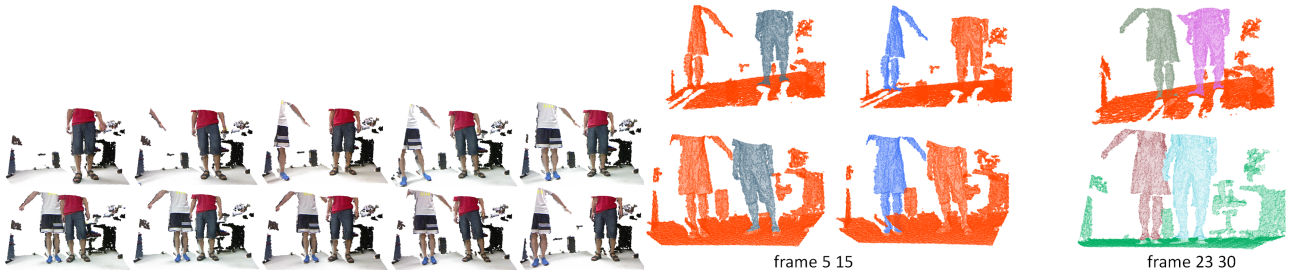
We described a novel method of segmenting and tracking deformable objects in RGBD, making minimal assumptions about the input data. The approach uses the Spatial Temporal Matrix constructed with geometric and motion features to assist tracking and segmentation procedure, and clearly improves the segmentation results when mutiple non-rigid objects are in contact. We have demonstrated those techniques on several RGBD sequences and have shown that good 3D non-rigid segmentation and tracking can be achieved. However, real-time performance is not feasible now. One possible solution is introducing GPU computing and parallel programming model into our method. While there



(a) Input of sequence 5 from the live pigs dataset.

(b) Output of sequence 5 from the live pigs dataset.

Figure 15. Multiple pig tracking for sequences 5 from the first dataset.



(a) Input of sequence 1 from the multiple people dataset.

(b) Output of sequence 1 from dataset of the multiple people dataset.

Figure 16. Multiple people tracking and split segmentation for sequence 2 from the second dataset.

remains more work to be done before a completely robust solution is available for unassisted segmentation and tracking. Our algorithm is suitable for collecting training examples of objects in unstructured environments. A complete solution to this task would have far-reaching ramifications in robotics, computer vision and model capture.

ACKNOWLEDGEMENTS

We thank Shangdong Weihai swine-breeding center of DA BEI NONG GROUP for providing us materials for experiment. This work was supported by National Natural Science Foundation of China [Grant No. 41601491].

REFERENCES

Bolme, D. S., Beveridge, J. R., Draper, B. A. and Lui, Y. M., 2010. Visual object tracking using adaptive correlation filters. In: *Computer Vision and Pattern Recognition*, pp. 2544–2550.

Camplani, M., Hannuna, S., Mirmehdi, M., Damen, D., Paiement, A., Tao, L. and Burghardt, T., 2015. Real-time rgb-d tracking with depth scaling kernelised correlation filters and occlusion handling. In: *Proceedings of the British Machine Vision Conference (BMVC)*, BMVA Press, pp. 145.1–145.11.

Choi, C. and Christensen, H. I., 2016. Rgb-d object pose estimation in unstructured environments. *Robotics and Autonomous Systems* 75, pp. 595 – 613.

Godec, M., Roth, P. M. and Bischof, H., 2011. Hough-based tracking of non-rigid objects. In: *International Conference on Computer Vision*, pp. 81–88.

Guo, H., Zhu, D. and Mordohai, P., 2016. Correspondence estimation for non-rigid point clouds with automatic part discovery. *Vis. Comput.* 32(12), pp. 1511–1524.

Kart, U., 2018. How to make an rgbd tracker ? In: *ECCV VOT Workshop*.

Matthews, S. G., Miller, A. L., Pltz, T. and Kyriazakis, I., 2017. Automated tracking to measure behavioural changes in pigs for health and welfare monitoring. *Scientific Reports* 7(1), pp. 17582.

Meshgi, K., Maeda, S., Oba, S., Skibbe, H., Li, Y. and Ishii, S., 2016. An occlusion-aware particle filter tracker to handle complex and persistent occlusions. *Computer Vision and Image Understanding* 150, pp. 81 – 94.

Mitra, N. J., Nguyen, A. and Guibas, L., 2004. Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry and Applications* 14(4–5), pp. 261–276.

Rusu, R. B. and Cousins, S., 2011. 3d is here: Point cloud library (pcl). In: *International Conference on Robotics and Automation*, pp. 1–4.

Song, S. and Xiao, J., 2013. Tracking revisited using rgbd camera: Unified benchmark and baselines. In: *International Conference on Computer Vision*, ICCV '13, IEEE Computer Society, Washington, DC, USA, pp. 233–240.

Teichman, A., Lussier, J. T. and Thrun, S., 2013. Learning to segment and track in rgbd. *IEEE Transactions on Automation Science and Engineering* 10(4), pp. 841–852.

Xue, H., Liu, Y., Cai, D. and He, X., 2016. Tracking people in rgbd videos using deep learning and motion clues. *Neurocomputing* 204, pp. 70 – 76. Big Learning in Social Media Analytics.

Zollhöfer, M., Stotko, P., Görlitz, A., Theobalt, C., Nießner, M., Klein, R. and Kolb, A., 2018. State of the Art on 3D Reconstruction with RGB-D Cameras. *Computer Graphics Forum (Eurographics State of the Art Reports 2018)*.