Real-time and Model-free Object Tracking using Particle Filter with Joint Color-Spatial Descriptor

Shile Li[1], Seongyong Koo[2] and Dongheui Lee[1]

***Abstract*— This paper presents a novel point-cloud descriptor for robust and real-time tracking of multiple objects without any object knowledge. Following with the framework of incremental model-free multiple object tracking from our previous work [5][7][6], 6 DoF pose of each object is firstly estimated with input point-cloud data which is then segmented according to the estimated objects, and incremental model of each object is updated from the segmented point-clouds. Here, we propose Joint Color-Spatial Descriptor (JCSD) to enhance the robustness of the pose hypothesis evaluation to the point-cloud scene in the particle filtering framework. The method outperforms widely used point-to-point comparison methods, especially in the partially occluded scene, which is frequently happened in the dynamic object manipulation cases. By means of the robust descriptor, we achieved unsupervised multiple object segmentation accuracy higher than 99%. The model-free multiple object tracking was implemented by using a particle filtering with JCSD as a likelihood function. The robust likelihood function is implemented with GPU, thus facilitating real-time tracking of multiple objects.**

I. INTRODUCTION

Object tracking is one of the core functions that enables autonomous robots to perform intelligent applications such as learning from human demonstrations [9], especially for grasping [13][16] and manipulation [8]. Its difficulty relies on two main aspects: availability of target object information, and uncertainty of the measurement data. For example, when a robot faces to a new environment, prior information of tracking targets is not always provided and the objects are easily occluded by obstacles or parts of the robot itself from the view point of the camera mounted on the robot.

These difficulties call for a model-free and robust tracking method, one of which was proposed in [6]. With a widely used and low-cost RGB-D camera, the tracking process has three main steps: *pose tracking*, *point-cloud segmentation*, and *model update*. The *pose tracking* estimates the target object's pose in newly arrived point-cloud data, then *point-cloud segmentation* groups the data according to the estimated objects (as an example, Fig. 1 shows the segmentation results of the four individual objects that are contacted with each other), and the *model update* changes the current object knowledge (*object model*) such that it can be adapted to the new observation based on the estimated pose.

These steps have mutual influences, where each step needs the result from the previous step as an input to proceed. Therefore, it is important to enhance accuracy and robustness in all steps to obtain better tracking performance.

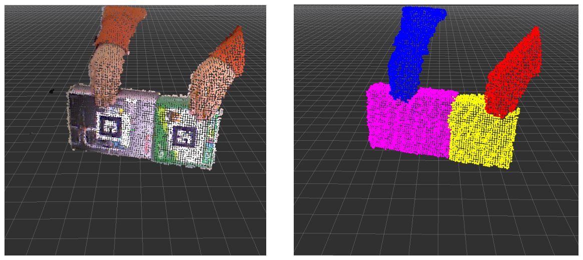
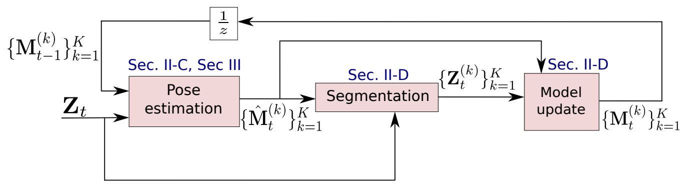

(a) Input point cloud (b) Object segmentation

Fig. 1. An example of multiple object segmentation result in the case of four individual objects that are contacted with each other, segmentation result is indicated in different colors.

This paper aims to devise a novel *pose hypothesis evaluation* method to improve both tracking accuracy and segmentation robustness for real-time model-free tracking of multiple objects. For object pose estimation using recursive Bayesian Filtering or optimization processes, often multiple pose hypotheses are proposed, where the hypotheses should be evaluated to reveal how each hypothesis closes to the given observed data. With the point-cloud data, the hypothesis evaluation is performed by comparing a set of measurement points in the neighborhood of the pose hypothesis (*hypothesis data*) and current knowledge of the object. Designing a robust and efficient hypothesis evaluation method is a challenging task, since the knowledge of object cannot be specified and the sensor data are often noisy and partially missed.

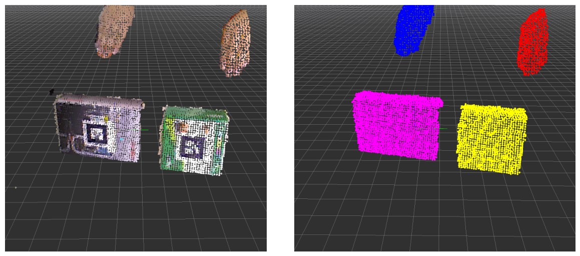
Here, we propose Joint Color-Spatial Descriptor (JCSD) that represents a probability density of a measurement point in the joint color-spatial space. Compared to widely used point-to-point comparison methods [2][14], where likelihood is estimated by summing up each point-pair likelihood (in terms of color coherence, spatial distance etc.), the pose hypothesis evaluation with JCSD outperforms, especially in the partially occluded scene, which is frequently happened in the dynamic object manipulation cases. Moreover, it is computationally efficient enough to rebuild the descriptor at each model update step, thus enabling real-time performance. A new model update scheme is also proposed to smoothly add new segmented point-cloud data into the object model to maintain its adaptivity to the new sensor data and at the same time its robustness against segmentation error. Compared to [13][5][7], which used simplified object model (Gaussian Mixture Model or supervoxel), precise object boundary shape

[1] Chair of Automatic Control Engineering, Department of Electrical Engineering and Computer Engineering, Technische Universität München, 80290 Munich, Germany `li.shile@mytum.de, dhlee@tum.de.` [2] S. Koo is currently with the Autonomous Intelligent Systems (AIS) Group, Computer Science Institute VI, University of Bonn, Germany. Email: koosy@ais.uni-bonn.de, this paper was submitted when he was in Munich.



$\{\mathbf{M}_{t-1}^{(k)}\}_{k=1}^{K}$
$\frac{1}{z}$
Sec. II-C, Sec III
Pose estimation
$\mathbf{Z}_t$
$\{\hat{\mathbf{M}}_t^{(k)}\}_{k=1}^{K}$
Sec. II-D
Segmentation
$\{\mathbf{Z}_t^{(k)}\}_{k=1}^{K}$
Sec. II-D
Model update
$\{\mathbf{M}_t^{(k)}\}_{k=1}^{K}$

Fig. 2. System overview for model-free object tracking. All bold symbols are presented as point cloud.



(a) Input point cloud (b) Euclidean clustering result

Fig. 3. An example of the non-contact case where four individual objects are clearly separated with Euclidean clustering method.

can be retained, thus higher object segmentation accuracy was achieved. The proposed method was implemented with GPU by using a particle filtering [4][3] with JCSD as a likelihood function. As the robust likelihood function allows to reduce the number of particles, a large number of hypotheses evaluations that are required by multiple particle filters can be calculated in real-time for tracking multiple objects at the same time. Moreover, the experiment results showed that the segmentation accuracy was achieved higher than 99%.

This paper is organized as follows. Model-free object tracking system based on particle filter and the new model update scheme is described in section II. The proposed new hypothesis evaluation method is explained in section III. The performance of our approach is shown with several experiments in section IV. Finally, a conclusion is given in section V.

II. MODEL-FREE OBJECT TRACKING SYSTEM

A. System overview

The proposed model-free tracking system consists of three parts as shown in Fig. 2. When a new point-cloud comes, the object's pose needs to be estimated at first (pose estimation). Then, a set of points from current sensor data, which belongs to the object, is extracted based on the estimated pose (segmentation). Finally, the object model is updated with old model data and newly the extracted points (model update).

In this work, at time step t, a frame of sensor data, $\mathbf{Z}_t$, and an object model, $\mathbf{M}_t$, are represented as a set of points, $\{(\mathbf{p}_1,\mathbf{c}_1),...,(\mathbf{p}_n,\mathbf{c}_n)\}$, each of which contains 3D location, $\mathbf{p}_i=(x,y,z)$, and its corresponding color, $\mathbf{c}_i=(r,g,b)$. For the multiple object case, $\{\mathbf{M}_t^{(k)}\}_{k=1}^{K}$ are K object models at time step t. Let's assume that the initial object models $\{\mathbf{M}_{t_0}^{(k)}\}_{k=1}^{K}$ can be obtained from the non-contact case[1] as shown in Fig. 3, where the input point-cloud can be easily segmented into object individuals using an Euclidean clustering method [15]. Here, t_0 is the time step for last frame of non-contact case.

When new frame data $\mathbf{Z}_t$ arrives at time step t, the pose estimation process is firstly applied based on a particle filter with the proposed JCSD. Then, the previous object model $\mathbf{M}_{t-1}^{(k)}$ is transformed from the object coordinate to the world coordinate based on the estimated pose. Next, point-cloud segmentation is performed based on the transformed object models $\hat{\mathbf{M}}_t^{(k)}$, which results in a subset of input point-cloud data $\mathbf{Z}_t^{(k)}$. New object model $\mathbf{M}_t^{(k)}$ is then built by fusing $\hat{\mathbf{M}}_t^{(k)}$ and $\mathbf{Z}_t^{(k)}$. Finally, the new object model $\mathbf{M}_t^{(k)}$ is transformed back to its object coordinate.

[1]The transition from non-contact to contact case is detected using algorithms provided by [7].

B. Object pose representation

All points in the object model $\mathbf{M}$ are described relative to the object coordinate, which is fixed on the centroid of the initial object model $\mathbf{M}_{t_0}$. At time step t, one object's pose state is presented as:

$$\mathbf{X}_t=(x,y,z,roll,pitch,yaw). \tag{1}$$

For clarity, we denote transformed point cloud $\hat{\mathbf{M}}$ from $\mathbf{M}$ based on the pose state $\mathbf{X}$ as:

$$\hat{\mathbf{M}}=\mathbf{X}\otimes\mathbf{M}. \tag{2}$$

C. Particle filter for pose estimation

Particle filter is a robust approach for tracking object pose, it is not limited to a linear system and does not require noise to be Gaussian [19][18]. At time step t, the posterior probability of an object's pose is approximated by a set of hypothesis samples $\{\mathbf{X}_t^{(i)}\}_{i=1}^{N}$ with their corresponding weights $\{\pi_t^{(i)}\}_{i=1}^{N}$. The set of hypothesis $\{\mathbf{X}_t^{(i)}\}_{i=1}^{N}$ is sampled by using Sampling Importance Resampling method. Each sample-weight pair is defined as a particle $S_t^{(i)}=(\mathbf{X}_t^{(i)},\pi_t^{(i)})$, where the weight values are normalized as $\sum_{i=1}^{N}\pi_t^{(i)}=1$. Each weight value is proportional to its corresponding hypothesis likelihood, which is the probability that an object model $\mathbf{M}_{t-1}$ fits to the current measurement $\mathbf{Z}_t$ based on the hypothesis pose state $\mathbf{X}_t^{(i)}$:

$$\pi_t^{(i)}\propto p(\mathbf{Z}_t|\mathbf{X}_t^{(i)}\otimes\mathbf{M}_{t-1}). \tag{3}$$

The likelihood estimation method will be described in Section III.

The final estimation of the current state $\mathbf{X}_t^{*}$ is then the integration of all particle states based on their weights:

$$\mathbf{X}_t^{*}=\sum_{i=1}^{N}\pi_t^{(i)}\cdot\mathbf{X}_t^{(i)} \tag{4}$$

D. Object segmentation and model update

Assuming that K objects are present, object segmentation task is to label each point $\{\mathbf{p}_i\}_{i=1}^{n}$ from $\mathbf{Z}_t$ with an object index o_i, where $o_i\in[1,K]$ and n is the number of points in $\mathbf{Z}_t$ and K is the number of objects. o_i can be determined by calculating Euclidean distance from one point to each object in their current estimated pose. The Euclidean distance between ith point $\mathbf{p}_i$ and kth object $\hat{\mathbf{M}}_t^{(k)} = \mathbf{X}_t^* \otimes \mathbf{M}_{t-1}^{(k)}$ is defined as $d_i^{(k)}$, which is the distance between $\mathbf{p}_i$ and the closest point in $\hat{\mathbf{M}}_t^{(k)}$:

$$d_i^{(k)} = \min_j \|\mathbf{p}_i - \hat{\mathbf{p}}_{t,j}^{(k)}\|, \tag{5}$$

where $\hat{\mathbf{p}}_{t,j}^{(k)}$ is the jth point of $\hat{\mathbf{M}}_t^{(k)}$. This is implemented using k-d tree search provided by Point Cloud Library [15].

The ith point is then labeled to the object, which results in the smallest $d_i^{(k)}$ and if $d_i^{(k)}$ is smaller than a threshold τ:

$$o_i = \begin{cases} \text{argmin}_k(d_i^{(k)}) & \textit{if } min(d_i^{(k)}) \leqq \tau \\ \emptyset & \textit{if } min(d_i^{(k)}) > \tau \end{cases} \tag{6}$$

The threshold τ is introduced here to avoid false association of irrelevant points (points from other objects or background) and is set as twice of the sampling distance between points in the sensor data. The subset of identified input point-cloud that is associated with the kth object is then presented as $\mathbf{Z}_t^{(k)}$.

In the model update step, if the newly segmented point-cloud $\{\mathbf{Z}_t^{(k)}\}_{k=1}^K$ simply replaces the object model $\{\mathbf{M}_t^{(k)}\}_{k=1}^K$, possibly existing segmentation errors will cause inaccurate object model estimation, and the errors are accumulated in the tracking process due to the feedback loop. To overcome this issue, we propose the following model update scheme to merge old object model and new segmented data. At time step t, we select $\alpha\%$ ($\alpha \in [0,100]$) points randomly from new segmented data $\{\mathbf{Z}_t^{(k)}\}_{k=1}^K$ and $(100-\alpha)\%$ points from $\hat{\mathbf{M}}_t^{(k)}$ to build the new object model $\mathbf{M}_t^{(k)}$. Intuitively, if the object appearance changes rapidly during tracking, a larger update ratio is required to adapt the model to recent observations, otherwise a smaller update ratio is preferred to maintain the stability of object model. Fig. 4 shows the examples of the object model with $\alpha = 2\%$, which is empirically obtained value for balancing the trade-off between stability and adaptability of the model update considering the object appearance changing speed in our tracking scenarios.

III. HYPOTHESIS EVALUATION

Hypothesis evaluation is a core part of the particle filtering to estimate an object pose in the point-cloud data. As seen in (3), it represents how likely a pose hypothesis is close to the measurement data, and decides the weight value of each particle. Thus, the form of the evaluation function determines robustness and computational efficiency of the tracking performance.

In order to evaluate between an object model and a point-cloud data, some previous approaches used local comparison where point-to-point correspondences are used. [14][2][1]. They estimated likelihood for each point pair and sum up the result for all pairs. It is advantageous for tracking accuracy but sensitive to the noise and partially missed data. On the other hand, global comparison approach has been used for object tracking in 2D video stream [10][12][11]. Most of the

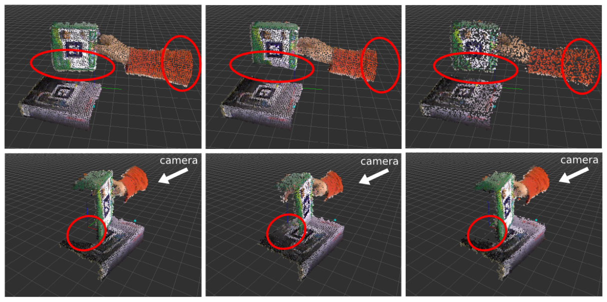


Fig. 4. Two examples of how the smoothly adapted object model (figures on the right column) can be updated comparing with current input point-clouds (figures on the left column) and fixed initial object models (figures on the middle column). An example on the upper row shows that the initially occluded but currently appeared part (red circle) is updated in the proposed object model. Another example on the lower row shows that initially appeared but currently occluded part (red circle) can be updated as well.

global comparison methods estimated color distributions of object model and hypothesis data, and compared these two distributions. Although it is a robust description, the tracking accuracy is less than local comparison, because small change of the object orientation does not affect the color distribution.

How to represent an object from a point-cloud such that it is locally efficient and globally robust? First, we select a set of *evaluation points* that are equally distributed in the interested point-cloud space. Each evaluation point represents local part of an object by calculating the proposed Joint Color-Spatial Descriptor (JCSD) based on its neighbouring measurement points. Then, the object is globally represented as a probability density of the JCSD that is estimated by interpolating the JCSDs of the neighboring evaluation points. In this way, local variation of an object can be reflected, and at the same time, robust description is retained in the global area of the object.

In this chapter, we propose Joint Color-Spatial Descriptor to combine the advantages of the local and global comparison methods. We first introduce the spatial representation of an object without considering color and then will show how to combine spatial representation and color feature into JCSD.

A. Spatial representation of an object

The spatial representation of an object is expressed as spatial probability density at multiple evaluation points. The spatial density of each evaluation point is obtained by the Kernel Density Estimation (KDE) method. We denote an arbitrary point-cloud $\mathbf{P}$ with n points. To build spatial representation on $\mathbf{P}$, first, a set of equal sized 3D grids with grid length r are created on $\mathbf{P}$ as shown in Fig. 5. Then, all grids' corners $\{\mathbf{s}_i\}_{i=1}^m$ are selected as evaluation points, where m is the number of corners. To estimate spatial density at the ith evaluation point $\mathbf{s}_i$, its neighboring points from $\mathbf{P}$ inside the kernel bandwidth are used, where the kernel bandwidth is set as the same as grid length r. Assuming the neighboring point set of $\mathbf{s}_i$ is $\mathbf{P}^{(i)}$ with n_i points, then the spatial density

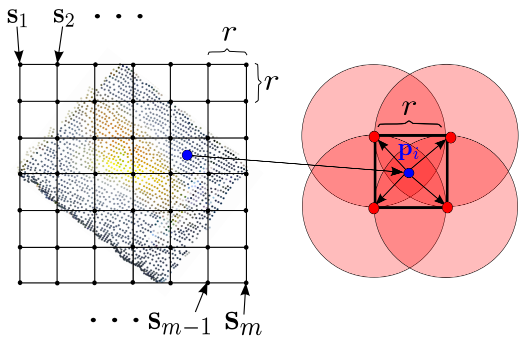


Fig. 5. A set of grids and corresponding evaluation points to represent spatial distribution of an object. Here, they are illustrated in 2D case for the simplicity. In the left figure, a bounding box with grid length r is covered on the point-cloud $\mathbf{P}$ (colored points of a book). Evaluation points are located on the grid corners $\{\mathbf{s}_i\}_{i=1}^m$. Spatial density of each evaluation point is calculated with a kernel bandwidth of r. The right figure explains the influence of a random point $\mathbf{p}_i$ from $\mathbf{P}$ to the neighboring evaluation points. Here, the point (blue color) only affect the four evaluation points (red color) because of the kernel bandwidth size (red circles). In the 3D case, there exists eight neighboring points on the corners of a cube.

of $\mathbf{P}^{(i)}$ at $\mathbf{s}_i$ can be obtained by KDE as following:

$$\hat{f}_r(\mathbf{s}_i) = \frac{1}{n}\sum_{j=1}^{n_i} K_r(\|\mathbf{s}_i - \mathbf{p}_j^{(i)}\|), \tag{7}$$

where $K_r(x)$ is a kernel function that determines the contribution of one point based on its distance. Here, we used a triangular kernel with a benefit of its computational efficiency:

$$K_r(x) = \frac{\max((r-x),0)}{r}. \tag{8}$$

With the constructed grids, we can achieve less total computational complexity for spatial density estimation. For an unorganized point-cloud, neighbor search for the evaluation points often requires k-d tree search with $\mathcal{O}(m\log n)$ complexity for m evaluation points and n number of measurement points. However, the proposed method can have linear complexity $\mathcal{O}(n)$ without neighbor search. For each point $\mathbf{p}_i$, a grid is first selected to which the point belongs. Then only $\mathbf{p}_i$'s eight surrounding evaluation points' spatial densities will be updated with $\mathbf{p}_i$, because $\mathbf{p}_i$ only contributes to these eight evaluation points as shown in Fig. 5. Therefore for each one of the n points from $\mathbf{P}$, eight update operations are needed, which results in $\mathcal{O}(8\cdot n) = \mathcal{O}(n)$ in total. The summary of the spatial density estimation for all evaluation points is shown in Algorithm 1.

B. Joint color-spatial representation

In order to represent appearance of an object, color distribution for each evaluation point should also be considered. The color feature should not only have low-dimensional space for the sake of computation memory size, but also retain discriminative property. Additionally in a dynamic environment, object pose and illumination intensity can be changing. These aspects should also be considered in the color feature selection. Wang et. al. proposed the Smoothed Color Ranging (SMR) method to describe color with 8 representative colors (red, yellow, green, cyan, blue, purple, light gray, dark gray) [17]. In their work, SMR using HSV (Hue, Saturation, Value) color space has been proven to be invariant against illumination changes.

Algorithm 1 Calculate spatial density at the evaluation points

```
Input: - $\{\mathbf{s}_i\}_{i=1}^m$ and $\mathbf{P} = \{\mathbf{p}_k\}_{k=1}^n$
Output: - $\{\hat{f}_r(\mathbf{s}_i)\}_{i=1}^m$
  - $\{\hat{f}_r(\mathbf{s}_i)\}_{i=1}^m \leftarrow 0$
  for $k = 1 : n$ do
    - Determine grid index of $\mathbf{p}_k$.
    - Determine the indexes $\{k_l\}_{l=1}^8$ of the 8 evaluation points for $\mathbf{p}_k$ based on its grid index.
    - Update each relevant evaluation point of $\mathbf{p}_k$:
    for $l = 1 : 8$ do
      - $\hat{f}_r(\mathbf{s}_{k_l}) \leftarrow \hat{f}_r(\mathbf{s}_{k_l}) + \frac{1}{n}K_r(\|\mathbf{s}_{k_l} - \mathbf{p}_k\|)$
    end for
  end for
```

Using SMR color feature, spatial density will be calculated eight times for each color range individually. To describe its color distribution, a 8-dimensional feature vector $\mathbf{h}$ has at most three non-zero elements and $\|\mathbf{h}\|_1 = 1$ will be estimated. For each evaluation point $\mathbf{s}_i$, eight different spatial densities $\{\hat{f}_r^{(c)}(\mathbf{s}_i)\}_{c=1}^{8}$ are evaluated to correlate spatial and color data. For the cth color range, to estimate the density $\hat{f}_r^{(c)}(\mathbf{s}_i)$ in that color range at evaluation point $\mathbf{s}_i$, Equation (7) is updated to:

$$\hat{f}_r^{(c)}(\mathbf{s}_i) = \frac{1}{n}\sum_{j=1}^{n_i} \mathbf{e}_c^T \cdot \mathbf{h}_j^{(i)} \cdot K_r(\|\mathbf{s}_i - \mathbf{p}_j^{(i)}\|), \tag{9}$$

where $\mathbf{e}_c \in \mathbb{R}^8$ has value 1 for cth element and 0 for all other elements.

As a result, the JCSDs for $\mathbf{P}$ with m evaluation points can be described as a Matrix $\mathbf{J_P} \in \mathbb{R}^{8\times m}$, where jth column contains the jth evaluation point's densities in eight different color ranges:

$$\mathbf{J_P} = \begin{pmatrix} \hat{f}_r^{(1)}(\mathbf{s}_1) & \hat{f}_r^{(1)}(\mathbf{s}_2) & \cdots & \hat{f}_r^{(1)}(\mathbf{s}_m) \\ \hat{f}_r^{(2)}(\mathbf{s}_1) & \hat{f}_r^{(2)}(\mathbf{s}_2) & \cdots & \hat{f}_r^{(2)}(\mathbf{s}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{f}_r^{(8)}(\mathbf{s}_1) & \hat{f}_r^{(8)}(\mathbf{s}_2) & \cdots & \hat{f}_r^{(8)}(\mathbf{s}_m) \end{pmatrix}. \tag{10}$$

C. Likelihood estimation

The likelihood estimation is a task to measure how well the object model $\mathbf{M}_{t-1}$ fits the observation data $\mathbf{Z}_t$ based on the ith hypothesis state $\mathbf{X}_t^{(i)}$. First, hypothesis data needs to be extracted from sensor data, which are the points near the object's last estimated pose for the time step $t-1$, because we can assume that object movement is limited between two consecutive frames. With this assumption, a part of points of $\mathbf{Z}_t$ that have large distance to the object model's last estimated pose $\mathbf{X}_{t-1}^{*} \otimes \mathbf{M}_{t-1}$ are removed. The remaining point cloud $\mathbf{Z}_t'$ with n points then compose the hypothesis data. For the ith particle state $\mathbf{X}_t^{(i)}$, the hypothesis data is transformed to the coordinate of the object to compare based on inverse of $\mathbf{X}_t^{(i)}$, resulting in $\mathbf{Z}_t'^{(i)} = \{\mathbf{z}_1^{(i)}, \ldots \mathbf{z}_n^{(i)}\}$.

Hypothesis evaluation is performed on each point from the hypothesis data $\mathbf{Z}'^{i}_{t}$ by estimating its fitting score to determine how well it fits to the object model's JCSD, $\mathbf{J}_{\mathbf{M}_{t-1}}$. The fitting score for $\mathbf{X}^{(i)}_{t}$ is then obtained by summing up all points' fitting score. The fitting score $s(\mathbf{z}^{(i)}_{j})$ for the jth point $\mathbf{z}^{(i)}_{j}$ from $\mathbf{Z}'^{(i)}_{t}$ is calculated based on its color feature $\mathbf{h}^{(i)}_{j}$. The JCSD $\{\hat{f}^{(c)}(\mathbf{z}^{(i)}_{j})\}^{8}_{c=1}$ for the object model at $\mathbf{z}^{(i)}_{j}$ is evaluated by using trilinear interpolation of the eight surrounding evaluation points: $\{\mathbf{s}_{j_k}\}^{8}_{k=1}$, which are the corners of a cube that surrounds $\mathbf{z}^{(i)}_{j}$:

$$\hat{f}^{(c)}(\mathbf{z}^{(i)}_{j}) = \sum_{k=1}^{8} \frac{\prod_{d=1}^{3}(\mathbf{1}_3 - |\mathbf{z}^{(i)}_{j} - \mathbf{s}_{j_k}|)_d}{l^{3}_{grid}} \cdot \mathbf{J}_{\mathbf{M}_{t-1}(c,j_k)}, \quad (11)$$

where c indicates the index of color range, $\mathbf{1}_3$ is a three-dimensional vector with all ones and $(\mathbf{x})_d$ indicates the dth element of vector $\mathbf{x}$. The fitting score of point $\mathbf{z}^{(i)}_{j}$ is then calculated by correlation between $\mathbf{h}^{(i)}_{j}$ and $\{\hat{f}^{(c)}(\mathbf{z}^{(i)}_{j})\}^{8}_{c=1}$:

$$s(\mathbf{z}^{(i)}_{j}) = \sum_{c=1}^{8} \mathbf{e}^{T}_{c} \cdot \mathbf{h}^{(i)}_{j} \cdot \hat{f}^{(c)}(\mathbf{z}^{(i)}_{j}). \quad (12)$$

As a result, the fitting score of ith particle s_i is then the summation of fitting scores of all points from $\mathbf{Z}'^{i}_{t}$:

$$s_i = \sum_{j=1}^{n} s(\mathbf{z}^{(i)}_{j}). \quad (13)$$

The summary of estimation of s_i is shown in Algorithm 2. After fitting score estimation for all particle states, the likelihood $p(\mathbf{Z}_t|\mathbf{X}^{i}_{t} \otimes \mathbf{M}_{t-1})$ is calculated as

$$p(\mathbf{Z}_t|\mathbf{X}^{(i)}_{t} \otimes \mathbf{M}_{t-1}) \propto exp(-\lambda \cdot (1 - \frac{s_i - s_{min}}{s_{max} - s_{min}})), \quad (14)$$

where s_{min} and s_{max} are the minimal and maximal score value among all hypothesis states respectively and λ is a constant controlling the preference degree to particles with higher fitting score.

Algorithm 2 Fitting score estimation

```
Input: - JCSDs of object model: $\mathbf{J}_{\mathbf{M}_{t-1}}$, $i$th hypothesis data:
    $\mathbf{Z}'^{(i)}_{t} = \{\mathbf{z}^{(i)}_{1}, ...\mathbf{z}^{(i)}_{n}\}$ and its corresponding color features:
    $\{\mathbf{h}^{(i)}_{1}, ...\mathbf{h}^{(i)}_{n}\}$
Output: - Fitting score $s_i$ of $\mathbf{Z}'^{i}_{t}$
    - $s_i \leftarrow 0$
    for $j = 1 : n$ do
        - Determine JCSD at location of $\mathbf{z}^{(i)}_{j}$: $\{\hat{f}^{(c)}(\mathbf{z}^{(i)}_{j})\}^{8}_{c=1}$
        (Equation 12).
        - Estimate fitting score of $\mathbf{z}^{(i)}_{j}$: $s(\mathbf{z}^{(i)}_{j})$ using correlation
        between $\mathbf{h}^{(i)}_{1}$ and JCSD $\{\hat{f}^{(c)}(\mathbf{z}^{(i)}_{j})\}^{8}_{c=1}$ (Eqaution 13).
        - $s_i \leftarrow s_i + s(\mathbf{z}^{(i)}_{j})$
    end for
```

IV. EXPERIMENT AND RESULT

In this section, we present a set of experiments that demonstrate pose tracking accuracy, segmentation accuracy, and real-time performance of our approach. As shown in Fig. 6, the experiments were performed by using three table-top scenarios with multiple object dynamic interactions as follows.

- *Stacking and unstacking two objects*: A human hand approaches object 1 and stacks the object onto another object 2, then the hand leaves the scene for a while and unstacks the object 1 to its original position.
- *Contacting two objects*: Two human hands grasp two objects respectively. Then both hands manipulate the objects such that both object make a contact with each other and move together.
- *Rotating an object*: A human hand grasps one object and rotates it arbitrarily.

The point-cloud data of a table are first excluded using a plane extraction method in [15], then to reduce the size of point-cloud, the remaining data were downsampled with 5mm sampling distance using VoxelGrid filter [15]. The length of grid used for JCSD was $15mm$ for all experiments. All experiments run on a standard desktop computer (CPU: Intel i5-2500 3.30GHz; GPU: Nvidia GeForce GTX 560) with ASUS Xtion Pro Live RGB-D camera.

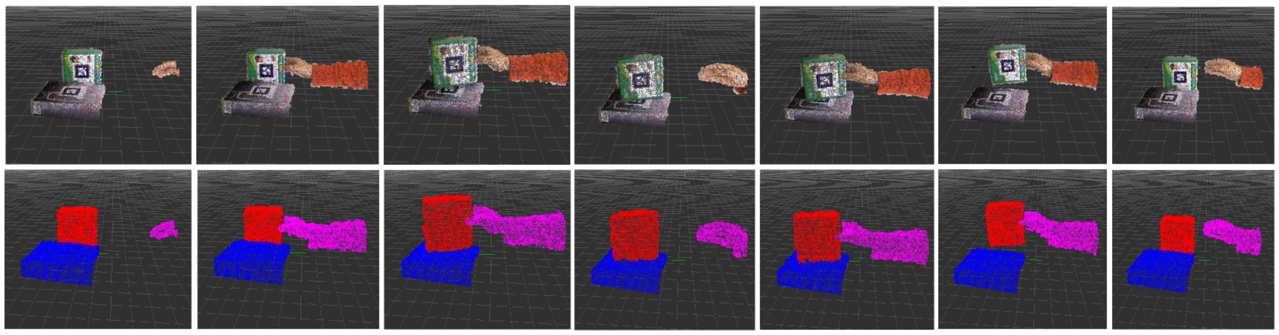
A. Tracking accuracy

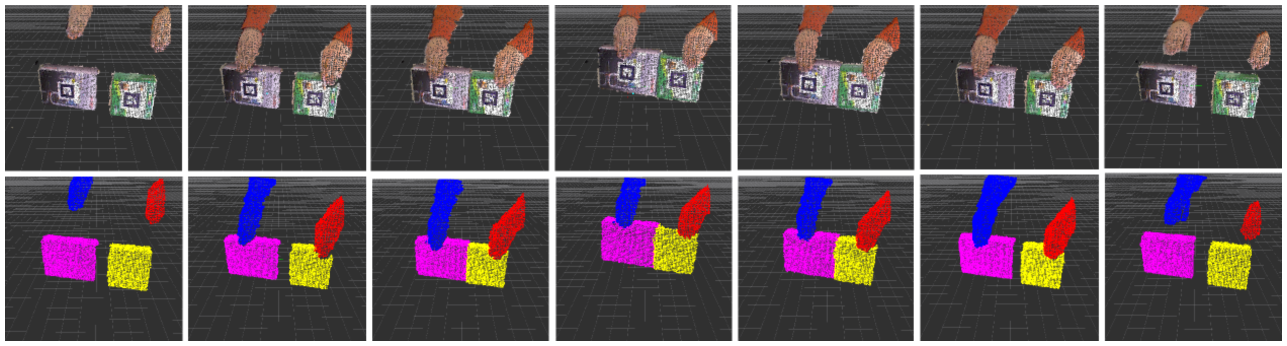
Since we performed model-free object tracking, it is hard to obtain the ground truth data of the object and hand poses. Instead of measuring the exact pose error, we used scene-fitting distance to evaluate the tracking accuracy. The fitting distance describes how well the object model fits to the sensor data based on its estimated pose, it is defined as the average distance of points from the object model in its estimated pose $\hat{\mathbf{M}}$ to the closest point from current sensor data. However, we excluded points from $\hat{\mathbf{M}}$ that are occluded by current sensor data, because they result in large distance even if the object model is in its true pose.

We compare our method with the point-point correspondence method using the likelihood function from [2] in terms of fitting distance. As shown in Fig. 7, our approach outperforms the point-point correspondence approach in both stack/unstack and multiple contact scenario, because in these two scenarios, partial occlusion of boxes caused inaccurate hypothesis likelihood of the point-point correspondence method, consequently resulting in inaccurate pose estimation.

B. Segmentation accuracy

To obtain the ground truth segmentation data, we attached Augmented Reality (AR) markers with $4cm \times 4cm$ size on the center of two objects and constructed manually their corresponding cube type point-clouds. The AR markers produced 6 DoF poses, and these values were filtered by using a Kalman Filter to obtain true marker poses. The ground truth segmentation data is then obtained by positioning the constructed cube model at the tracked marker poses onto the scene data.


(a) Stacking and unstacking two objects


(b) Contacting two objects

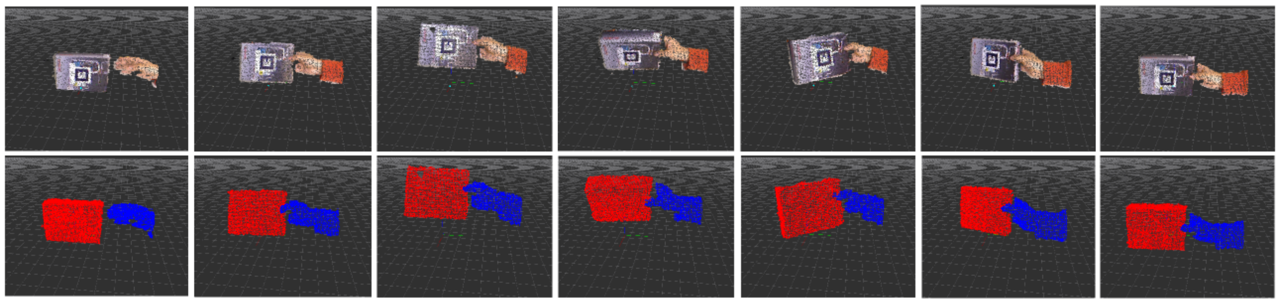

(c) Rotating an object

Fig. 6. Object segmentation result. Notice our approach can clearly segment finger in stack/unstack and rotation case.

In this experiment, we set the model update ratio $\alpha = 2\%$ and used 200 particles for the particle filtering. Since we removed the table point-cloud priorly, all points belong to a certain object. The accuracy of the object segmentation is defined as the ratio between the size of correctly associated points and the overall point-cloud size:

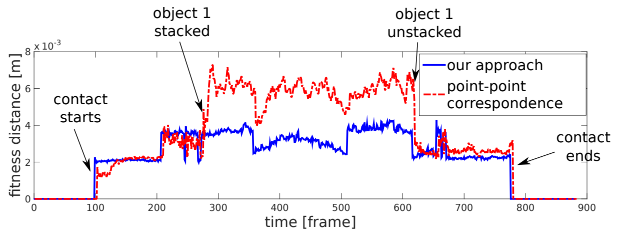
$$Accuracy = \frac{\text{\#correct associated points}}{\text{\#all points}}, \tag{15}$$

which is equivalent as the segmentation accuracy evaluation method PMOTA (Point-level Multiple Object Tracking Accuracy) that is presented in [7]. The average object segmentation accuracy for all three scenarios is 99.4%. This outperforms the result from [7], where they had similar experiment settings and achieved ca. 90% segmentation accuracy. The qualitative segmentation results can be seen in Fig. 6.

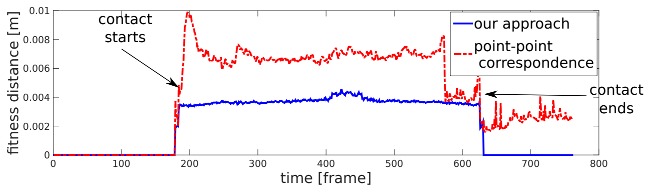
C. Computation time

We first implemented our hypothesis evaluation method with CPU programming. To evaluate one particle for an object model with ca. 900 measurement points, our CPU implementation requires in average $0.29ms$ while point-point correspondence approach needs in average $0.9ms$. This is because our approach has linear complexity, whereas point-point correspondence search requires k-d tree search, which results in $\mathcal{O}(m \log n)$ complexity.

We further implemented hypothesis evaluation method with GPU programming to parallelize the particle weighting process. To evaluate real-time performance, we used the stacking/unstacking case, where three objects are present and average point number is 3213 for each frame. For different particle numbers, we calculated the required computation time for each frame and the average fitting distance over all frames. As expected, Fig. 8 shows with more particle number, computation time grows and fitting distance decreases. After ca. 100 particles, there is no significant



(a) Stacking and unstacking two objects scenario



(b) Contacting two objects scenario

Fig. 7. Fitting distance comparison: blue line is from our approach, red line is from point-point correspondence approach [2]. In both scenarios, as soon as partial occlusion occurs, our approach outperforms.

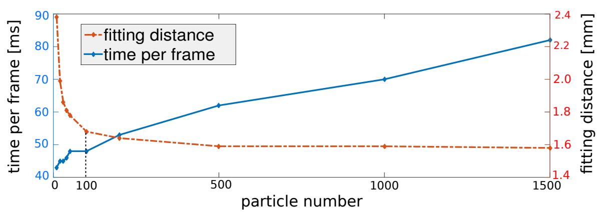


Fig. 8. Computation time and fitting distance in relation with particle number.

decrease of fitting distance while computation time still grows. Therefore, a reasonable particle size is around ca. 100 with 21 FPS computation speed.

V. CONCLUSION

In this work, we proposed a novel multiple model-free object tracking method with Joint Color-Spatial Descriptor (JCSD) for the hypothesis evaluation step inside a particle filtering framework. Using JCSD, we achieved more accurate pose tracking result in partial occlusion cases compared to point-point correspondence method [2]. The obtained pose tracking result is accurate enough to achieve object segmentation accuracy higher than 99%, which is higher than the result from [7]. In addition to the accuracy, the method is computationally efficient to be implemented with GPU in real-time. The experiment results showed that with the minimum number of particles the method can be used for tracking three objects at the same time in 21 FPS. Thanks to the accurate segmentation result of the unknown objects in real-time, on-line object learning tasks will be more investigated in future work.

ACKNOWLEDGMENT

This work has been partially supported by TUM Institute for Advanced Study, funded by the German Excellence Initiative.

REFERENCES

[1] James Anthony Brown and David W Capson. A framework for 3d model-based visual tracking using a gpu-accelerated particle filter. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):68–80, 2012.
[2] Changhyun Choi and Henrik I Christensen. Rgb-d object tracking: A particle filter approach on gpu. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1084–1091.
[3] Arnaud Doucet, Nando De Freitas, and Neil Gordon. *An introduction to sequential Monte Carlo methods*. Springer, 2001.
[4] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.
[5] Seongyong Koo, Dongheui Lee, and Dong-Soo Kwon. Gmm-based 3d object representation and robust tracking in unconstructed dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1114–1121, 2013.
[6] Seongyong Koo, Dongheui Lee, and Dong-Soo Kwon. Incremental object learning and robust tracking of multiple objects from rgb-d point set data. *Journal of Visual Communication and Image Representation*, 25(1):108–121, 2014.
[7] Seongyong Koo, Dongheui Lee, and Dong-Soo Kwon. Unsupervised object individuation from rgb-d image sequences. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4450–4457, 2014.
[8] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox. Manipulator and object tracking for in-hand 3d object modeling. *The International Journal of Robotics Research*, 30(11):1311–1327, 2011.
[9] Dongheui Lee and Yoshihiko Nakamura. Mimesis model from partial observations for a humanoid robot. *The International Journal of Robotics Research*, 29(1):60–80, 2010.
[10] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. An adaptive color-based particle filter. *Image and vision computing*, 21(1):99–110, 2003.
[11] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *Computer Vision-ECCV*, pages 28–39. Springer, 2004.
[12] Vasilis Papadourakis and Antonis Argyros. Multiple objects tracking in the presence of long-term occlusions. *Computer Vision and Image Understanding*, 114(7):835–846, 2010.
[13] Jeremie Papon, Tomas Kulvicius, Eren Erdal Aksoy, and Florentin Worgotter. Point cloud video object segmentation using a persistent supervoxel world-model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3712–3718, 2013.
[14] Jeremie Papon and Florentin Wörgötter. Spatially stratified correspondence sampling for real-time point cloud tracking. In *IEEE Conference on Applications of Computer Vision (WACV)*, 2015.
[15] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, 2011.
[16] Alexander M Schmidts, Dongheui Lee, and Angelika Peer. Imitation learning of human grasping skills from motion and force data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1002–1007, 2011.
[17] Wei Wang, Lili Chen, Dongming Chen, Shile Li, and K Kuhnlenz. Fast object recognition and 6d pose estimation using viewpoint oriented color-shape histogram. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2013.
[18] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, 2011.
[19] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.