PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking

Xinke Deng, Arsalan Mousavian, Yu Xiang, *Member, IEEE,* Fei Xia, *Member, IEEE,*
Timothy Bretl, *Member, IEEE,* Dieter Fox, *Fellow, IEEE*

***Abstract*—Tracking 6D poses of objects from videos provides rich information to a robot in performing different tasks such as manipulation and navigation. In this work, we formulate the 6D object pose tracking problem in the Rao-Blackwellized particle filtering framework, where the 3D rotation and the 3D translation of an object are decoupled. This factorization allows our approach, called PoseRBPF, to efficiently estimate the 3D translation of an object along with the full distribution over the 3D rotation. This is achieved by discretizing the rotation space in a fine-grained manner, and training an auto-encoder network to construct a codebook of feature embeddings for the discretized rotations. As a result, PoseRBPF can track objects with arbitrary symmetries while still maintaining adequate posterior distributions. Our approach achieves state-of-the-art results on two 6D pose estimation benchmarks. We open-source our implementation at https://github.com/NVlabs/PoseRBPF.**

***Index Terms*—6D object pose tracking, computer vision, state estimation.**

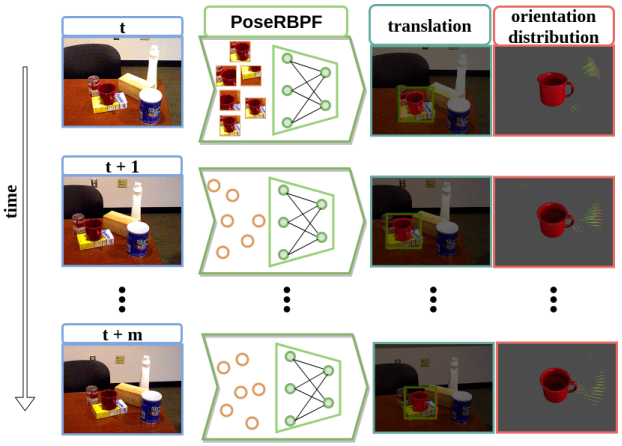


Fig. 1. Overview of our PoseRBPF framework for 6D object pose tracking. Our method leverages a Rao-Blackwellized particle filter and an auto-encoder network to estimate the 3D translation and a full distribution of the 3D rotation of a target object from a video sequence.

I. INTRODUCTION

Estimating the 6D pose of objects from camera images, i.e., 3D rotation and 3D translation of an object with respect to the camera, is an important problem in robotic applications. For instance, in robotic manipulation, 6D pose estimation of objects provides critical information to the robot for planning and executing grasps. In robotic navigation tasks, localizing objects in 3D provides useful information for planning and obstacle avoidance. Due to its significance, various efforts have been devoted to tackling the 6D pose estimation problem from both the robotics community [9, 5, 75, 67] and the computer vision community [53, 37, 21].

Traditionally, the 6D pose of an object is estimated using local-feature or template matching techniques, where features extracted from an image are matched against features or viewpoint templates generated for the 3D model of the object. Then the 6D object pose can be recovered using 2D-3D correspondences of these local features or by selecting the best matching viewpoint [9, 20, 21]. More recently, machine learning techniques have been employed to detect key points or learn better image features for matching [3, 32]. Thanks to the advances in deep learning, convolutional neural networks have recently been shown to significantly boost the pose estimation accuracy and robustness [27, 76, 48, 62, 75],

So far, the focus of image-based 6D pose estimation has been on the *accuracy of single image estimates*; most techniques ignore temporal information and provide only a single hypothesis for an object pose. In robotics, however, temporal data and information about the *uncertainty* of estimates can also be very important for tasks such as grasp planning or active sensing. Temporal tracking in video data can improve pose estimation [46, 7, 31, 10]. In the context of point-cloud based pose estimation, Kalman filtering has also been used to track 6D poses, where Bingham distributions have been shown to be well suited for orientation estimation [59]. However, unimodal estimates are not sufficient to adequately represent the complex uncertainties arising from occlusions and possible object symmetries.

In this work, we introduce a particle filter-based approach to estimate full posteriors over 6D object poses. Our approach, called PoseRBPF, factorizes the posterior into the 3D translation and the 3D rotation of the object, and uses a Rao-Blackwellized particle filter that samples object poses and estimates discretized distributions over rotations for each particle. To achieve accurate estimates, the 3D rotation is discretized at 5 degree resolution, resulting in a distribution over $72 \times 37 \times 72 = 191,808$ bins for each particle (elevation ranges only from -90 to 90 degree). To achieve real time performance, we pre-compute a codebook over embeddings for all

X. Deng was with the University of Illinois at Urbana-Champaign. A. Mousavian and Y. Xiang are with NVIDIA. F. Xia is with Stanford University. T. Bretl is with the University of Illinois at Urbana-Champaign. D. Fox is with NVIDIA and the University of Washington.

Part of the work was done during X. Deng and F. Xia's internships at NVIDIA.

Manuscript received May 27, 2020; revised October 17, 2020; Accepted December 28, 2020.

discretized rotations, where embeddings come from an auto-encoder network trained to encode the visual appearance of an object from arbitrary viewpoints at a certain scale (inspired by [60]). For each particle, PoseRBPF first uses the 3D translation to determine the center and size of the object bounding box in the image, then computes the embedding for that bounding box using the auto-encoder, and finally updates the rotation distribution by comparing the embedding vector with the pre-computed entries in the codebook using cosine distances. The weight of each particle is given by the normalization factor of the rotation distribution. Motion updates are performed efficiently by sampling from a motion model over poses and a convolution over the rotations. Fig. 1 illustrates our PoseRBPF framework for 6D object pose tracking. Experiments on the YCB-Video dataset [75] and the T-Less dataset [24] show that PoseRBPF is able to represent uncertainties arising from various types of object symmetries and can provide more accurate 6D pose estimation.

Our work makes the following main contributions:

- We introduce a novel and versatile 6D object pose estimation framework that combines a Rao-Blackwellized particle filtering with a learned auto-encoder network in an efficient and principled way.
- Our framework is able to track full distributions over 6D object poses based on RGB or RGB-D inputs. It can also do so for objects with arbitrary kinds of symmetries, without the need for any manual symmetry labeling.

Compared to the previous version of PoseRBPF [12], we introduce the following improvements in this paper:

- We propose an efficient modification inspired by [17], where we apply RoI pooling to speed up particle evaluation. Experiments show that the RGB-D tracking speed can be improved by more than 68% without sacrificing tracking accuracy.
- Apart from encoding RGB measurements using auto-encoders, we propose to encode depth measurements using separate auto-encoders, and show that the tracking performance can be significantly improved.
- We show that our pose estimation framework can be combined with a Signed Distance Function (SDF) based pose refinement module to further improve the pose estimation accuracy.
- We show that, when object detection is not available, PoseRBPF can be initialized by uniformly sampling particles over the first video frame and then refining this estimate over consecutive frames.

The rest of the paper is organized as follows. After discussing the related work, we present our Rao-Blackwellized particle filtering framework for 6D object pose tracking, followed by experimental evaluations and a conclusion.

II. RELATED WORK

6D Object Pose Estimation. Our work is closely related to recent advances in 6D object pose estimation using deep neural networks. The current trend is to augment state-of-the-art 2D object detection networks with the ability to estimate 6D object pose. For instance, [27] extend the SSD detection network [40] to 6D pose estimation by adding viewpoint classification to the network. [62] utilize the YOLO architecture [51] to detect 3D bounding box corners of objects in the images, and then recover the 6D pose by solving the PnP problem. Detecting 3D bounding box corners or object key points for 6D object pose estimation is also explored in [67, 42, 49, 58]. PoseCNN [75] designs an end-to-end network for 6D object pose estimation based on the VGG architecture [57]. Although these methods significantly improve the 6D pose estimation accuracy over the traditional methods [21, 3, 32], they still face difficulty in dealing with symmetric objects, where most methods manually specify the symmetry axis for each such object. To handle symmetric objects, [64] propose to uniformly sample rotation anchors and estimate deviations of the anchors to the target. In addition, [60, 61] introduce an implicit way of representing 3D rotations by training an auto-encoder for image reconstruction, which does not need to pre-define the symmetry axes for symmetric objects. We leverage this implicit 3D rotation representation in our work, and show how to combine it with particle filtering for 6D object pose tracking.

6D Object Pose Tracking. Another set of related work is on object tracking from videos. Early works [18, 68, 6] track objects with image features such as edges and key points. However, these methods cannot handle environments with complex texture and occlusions. They are limited in real robotic tasks. The introduction of RGB-D sensors greatly simplifies the 6D pose tracking problem since the structure of the scene can be directly perceived in compliment to the color information. Object tracking using RGB-D data receives more attention [7, 52, 55, 28, 16, 72, 71]. Although significant progress has been made, these methods still cannot work robustly in large-scale or outdoor environments, neither for small or thin objects due to the limitation of depth sensors. Recent progress on 6D pose tracking with RGB data includes [50, 65, 66, 41]. In [50], the pose of object is updated by optimizing the projected contour from the 3D model. The approach is improved in [65] with a new optimization scheme and through GPU parallelization. [66] improve pose tracking with a temporally consistent local color histogram. Meanwhile, deep neural networks are explored for 6D object pose tracking. Very recently, deep neural networks are used to predict the pose difference between consecutive frames and track the 6D object pose accordingly [36, 41, 72]. These methods significantly improve the robustness and accuracy of tracking compared to methods that use hand-crafted features [50, 65, 66]. However, object symmetries are either ignored or manually specified in these works, and 6D object pose estimation is required to initialize the tracking pipelines. We show that our framework can deal with symmetries automatically, and object pose tracking can be initialized with only 2D information, i.e., center of the object in the first video frame, or even initialized without any prior spatial information by sampling particles uniformly in the image.

Particle Filtering. The particle filtering framework has been widely applied to different tracking applications in the literature [45, 56, 29, 54], thanks to its flexibility in incorporating different observation models and motion priors. Meanwhile, it offers a rigorous probabilistic formulation to es-

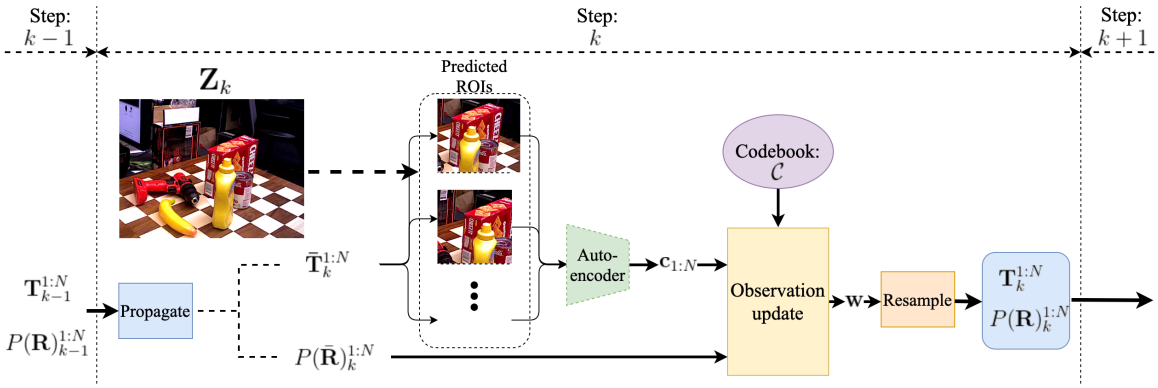


Fig. 2. Architecture of PoseRBPF. For each particle, the *rotation distribution* is estimated conditioned on translation estimation, while the translation estimation is evaluated with the corresponding RoIs.

timate uncertainty in the tracking results. Different approaches have also been proposed to track the poses of objects using particle filters [2, 8, 47, 74, 35]. However, in order to achieve good tracking performance, a particle filter requires a strong observation model. Also, the tracking frame rate is limited by the particle sampling and evaluation efficiency. In this work, we factorize the 6D object pose tracking problem and deploy Rao-Blackwellized particle filters [15], which have been shown to scale to complex estimation problems such as SLAM [63, 44] and multi-model target tracking [34, 54]. We also employ a deep neural network as an observation model that provides robust estimates for object orientations even under occlusions and symmetries. Our design allows us to evaluate all possible orientations in parallel using an efficient GPU implementation. As a result, our method can track the distribution of the 6D pose of an object at 20fps.

III. 6D Object Pose Tracking with PoseRBPF

In this section, we first state the problem of 6D object pose tracking, and provide a high-level overview of PoseRBPF. After formulating the problem in a particle filtering framework, we describe in detail how to utilize a deep neural network to compute the likelihoods of the particles and to achieve an efficient sampling strategy for tracking.

A. Problem Formulation

Given a sequence of input images $\mathbf{Z}_{1:k}$ up to time k, the goal of 6D object pose tracking of an object is to estimate the rigid body transformation between the camera coordinate frame $\mathcal{C}$ and the object coordinate frame $\mathcal{O}$ for every image in the image stream. We assume that the 2D center (u, v) of the object in the first image is provided by an object detector such as [17, 51] for pose tracking initialization, and the 3D CAD model of the object is known. The rigid body transformation consists of a 3D rotation $\mathbf{R}_k$ and a 3D translation $\mathbf{T}_k$ of the object at time k. In this paper, instead of providing a single estimation $\{\mathbf{R}_k, \mathbf{T}_k\}$, our primary goal is to estimate the posterior distribution of the 6D pose of an object $P(\mathbf{R}_k, \mathbf{T}_k|\mathbf{Z}_{1:k})$.

B. Overview of PoseRBPF

Fig. 2 illustrates the architecture of our 6D object pose tracking framework. Each particle in PoseRBPF is represented by a translation hypothesis and a rotation distribution conditioned on the translation hypothesis. In each step, the particles are first propagated according to a motion model described in Sec. III-E. Each particle determines an unique Region of Interest (RoI) according to its translation, and the RoI is fed into an auto-encoder network to compute a feature embedding. The observation likelihood is computed by matching the embedding with the embeddings in a pre-computed codebook, which is detailed in Sec. III-D. Finally, the weights of the particles can be computed with the observation likelihoods, and the particles are resampled accordingly.

C. Rao-Blackwellized Particle Filter Formulation

To estimate posterior distribution $P(\mathbf{R}_k, \mathbf{T}_k|\mathbf{Z}_{1:k})$ using a standard particle filter [11, 63] to sample over this 6D space is not feasible, especially when there is large uncertainty over the rotation of the object. Such uncertainties occur frequently when objects are heavily occluded or have symmetries that result in multiple valid rotation hypotheses. We thus propose to factorize the 6D pose estimation problem into 3D rotation estimation and 3D translation estimation. This idea is based on the observation that the 3D translation can be estimated from the location and the size of the object in the image. The translation estimation provides the center and scale of the object in the image, based on which the 3D rotation can be estimated from the appearance of the object inside the bounding box. Specifically, we decompose the posterior into:

$$P(\mathbf{R}_k, \mathbf{T}_k|\mathbf{Z}_{1:k}) = P(\mathbf{T}_k|\mathbf{Z}_{1:k})P(\mathbf{R}_k|\mathbf{T}_k, \mathbf{Z}_{1:k}), \quad (1)$$

where $P(\mathbf{T}_k|\mathbf{Z}_{1:k})$ encodes the location and scale of the object, and $P(\mathbf{R}_k|\mathbf{T}_k, \mathbf{Z}_{1:k})$ models the rotation distribution conditioned on the translation and the images.

This factorization directly leads to an efficient sampling scheme for a Rao-Blackwellized particle filter [15, 63], where the posterior at time k is approximated by a set of N weighted samples $\mathcal{X}_k = \{\mathbf{T}_k^i, P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_{1:k}), w_k^i\}_{i=1}^N$. Here, $\mathbf{T}_k^i$ denotes the translation of the ith particle, $P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_{1:k})$ denotes the discrete *distribution* of the particle over the object

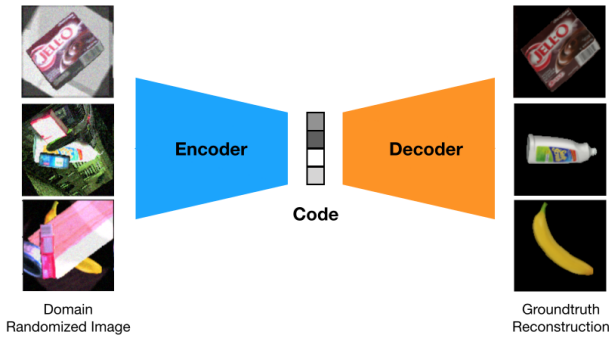


Fig. 3. Illustration of the inputs and outputs of the auto-encoder. Images with different lighting, background and occlusion are feed into the network to reconstruct synthetic images of the objects from the same 6D poses. The encoder generates a feature embedding (code) of the input image.

rotation conditioned on the translation and the images, and w_k^i is the importance weight. To achieve accurate pose estimation, the 3D object rotation consisting of azimuth, elevation, and in-plane rotation is discretized into bins of size 5 degree, resulting in a distribution over $72 \times 37 \times 72 = 191,808$ bins for each particle (elevation ranges only from -90 to 90 degrees). At every time step k, the particles are propagated through a motion model to generate a new set of particles $\mathcal{X}_{k+1}$, from which we can estimate the 6D pose distribution.

According to the particle filter formulation, $P(\mathbf{T}_k|\mathbf{Z}_{1:k}) = \sum_i w_k^i \delta(\mathbf{T}_k - \mathbf{T}_k^i)$, where $\delta(\cdot)$ represents a Dirac delta function at zero. The weights w_k^i can be computed as:

$$w_k^i \propto P(\mathbf{Z}_k|\mathbf{T}_k^i) \tag{2}$$

$$= \int P(\mathbf{Z}_k|\mathbf{T}_k^i, \mathbf{R}_k) P(\mathbf{R}_k) d\mathbf{R}_k \tag{3}$$

$$\approx \sum_j P(\mathbf{Z}_k|\mathbf{T}_k^i, \mathbf{R}_k^j) P(\mathbf{R}_k^j), \tag{4}$$

where $\mathbf{R}_k^j$ denotes discretized rotations.

D. Observation Likelihoods

The observation likelihood $P(\mathbf{Z}_k|\mathbf{T}_k, \mathbf{R}_k)$ measures the compatibility of the observation $\mathbf{Z}_k$ with the object pose at the 3D rotation $\mathbf{R}_k$ and the 3D translation $\mathbf{T}_k$. Intuitively, a 6D object pose estimation method, such as [27, 62, 75], can be employed to estimate the observation likelihoods. However, these methods only provide a single estimation of the 6D pose instead of estimating a probability distribution, i.e., there is no uncertainty in their estimation. Also, these methods are computationally expensive if we would like to evaluate a large number of samples in the particle filtering.

Ideally, if we can synthetically generate an image of the object with the pose $(\mathbf{R}_k, \mathbf{T}_k)$ into the same scene as the observation $\mathbf{Z}_k$, we can compare the synthetic image with the input image $\mathbf{Z}_k$ to measure the likelihoods. However, this is not feasible since it is very difficult to synthesize the same lighting, background or even occlusions between objects as in the input video frame. In contrast, it is straightforward to render a synthetic image of the object using constant lighting, blank background and no occlusion, given the 3D model of the object. Therefore, inspired by [60], we apply an auto-encoder to transform the observation $\mathbf{Z}_k$ into the same domain as the synthetic rendering of the object. Then we can compare image features in the synthetic domain to measure the likelihoods of 6D poses efficiently.

1) Auto-encoder: An auto-encoder is trained to map an image $\mathbf{Z}$ of the target object with pose $(\mathbf{R}, \mathbf{T})$ to a synthetic image $\mathbf{Z}'$ of the object rendered from the same pose, where the synthetic image $\mathbf{Z}'$ is rendered using constant lighting, and there is no background and occlusion in the synthetic image. In this way, the auto-encoder is forced to map images with different lighting, background and occlusion to the common synthetic domain. Fig. 3 illustrates the input and output of the auto-encoder during training. In addition, the auto-encoder learns a feature embedding $f(\mathbf{Z})$ of the input image.

Instead of training the auto-encoder to reconstruct images with arbitrary 6D poses, which makes the training challenging, we fix the 3D translation to a canonical one $\mathbf{T}_0 = (0, 0, z)^T$, where z is a constant distance. The canonical translation indicates that the target object is in front of the camera with distance z. z can be computed by optimizing the distance of the 3D model to the camera which makes sure renderings from all the rotations well-fitted to the training image size (128x128 in our experiments). The 3D rotation $\mathbf{R}$ is uniformly sampled during training. After training, for each discretized 3D rotation $\mathbf{R}^i$, a feature embedding $f(\mathbf{Z}(\mathbf{R}^i, \mathbf{T}_0))$ is computed using the encoder, where $\mathbf{Z}(\mathbf{R}^i, \mathbf{T}_0)$ denotes a rendered image of the target object from pose $(\mathbf{R}^i, \mathbf{T}_0)$. We consider the set of all the feature embeddings of the discretized 3D rotations to be the codebook of the target, and we show how to compute the likelihoods using the codebook next.

2) Codebook Matching: Given a 3D translation hypothesis $\mathbf{T}_k$, we can crop a RoI from the image $\mathbf{Z}_k$, and then feed the RoI into the encoder to compute a feature embedding of the RoI. Specifically, the 3D translation $\mathbf{T}_k = (x_k, y_k, z_k)^T$ is projected to the image to find the center (u_k, v_k) of the RoI :

$$\begin{bmatrix} u_k \\ v_k \end{bmatrix} = \begin{bmatrix} f_x \frac{x_k}{z_k} + p_x \\ f_y \frac{y_k}{z_k} + p_y \end{bmatrix}, \tag{5}$$

where f_x and f_y indicate the focal lengths of the camera, and $(p_x, p_y)^T$ is the principal point. The size of the RoI is determined by $\frac{z_k}{z} s$, where z and s are the canonical distance and the RoI size in training the auto-encoder, respectively. Note that each RoI is a square region in our case, which makes the RoI independent from the rotation of the object.

The RoI is feed into the encoder to compute the feature embedding $\mathbf{c} = f(\mathbf{Z}_k(\mathbf{T}_k))$. Finally, we compute the cosine distance, which is also referred as a similarity score, between the feature embedding of the RoI and a code in the codebook to measure the observation likelihood:

$$P(\mathbf{Z}_k|\mathbf{T}_k, \mathbf{R}_c^j) = \phi\Big(\frac{\mathbf{c} \cdot f(\mathbf{Z}(\mathbf{R}_c^j, \mathbf{T}_0))}{\|\mathbf{c}\| \cdot \|f(\mathbf{Z}(\mathbf{R}_c^j, \mathbf{T}_0))\|}\Big), \tag{6}$$

where $\mathbf{R}_c^j$ is one of the discretized rotations in the codebook, and $\phi(\cdot)$ is a Gaussian probability density function centered at the maximum cosine distance among all the codes in the codebook for all the particles. Fig. 4 illustrates the computation of the rotation likelihoods by the codebook matching. In this way,

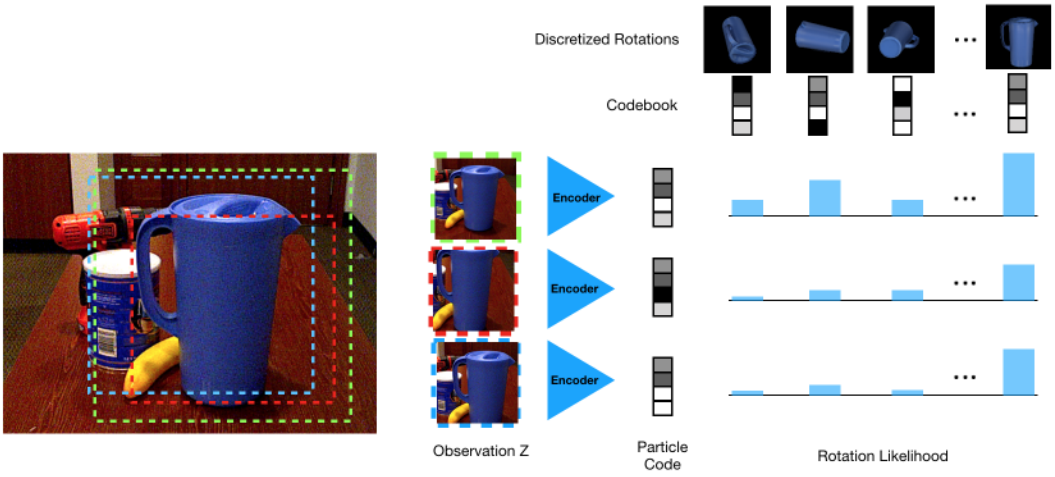


Fig. 4. Illustration of the computation for the conditional rotation likelihood by codebook matching. Left) Each particle crops the image based on its translation hypothesis. The RoI for each particle is resized and the corresponding code is computed using the encoder. Right) The rotation distribution $P(\mathbf{R}|\mathbf{Z},\mathbf{T})$ is computed from the distance between the code for each hypothesis and those in the codebook.

Fig. 5. Visualization of reconstruction of the RoIs from auto-encoder. Left is the ground truth RoI. The other two columns show the reconstruction with shifting and scale change. As it is shown, the reconstruction quality degrades with deviations from the ground truth RoI. In this example, the similarity score drops from 0.91 to 0.62 and 0.72 with the deviations respectively. This property makes the auto-encoder a suitable choice for computing the observation likelihood.

we can also obtain a probabilistic likelihood distribution of all the rotations in the codebook given a translation according to Bayes rule:

$$P(\mathbf{R}_c^j|\mathbf{T}_k,\mathbf{Z}_k) \propto P(\mathbf{Z}_k|\mathbf{T}_k,\mathbf{R}_c^j). \tag{7}$$

Since the auto-encoder is trained with the object being at the center of the image and at a certain scale, i.e., with the canonical translation $\mathbf{T}_0$, any change in scale or deviation of the object from the image center results in poor reconstructions (see Fig. 5). Particles with incorrect translations would generate RoIs where the object is not in the center of the RoI or with the wrong scale. Then we can check the reconstruction quality of the RoI to measure the likelihood of the translation hypothesis. Intuitively, if the translation $\mathbf{T}_k$ is correct, the similarity scores in Eq. (6) for rotation $\mathbf{R}^i$ that is close to the ground truth rotation would be high. Finally, the translation likelihood $P(\mathbf{Z}_k|\mathbf{T}_k)$ can be computed as in Eq. (4).

E. Motion Priors

Motion prior is used to propagate the distribution of the poses from the previous time step $k-1$ to the current time step k. We use a constant velocity model to propagate the probability distribution of the 3D translation:

$$P(\mathbf{T}_k|\mathbf{T}_{k-1},\mathbf{T}_{k-2}) = \mathcal{N}\left(\mathbf{T}_{k-1} + \alpha(\mathbf{T}_{k-1}-\mathbf{T}_{k-2}), \mathbf{\Sigma}_\mathbf{T}\right), \tag{8}$$

where $\mathcal{N}(\mu,\mathbf{\Sigma})$ denotes the multivariate normal distribution with mean μ and covariance matrix $\mathbf{\Sigma}$, and α is a hyper-parameter of the constant velocity model. The rotation prior is defined as a normal distribution with mean $\mathbf{R}_{k-1}$ and fixed covariance $\mathbf{\Sigma}_\mathbf{R}$:

$$P(\mathbf{R}_k|\mathbf{R}_{k-1}) = \mathcal{N}\left(\mathbf{R}_{k-1}, \mathbf{\Sigma}_\mathbf{R}\right), \tag{9}$$

where we represent the rotation $\mathbf{R}$ using Euler angles. Then the rotation prior can be implemented by a convolution on the previous rotation distribution with a 3D Gaussian kernel.

F. 6D Object Pose Tracking Framework

The tracking process can be initialized from any 2D object detector that outputs a 2D bounding box of the target object. Given the first frame $\mathbf{Z}_1$, we backproject the center of the 2D bounding box to compute the (x,y) components of the 3D translation and sample different zs uniformly to generate a set of translation hypotheses. The translation $\mathbf{T}_1$ with the highest likelihood $P(\mathbf{Z}_1|\mathbf{T})$ is used as the initial hypothesis and $P(\mathbf{R}|\mathbf{T}_1,\mathbf{Z}_1)$ as the initial rotation distribution.

At each following frame, we first propagate the N particles with the motion priors. Then the particles are updated with the latest observation $\mathbf{Z}_k$. Specifically, for each particle, the translation estimation $\mathbf{T}_k^i$ is used to compute the RoI of the object in image $\mathbf{Z}_k$. The resulting RoI is passed through the auto-encoder to compute the corresponding code. For each particle, the rotation distribution is updated with:

$$P(\mathbf{R}_k|\mathbf{T}_k^i,\mathbf{Z}_{1:k}) = \eta P(\mathbf{R}_k|\mathbf{T}_k^i,\mathbf{Z}_k)P(\mathbf{R}_k|\mathbf{R}_{k-1})P(\mathbf{R}_{k-1}),$$

Algorithm 1: 6D Pose Tracking with PoseRBPF

input : $\mathbf{Z}_k$, $(\mathbf{T}_{k-1}^{1:N}, P(\mathbf{R})_{k-1}^{1:N})$
output: $(\mathbf{T}_k^{1:N}, P(\mathbf{R})_k^{1:N})$
begin
  $\{w^i\}_{i=1}^N \leftarrow \emptyset$;
  $(\bar{\mathbf{T}}_k^{1:N}, P(\bar{\mathbf{R}})_k^{1:N}) \leftarrow$
   $Propagate(\mathbf{T}_{k-1}^{1:N}, P(\mathbf{R})_{k-1}^{1:N})$;
  for $(\bar{\mathbf{T}}_k^i, P(\bar{\mathbf{R}})_k^i) \in (\bar{\mathbf{T}}_k^{1:N}, P(\bar{\mathbf{R}})_k^{1:N})$ **do**
    $P(\bar{\mathbf{R}})_k^i \leftarrow Codebook\_Match(\mathbf{Z}_k, \bar{\mathbf{T}}_k^i) * P(\bar{\mathbf{R}})_k^i$;
    $w^i \leftarrow Evaluate(\mathbf{Z}_k, \bar{\mathbf{T}}_k^i, P(\bar{\mathbf{R}}_k^i))$;
  end
  $(\mathbf{T}_k^{1:N}, P(\mathbf{R})_k^{1:N}) \leftarrow$
   $Resample(\bar{\mathbf{T}}_k^{1:N}, P(\bar{\mathbf{R}})_k^{1:N}, \{w^i\}_{i=1}^N)$;
end

where $P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_k)$ is the rotation distribution defined in Eq. (7), $P(\mathbf{R}_k|\mathbf{R}_{k-1})$ is the motion prior, and η is a constant normalizer. Finally, we compute the posterior of the translation $P(\mathbf{T}_k^i|\mathbf{Z}_{1:k})$ with the weight w^i of this particle according to Eq. (4). The systematic resampling method [14] is used to resample the particles according to the weights $w^{1:N}$.

Some robotic tasks require the expectation of the 6D pose of the object $(\mathbf{T}_k^E, \mathbf{R}_k^E)$ from the particle filter for decision making. The translation expectation $\mathbf{T}_k^E$ can be computed with

$$\mathbf{T}_k^E = \sum_{i=1}^{N} w_k^i \mathbf{T}_k^i \tag{10}$$

for all the N particles due to the uni-modal nature of translation in the object tracking task. Computing the rotation expectation $\mathbf{R}_k^E$ is less obvious since the distribution $P(\mathbf{R}_k)$ might be multi-modal and simply performing weighted averaging over all the discrete rotations is not meaningful. To compute the rotation expectation, we first compute the expectation of rotation distribution $P(\mathbf{R}_k^E)$ with

$$P(\mathbf{R}_k^E) = \sum_{i=1}^{N} w_k^i P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_{1:k}). \tag{11}$$

The rotation expectation $\mathbf{R}_k^E$ is then computed by weighted averaging the discrete egocentric rotations within a neighborhood of the previous rotation expectation $\mathbf{R}_{k-1}^E$ using the quaternion averaging method proposed in [43]. The difference between egocentric orientation and allocentric orientation is described in [33].

Performing codebook matching with the estimated RoIs also provides a way to detect tracking failures. We can first find the maximum similarity score among all the particles. Then if the maximal score is lower than a pre-defined threshold, we determine it is a tracking failure. Algorithm 1 summarizes our Rao-Blackwellized particle filter for 6D object pose tracking.

G. RGB-D Extension of PoseRBPF

PoseRBPF is a versatile framework, and can be extended with additional depth measurements in the observation likelihood. With the RGB input $\mathbf{Z}_k^C$ and the additional depth measurements $\mathbf{Z}_k^D$, the observation likelihood $P(\mathbf{Z}_k|\mathbf{T}_k, \mathbf{R}_k)$ can be rewritten as:

$$\begin{aligned} P(\mathbf{Z}_k|\mathbf{T}_k, \mathbf{R}_k) &= P(\mathbf{Z}_k^C, \mathbf{Z}_k^D|\mathbf{T}_k, \mathbf{R}_k) \\ &= P(\mathbf{Z}_k^C|\mathbf{T}_k, \mathbf{R}_k) P(\mathbf{Z}_k^D|\mathbf{T}_k, \mathbf{R}_k). \end{aligned} \tag{12}$$

We propose two ways to compute the observation likelihood for depth measurements $P(\mathbf{Z}_k^D|\mathbf{T}_k, \mathbf{R}_k)$. In the first method, a depth map can be rendered according to the translation $\mathbf{T}_k^i$ and the most likely rotation $\mathbf{R}_k^*$ from the rotation distribution $P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_k^i)$ of each particle. The observation likelihood of depth can be computed by comparing the rendered depth map and the depth measurements. In the second method, depth measurements can be also encoded with an auto-encoder, and the observation likelihood of depth can be computed similar to RGB images. In addition to filtering with PoseRBPF, depth can be used to further refine the estimated pose with the Signed Distance Function (SDF) of the 3D object model.

1) Render and Compare: To compute the likelihood $P(\mathbf{Z}_k^D|\mathbf{T}_k^i, \mathbf{R}_k)$ for the ith particle, we can render the object with the pose $(\mathbf{T}_k^i, \mathbf{R}_k^*)$, where $\mathbf{R}_k^* = \arg\max_{\mathbf{R}_k} P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_k)$. For comparing the rendered depth map $\hat{\mathbf{Z}}_k^{Di}$ with the depth measurements $\mathbf{Z}_k^D$, we estimate the visibility mask $\hat{V}_k^i = \{\forall p, |\hat{\mathbf{Z}}_k^{Di}(p) - \mathbf{Z}_k^D(p)| < m\}$, where p indicates a pixel in the image and m is a small positive constant margin to account for sensor noises. Therefore, a rendered pixel p with depth within $\mathbf{Z}_k^D(p) \pm m$ is determined as visible. With the estimated visibility mask, the *visible depth discrepancy* between the two depth maps is computed as:

$$\Delta_k^i(\hat{\mathbf{Z}}_k^{Di}, \mathbf{Z}_k^D, \hat{V}_k^i, \tau) = \operatorname*{avg}_{p \in \hat{V}_k^i} \Big(\min\big(\frac{|\mathbf{Z}_k^D(p) - \hat{\mathbf{Z}}_k^{Di}(p)|}{\tau}, 1\big)\Big), \tag{13}$$

where τ is a pre-defined threshold. For every particle, we compute its *depth score* as $s_d^i = v_k^i(1 - \Delta_k^i)$, where v_k^i is the visibility ratio of the object, i.e., the number of visible pixels according to the visibility mask divided by the total number of pixels rendered. Finally, we compute $P(\mathbf{Z}_k^D|\mathbf{T}_k^i, \mathbf{R}_k)$ as $\phi'(s_d^i)$, where $\phi'(\cdot)$ is a gaussian probability density function centered at the maximum depth score among all the particles.

2) Encode Depth Measurements: Another way to exploit depth measurements is computing the observation likelihood $P(\mathbf{Z}_k^D|\mathbf{T}_k, \mathbf{R}_k)$ with separate auto-encoder network. For each particle $\big\{\mathbf{T}_k^i, P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_{1:k})\big\}$, we first normalize the depth measurements with

$$\bar{\mathbf{Z}}_k^D = f_c\big(\frac{\mathbf{Z}_k^D - z_k^i}{d} + 0.5\big), \tag{14}$$

where d is the diameter of the object, z_k^i represents the depth of the object, and $f_c(x)$ is a clamping function defined as $f_c(x) = \max(0, \min(1, x))$. Essentially, Eq. (14) normalizes depth measurements to $[0, 1]$ according to the particles. Our experiments in Sec. IV show that the normalization significantly improves the tracking accuracy compared to encoding

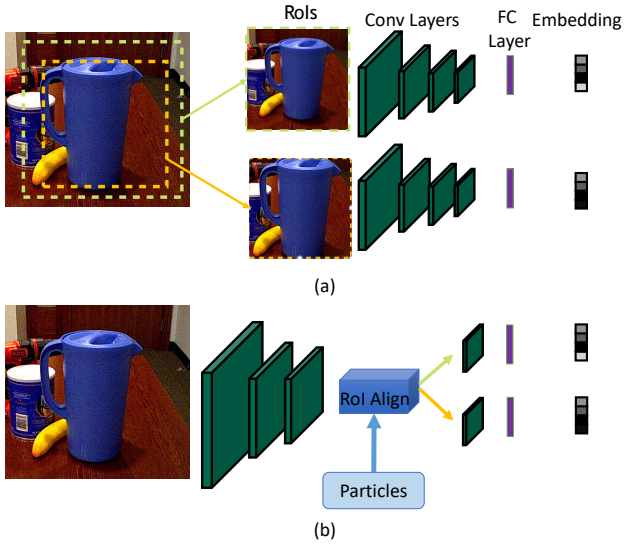


Fig. 6. A comparison between the auto-encoders in the original PoseRBPF and Fast PoseRBPF. In the original PoseRBPF, each particle crops its RoI directly on the input image and pass through the encoder individually (a); in Fast PoseRBPF, each particle crops on the shared feature maps, so that the early convolutions can be shared among particles (b).

the original depth values. We train a separate auto-encoder for the normalized depth, and estimate $P(\mathbf{Z}_k^D|\mathbf{T}_k, \mathbf{R}_k)$ in the same way as estimating the likelihood for the RGB images $P(\mathbf{Z}_k^C|\mathbf{T}_k, \mathbf{R}_k)$. The observation likelihoods are fused in the particle filter framework according to Eq. (12).

3) Pose Refinement with SDFs: The estimated object pose from PoseRBPF can be further refined by matching the 3D points from the depth measurements against the Signed Distance Function (SDF) of the target object. We first estimate the segmentation mask $\bar{\mathbf{V}}$ of the object by rendering the object according to the pose expectation $(\mathbf{T}_k^E, \mathbf{R}_k^E)$, and comparing with the depth measurements as described in Sec. III-G1. The point cloud of the object $\mathbf{P}_{\text{obj}}$ can be computed by back-projecting the pixels in $\bar{\mathbf{V}}$:

$$\mathbf{P}_{\text{obj}} = \left\{ \mathbf{Z}_k^D(p)\mathbf{K}^{-1}\bar{p}^T, p \in \bar{\mathbf{V}} \right\}, \tag{15}$$

where $\mathbf{K}$ represents the intrinsic matrix of the camera, $\bar{p}$ represents the homogeneous coordinates of the pixel p.

After computing the 3D points on the object, we optimize the pose by matching these points against the Signed Distance Function (SDF) of the object model as in [55]. The optimization problem we solve is

$$(\mathbf{T}^*, \mathbf{R}^*) = \arg\min_{\mathbf{T},\mathbf{R}} \sum_{\mathbf{p}_i \in \mathbf{P}_{\text{obj}}} \left| \mathbf{SDF}_{\text{obj}}(\mathbf{p}_i, \mathbf{T}, \mathbf{R}) \right|, \tag{16}$$

where $\mathbf{p}_i$ is a 3D point in the point cloud $\mathbf{P}_{\text{obj}}$, $\mathbf{SDF}_{\text{obj}}(\mathbf{p}_i, \mathbf{T}, \mathbf{R})$ denotes the signed distance value by transforming the point $\mathbf{p}_i$ from the camera coordinate into the object model coordinate using pose $(\mathbf{T}, \mathbf{R})$. The optimization problem can be solved in an iterative manner with gradient-based methods. In our approach, the solution is initialized with the pose expectation $(\mathbf{T}_k^E, \mathbf{R}_k^E)$, and optimized with the Adam optimizer [30].

H. Fast PoseRBPF

Inspired by [17], we propose a modification on PoseRBPF to accelerate the evaluation of the particles. It is observed that there are significant overlaps among the RoIs of the particles during tracking. As shown in Fig. 6, instead of cropping the RoIs directly on the input image and passing them through the encoder individually for each particle, we propose to crop the feature map from the encoder according to the RoIs, so that the early convolutions can be shared among particles. We call this efficient variant as Fast PoseRBPF.

Specifically, denoting the mean translation of the particles as $\bar{\mathbf{T}} = (\bar{x}, \bar{y}, \bar{z})^T$, its projection on the 2D image $(\bar{u}, \bar{v})^T$ can be computed with Eq. (5). We first crop the input image with center $(\bar{u}, \bar{v})^T$ and size $\beta\frac{\bar{z}}{z}s$, where z and s are the canonical distance and size in training the auto-encoder, and β is a scaling factor to ensure the cropped image is big enough to cover all the RoIs for the particles. The cropped image is passed through the first three convolution layers to compute the feature map. For each particle with translation $\mathbf{T_k} = (x_k, y_k, z_k)^T$ and projected 2D center $(u_k, v_k)^T$, the size of the corresponding RoI on the feature map can be computed as $\frac{\bar{z}}{\beta z_k}s_o$, where s_o represents the size of the feature map after the shared convolution layers. The center (u_c, v_c) can be computed as

$$(u_c, v_c) = \left(\frac{(u_k - \bar{u})\bar{z}}{z} \cdot \frac{s_o}{\beta s} + \frac{s_o}{2}, \frac{(v_k - \bar{v})\bar{z}}{z} \cdot \frac{s_o}{\beta s} + \frac{s_o}{2}\right).$$

The feature map is cropped with the RoI Align operation proposed in [19], and the cropped features are fed into the following network layers separately to generate the codes of the particles.

IV. EXPERIMENTS

A. Implementation Details

1) Networks Architecture: The auto-encoder for RGB inputs is the same as the one in [60]. It takes in images of size 128×128, and consists of four 5×5 convolutional layers and four 5×5 deconvolutional layers for the encoder and the decoder, respectively. After the convolutional layers, a fully connected layer is used to produce 128 dimensional embeddings. We use a similar architecture but reduce the number of channels in the convolution layers by half for the depth auto-encoder to avoid over-fitting. For Fast PoseRBPF, scaling factor β is set to 2. So the size of the input images for generating the feature map is 256×256. Particles share the first three layers of convolution operations, therefore s_o is 32.

2) Training: The RGB-D training data is purely synthetic and generated by rendering an object at random rotations according to the given CAD models. The rendered images are superimposed at random crops of the MS-COCO dataset [38] at resolution 128×128. The depth data is first normalized according to Eq. (14) with $d = 0.4$. The background data for normalized depth is generated by averaging the RGB channels at random crops in MS-COCO dataset. In addition to the target object, three additional objects are sampled at random locations and scales to generate training data with occlusions. The target object is positioned at the center of the image

and jittered with 5 pixels. The object is sampled uniformly at scales between 0.975 and 1.025 with random lighting. Color is randomized in HSV space. We also add Gaussian noises with standard deviation 0.1 and 0.5 to RGB and normalized depth, respectively, to reduce the gap between the real and synthetic data. The images are rendered online for each training step to provide a more diverse set of training data.

The auto-encoders are trained for each object separately for $150,000$ iterations with batch size of 64 using the Adam optimizer with learning rate 0.0002. The auto-encoder is optimized with the L2 loss on the N pixels with largest reconstruction errors. Larger Ns are more suitable for textured objects to capture more details. We use $N = 2000$ for textured objects and $N = 1000$ for non-textured objects.

3) Testing: During test time, the standard deviation used to compute observation likelihoods in Eq. (6) is 0.05. The codebook for each object is pre-computed offline and loaded during test time. Computation of observation likelihoods is performed efficiently on a GPU. With depth input, for the render and compare approach described in III-G, the margin m is chosen as $2cm$ and the threshold τ is set to $3cm$ in our implementation. The standard deviation of $\phi'(\cdot)$ is set to 0.05. Since rendering an individual depth map for each particle can be expensive and the primary goal for the render and compare approach is to improve the translation estimation, in our implementation, we render the depth map with the most likely pose for all the particles during tracking, then the rendered depth map is adjusted by compensating the difference between the translation used for rendering and the translation for each particle. For initialization, the depth maps are rendered individually for each particle. For refining pose estimation with SDFs, we set the learning rate for the Adam optimizer to 0.01, and run the optimizer for 100 steps. We conduct experiments on a desktop computer with a Intel i7 CPU and a NVIDIA TitanXp GPU.

B. Datasets

We evaluate our method on two datasets for 6D object pose estimation: the T-LESS dataset [24] and the YCB Video dataset [75]. The T-LESS dataset contains RGB-D sequences of 30 non-textured industrial objects. Evaluation is performed on 20 test scenes. The T-LESS dataset is challenging because the objects do not have texture and they have various forms of symmetries and occlusions. The YCB Video dataset contains RGB-D video sequences of 21 objects from the YCB Object and Model Set [4]. It contains textured and textureless household objects in different arrangements. In both datasets, objects are annotated with 6D poses.

C. Evaluation Metrics

For the T-LESS dataset, we use *Visible Surface Discrepancy* err_{vsd} [23] to evaluate the quality of the pose estimation. It is computed as:

$$
\begin{aligned}
err_{\text{vsd}} &= \underset{p \in \hat{V} \cup V_{\text{gt}}}{\text{avg}} \; c(p, \hat{D}, D_{\text{gt}}, \tau) \\
c(p, \hat{D}, D_{\text{gt}}, \tau) &= \begin{cases} d/\tau \text{ if } p \in \hat{V} \cap V_{\text{gt}} \wedge d < \tau \\ 1 \text{ otherwise,} \end{cases}
\end{aligned}
$$

TABLE I
ABLATION STUDIES ON POSERBPF ARCHITECTURES AND DEPTH UTILIZATION.

Input	Row	Fast Arch.	Render& Compare	Depth Embed.	Fusion	SDF	FPS	Overall Recall
RGB	1	no	-	-	-	-	11.5	41.7
	2	yes	-	-	-	-	19.6	28.6
RGB-D	3	no	yes	no	no	no	9.5	73.2
	4	yes	yes	no	no	no	18.3	76.5
	5	yes	no	normalized	late	no	14.9	61.8
	6	yes	yes	raw	late	no	13.3	40.7
	7	yes	yes	normalized	late	no	13.9	80.5
	8	yes	yes	normalized	early	no	12.3	74.6
	9	yes	yes	normalized	late	yes	6.4	**82.6**
Depth Only	10	-	yes	normalized	-	no	16.2	60.1

where $\hat{V}$, $\hat{D}$ and V_{gt}, D_{gt} represent the mask and depth map of the object computed by rendering the object according to estimated pose and ground truth pose respectively; p represents a pixel in the image; d is the depth error and can be computed with $d = |\hat{D}(p) - D_{\text{gt}}(p)|$; τ is a constant tolerance. We report the recall of correct 6D poses where $err_{\text{vsd}} < 0.3$ with $\tau = 2cm$ and visibility of more than 10% following [22].

For the YCB Video dataset, we use ADD and ADD-S [21, 75] as evaluation metrics. The two metrics can be computed as

$$
\begin{aligned}
\text{ADD} &= \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \|(\mathbf{R}\mathbf{x} + \mathbf{T}) - (\tilde{\mathbf{R}}\mathbf{x} + \tilde{\mathbf{T}})\| \\
\text{ADD-S} &= \frac{1}{m} \sum_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \|(\mathbf{R}\mathbf{x}_1 + \mathbf{T}) - (\tilde{\mathbf{R}}\mathbf{x}_2 + \tilde{\mathbf{T}})\|,
\end{aligned}
$$

where $\mathcal{M}$ denotes the set of 3D model points and m is the number of points. $(\mathbf{R}, \mathbf{T})$ and $(\tilde{\mathbf{R}}, \tilde{\mathbf{T}})$ are the ground truth pose and estimated pose, respectively.

D. Ablation Studies

We conduct ablation studies on the T-LESS dataset to justify our design choices. The particle filter is initialized with the detection outputs from RetinaNet [39] and has 100 particles. Table I shows the ablation studies results.

1) Original PoseRBPF vs Fast PoseRBPF: Rows 1 to 4 show the comparison between the original PoseRBPF and the Fast PoseRBPF. It is shown that the Fast PoseRBPF architecture significantly improve the tracking speed by 70% for RGB and 93% for RGB-D. For 6D object pose tracking accuracy with RGB inputs, the original PoseRBPF significantly outperforms the Fast PoseRBPF (rows 1 and 2). This performance gap is unsurprising since the spatial resolution decreases after convolution operations, and cropping on the feature space will result in less accurate translation estimation. However, the performance gap is bridged with depth measurements (rows 3 and 4) since translation is more accurately estimated by rendering the object according to the particles and comparing with the depth measurements. Therefore we only use the fast architecture for RGB-D tracking to retain the accuracy in RGB tracking.

2) Render and Compare: The benefits of utilizing depth measurements and the effectiveness of the proposed Render and Compare strategy can be shown by comparing rows 1

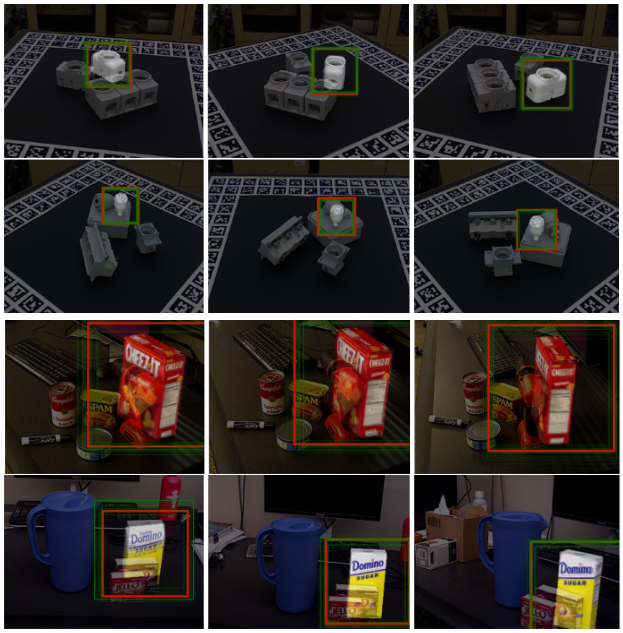


Fig. 7. Visualization of estimated poses on the T-LESS dataset (first two rows) and the YCB Video dataset (last two rows). Ground truth bounding boxes are red, green bounding boxes are particle RoIs, and the object models are superimposed on the images at the pose estimated by PoseRBPF.

and 3. By including depth inputs, the observation likelihood can be better estimated and more accurate 6D pose estimation can be achieved.

3) Depth Embeddings: We first investigate if encoding depth measurements helps improve the tracking accuracy. Rows 2 and 5 show that encoding depth measurements improves the tracking accuracy by more than 100%. Although the improvement of encoding depth measurements is less significant than the Render and Compare strategy (row 4), the two methods can be utilized together to further improve the tracking accuracy (row 7). The depth auto-encoder can be naively trained to encode the raw depth measurements which are metric. However, by comparing rows 6 and 7, the tracking performance is significantly deteriorated by including the auto-encoder for raw depth. This negative effect results from the large variance of the raw depth, and it justifies the normalization of the depth with Eq. (14) before feeding depth into the auto-encoder.

4) RGB and Depth Fusion: In addition to depth representations, we also investigate different ways to fuse RGB and depth measurements. A straight-forward approach is augmenting the auto-encoder for RGB inputs with an additional channel for depth. In the case, the depth measurements are fused with RGB inputs in the auto-encoder, which is referred as early fusion opposite to fusing the observation likelihoods in particle filter (late fusion) proposed in Sec. III-G. We can see the architecture with late fusion achieves better accuracy than the one with early fusion by comparing rows 7 and 8. This is because the particle filter balances the different modalities in the Bayesian estimation framework, while it is difficult to learn the relative importance from the synthetic data. By comparing rows 7 and 10, it can be shown that fusing RGB and depth inputs together can result in significantly more accurate pose tracking than using depth information alone.

5) Pose Refinement: It also can be seen from rows 7 and 9 that the pose estimation can be further refined by optimizing the signed distance function as described in Sec. III-G.

TABLE II
T-LESS RESULTS: OBJECT RECALL FOR $err_{\text{VSD}} < 0.3$ ON ALL PRIMESENSE TEST SCENES

	Without GT 2D BBs						With GT 2D BBs		
	RGB		RGB-D						
Object	[60]	Ours	[60]+ICP	[69]	Ours	Ours+SDF	[60]	[1]	Ours
1	8.87	**27.60**	22.32	43	76.20	**82.90**	12.33	75.50	**80.90**
2	13.22	**26.60**	29.49	47	80.10	**81.50**	11.23	**88.00**	85.80
3	12.47	**37.70**	38.26	69	81.90	**88.90**	13.11	84.00	**85.60**
4	6.56	**23.90**	23.07	63	85.40	**86.10**	12.71	**66.20**	62.00
5	34.80	**54.40**	76.10	69	90.40	**91.00**	66.70	73.00	**89.80**
6	20.24	**73.00**	67.64	67	**92.50**	92.00	52.30	65.10	**97.80**
7	16.21	**51.60**	73.88	77	**88.70**	85.50	36.58	22.20	**91.20**
8	19.74	**37.90**	67.02	79	**82.80**	81.70	22.05	42.00	**95.60**
9	**36.21**	41.60	78.24	**90**	88.60	89.10	46.49	47.40	**77.10**
10	11.55	**41.50**	77.65	68	76.00	**83.70**	14.31	13.30	**85.30**
11	6.31	**38.30**	35.89	69	**70.90**	**70.90**	15.01	66.00	**89.50**
12	8.15	**39.60**	49.30	82	83.50	**84.70**	31.34	49.40	**91.20**
13	4.91	**20.40**	42.50	**56**	46.60	47.80	13.60	66.60	**89.30**
14	4.61	**32.00**	30.53	47	**75.80**	72.60	45.32	34.40	**70.20**
15	26.71	**41.60**	**83.73**	52	79.10	83.30	50.00	58.20	**96.60**
16	21.73	**39.10**	67.42	81	86.00	**89.50**	36.09	80.60	**97.00**
17	**64.84**	40.00	86.17	83	**89.70**	79.00	81.11	54.30	**87.00**
18	14.30	**47.90**	84.34	80	81.40	**85.10**	52.62	63.00	**89.70**
19	22.46	**40.60**	50.54	55	71.40	**71.80**	50.75	28.50	**83.20**
20	5.27	**29.60**	14.75	47	**64.00**	63.60	37.75	12.40	**70.00**
21	17.93	**47.20**	40.31	63	86.60	**87.00**	50.89	33.30	**84.40**
22	18.63	**36.60**	35.23	70	72.10	**81.20**	47.60	9.60	**77.70**
23	18.63	**42.00**	42.52	85	82.90	**88.50**	35.18	17.00	**85.90**
24	4.23	**48.20**	59.54	70	87.00	**90.70**	11.24	68.20	**91.80**
25	18.76	**39.50**	70.89	48	68.50	**77.50**	37.12	35.00	**88.70**
26	12.62	**47.80**	66.20	55	93.40	**97.80**	28.33	43.30	**90.90**
27	21.13	**41.30**	**73.51**	60	61.10	66.80	21.86	40.00	**79.10**
28	23.07	**49.50**	61.20	69	87.20	**88.70**	42.58	54.20	**72.10**
29	26.65	**60.50**	73.04	65	96.00	**96.90**	57.01	38.50	**96.00**
30	29.58	**52.70**	**92.90**	84	90.80	91.50	70.42	**92.00**	77.00
Mean	18.35	**41.67**	57.14	66.3	80.52	**82.58**	36.79	50.74	**85.28**

E. Results on the T-LESS Dataset

Table II compares our approach with several other methods on the T-LESS dataset. We use 100 particles to track the objects. For pose estimation with RGB inputs, we compared our method with [60] which uses a similar auto-encoder. However the translation and orientation are estimated separately, and temporal consistency is not exploited in [60]. We perform evaluation with the detection output from RetinaNet [39] that is used in [60] as well. When multiple instances of the sample object are detected, we use the first instance in the list for initialization. The results show that the recall for the correct object poses doubles by estimating translation and orientation jointly in the particle filter and considering temporal consistency. With additional depth images, the recall can be further improved by around 98%. We use Fast PoseRBPF with encoding depth for comparison here. Without refinement, our approach outperforms [60] with ICP by 41%, [26] by 124%, and [69] by 21%. With refinement, our approach outperforms refining [60] with ICP by 45%, [26] by 130%, and [69] by 25%. For the experiments with ground truth bounding boxes, rotation is tracked using the particle filter and translation is inferred from the scale of the ground truth bounding box. This

TABLE III
RESULTS ON THE YCB VIDEO DATASET: COMPARISON WITH SINGLE-VIEW BASED 6D OBJECT POSE ESTIMATION METHODS

	RGB								RGB-D							
	PoseCNN [75]		PoseCNN +DeepIM [36]		Ours		Ours++		PoseCNN+ICP [75]		Dense Fusion [70]	PoseCNN +DeepIM [36]		Ours		Ours + SDF	
objects	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
master_chef_can	50.9	84.0	**71.2**	**93.1**	49.2	62.6	55.1	68.5	69.0	95.8	96.4	78.0	96.3	**91.9**	95.8	89.3	**96.7**
cracker_box	51.7	76.9	**83.6**	**91.0**	74.4	85.2	77.4	87.4	80.7	91.8	95.5	91.4	95.3	91.8	94.9	**96.0**	**97.1**
sugar_box	68.6	84.3	**94.1**	**96.2**	74.6	86.1	80.8	90.0	97.2	**98.2**	97.5	**97.6**	**98.2**	94.0	96.1	94.0	96.4
tomato_soup_can	66.0	80.9	**86.1**	**92.4**	75.0	84.5	76.7	85.2	81.6	94.5	94.6	90.3	94.8	**91.0**	94.6	87.2	**95.2**
mustard_bottle	79.9	90.2	**91.5**	**95.1**	75.7	87.3	81.7	90.3	97.0	98.4	97.2	97.1	98.0	93.2	96.3	**98.3**	**98.5**
tuna_fish_can	70.4	87.9	**87.7**	**96.1**	70.8	86.6	60.5	83.7	83.1	97.1	96.6	**92.2**	**98.0**	80.0	88.2	86.8	93.6
pudding_box	62.9	79.0	**82.7**	**90.7**	62.1	76.6	69.1	80.9	**96.6**	**97.9**	96.5	83.5	90.6	80.6	90.6	60.9	87.1
gelatin_box	75.2	87.1	**91.9**	**94.3**	88.3	92.4	88.1	92.9	**98.2**	**98.8**	98.1	98.0	98.5	96.4	97.7	**98.2**	98.6
potted_meat_can	59.6	78.5	**76.2**	**86.4**	43.7	55.2	48.3	58.4	83.8	**92.8**	91.3	**92.2**	90.3	77.8	83.0	76.4	83.5
banana	72.3	85.9	**81.2**	**91.3**	40.3	59.7	48.0	65.3	91.6	96.9	96.6	**94.9**	97.6	87.5	95.0	92.8	**97.7**
pitcher_base	52.5	76.8	**90.1**	**94.6**	74.9	87.5	76.3	88.2	96.7	97.8	97.1	97.4	97.9	89.8	95.0	**97.7**	**98.1**
bleach_cleanser	50.5	71.9	**81.2**	**90.3**	52.7	67.8	59.1	72.8	92.3	96.8	95.8	91.6	96.9	88.6	94.5	95.9	97.0
bowl	6.5	69.7	8.6	81.4	24.9	**87.6**	**34.7**	83.8	17.5	78.3	88.2	8.1	87.0	**46.8**	90.7	34.0	**93.0**
mug	57.7	78.0	**81.4**	**91.3**	64.4	82.1	78.3	90.6	81.4	95.1	97.1	**94.2**	**97.6**	91.4	96.7	86.9	96.7
power_drill	55.1	72.8	**85.5**	**92.3**	60.0	77.1	80.2	89.6	96.9	98.0	96.0	97.2	97.9	95.1	96.7	**97.8**	**98.2**
wood_block	31.8	65.8	**60.0**	**81.9**	7.7	18.4	28.1	46.2	79.2	90.5	89.7	**81.1**	91.5	33.4	92.2	37.8	**93.6**
scissors	35.8	56.2	**60.9**	**75.4**	28.5	43.7	48.0	66.2	78.4	92.2	95.2	**92.7**	**96.0**	89.0	94.1	72.7	85.5
large_marker	58.0	71.4	**75.6**	**86.2**	51.3	60.1	60.7	69.6	85.4	97.2	97.5	88.9	**98.2**	91.6	96.1	**89.2**	97.3
large_clamp	25.0	49.9	48.4	74.3	55.6	73.7	**58.1**	**76.2**	52.6	75.4	72.9	54.2	77.9	**90.9**	95.3	90.1	**95.5**
extra_large_clamp	15.8	47.0	31.0	**73.3**	**51.2**	71.4	50.8	72.0	28.7	65.3	69.8	36.5	77.8	77.0	89.8	**84.4**	**94.1**
foam_brick	40.4	87.8	35.9	81.9	77.7	88.9	**83.8**	**91.8**	48.3	97.1	92.5	48.2	97.6	95.3	97.3	**96.1**	**98.3**
ALL	53.7	75.9	**71.7**	**88.1**	60.4	75.4	66.3	79.6	79.3	93.0	93.1	80.7	94.0	86.8	93.7	**87.5**	**95.2**

TABLE IV
RESULTS ON THE YCB VIDEO DATASET: COMPARISON WITH 6D OBJECT POSE TRACKING METHODS

	RGB								RGB-D											
	Baseline Particle Filter (RGB)		DeepIM [36]		Ours		Ours++		Baseline Particle Filter (RGB-D)		RGF (Depth-based) [25]		Wüthrich's (Depth-based) [73]		se(3)-TrackNet [72]		Ours		Ours + SDF	
objects	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
master_chef_can	12.0	34.4	**89.0**	**93.8**	49.2	62.6	55.1	68.5	54.3	94.6	46.2	90.2	55.6	90.7	**93.8**	95.9	91.9	95.8	89.3	**96.7**
cracker_box	22.2	32.8	**88.5**	**93.0**	74.4	85.2	77.4	87.4	90.4	94.3	57.0	72.3	**96.4**	97.2	**96.4**	**97.1**	91.8	94.9	96.0	**97.1**
sugar_box	32.6	51.4	**94.3**	**96.3**	74.6	86.1	80.8	90.0	92.3	96.2	50.4	72.7	97.1	97.9	**97.6**	**98.1**	94.0	96.1	94.0	96.4
tomato_soup_can	46.4	64.3	**89.1**	**93.2**	75.0	84.5	76.7	85.2	73.3	88.4	72.4	91.6	64.7	89.6	94.8	97.1	90.1	94.6	87.2	95.2
mustard_bottle	49.7	72.3	**92.0**	**95.1**	75.7	87.3	81.7	90.3	93.1	96.7	87.7	98.2	97.1	98.0	95.7	97.4	93.2	96.3	**98.3**	**98.5**
tuna_fish_can	29.4	45.8	**92.0**	**96.4**	70.8	86.6	60.5	83.7	82.4	96.3	28.7	52.9	69.1	93.3	86.5	91.1	80.0	88.2	**86.8**	**93.6**
pudding_box	5.6	14.6	**80.1**	**88.3**	62.1	76.6	69.1	80.9	86.6	94.4	12.7	18.0	96.9	97.9	**97.9**	**98.4**	80.6	90.6	60.9	87.1
gelatin_box	55.2	64.9	**92.0**	**94.4**	88.3	92.4	88.1	92.9	95.4	97.7	49.1	70.7	97.5	98.4	97.7	98.5	96.4	97.7	**98.2**	**98.6**
potted_meat_can	26.3	40.4	**78.0**	**88.9**	43.7	55.2	48.3	58.4	67.5	73.0	44.1	45.6	83.7	86.7	74.5	82.4	77.8	83.0	**76.4**	**83.5**
banana	16.2	30.1	**81.0**	**90.5**	40.3	59.7	48.0	65.3	70.5	92.8	93.3	97.7	97.3	97.7	84.6	95.2	87.5	95.0	**92.8**	**97.7**
pitcher_base	6.1	40.7	**90.4**	**94.7**	74.9	87.5	76.3	88.2	90.5	95.0	97.9	98.2	97.3	97.7	96.7	97.4	89.8	95.0	**97.7**	**98.1**
bleach_cleanser	35.8	55.9	**81.7**	**90.5**	52.7	67.8	59.1	72.8	66.8	85.7	**95.9**	**97.3**	95.2	**97.3**	**95.9**	97.2	88.6	94.5	**95.9**	97.0
bowl	0.8	16.4	**38.8**	**90.6**	24.9	87.6	34.7	83.8	5.1	47.3	24.3	82.4	30.4	97.2	39.1	**95.6**	**46.8**	90.7	34.0	93.0
mug	1.1	6.2	**83.2**	**92.0**	64.4	82.1	78.3	90.6	60.2	93.8	60.0	71.2	83.2	93.4	**91.6**	**96.9**	91.4	96.7	86.9	96.7
power_drill	63.3	79.1	**85.4**	**92.3**	60.0	77.1	80.2	89.6	92.8	95.9	97.9	98.4	97.1	97.8	96.4	97.4	95.1	96.7	**97.8**	**98.2**
wood_block	1.0	2.5	**44.3**	**75.4**	7.7	18.4	28.1	46.2	0.9	1.9	**45.7**	62.5	95.5	96.9	33.9	**95.9**	33.4	92.2	37.8	93.6
scissors	14.7	30.3	**70.3**	**84.5**	28.5	43.7	48.0	66.2	88.4	95.2	20.9	38.6	4.2	16.2	**95.7**	**97.5**	89.0	94.1	72.7	85.5
large_marker	1.0	2.8	**80.4**	**91.2**	51.3	60.1	60.7	69.6	3.4	5.7	12.2	18.9	35.6	53.0	89.0	94.2	91.6	96.1	**89.2**	**97.3**
large_clamp	11.5	48.5	**73.9**	**84.4**	55.6	73.7	58.1	76.2	39.1	88.3	62.8	80.1	61.3	72.4	71.6	**96.9**	**90.9**	95.3	90.1	95.5
extra_large_clamp	10.1	16.7	**49.3**	**90.3**	51.2	71.4	50.8	72.0	12.9	28.1	67.5	69.7	**93.7**	**96.6**	64.6	95.8	77.7	89.8	84.4	94.1
foam_brick	18.8	44.7	**91.6**	**95.5**	77.7	88.9	83.8	91.8	45.7	97.4	70.0	86.6	96.8	98.1	40.7	94.7	95.3	97.3	**96.1**	**98.3**
ALL	25.5	42.3	**79.3**	**91.0**	60.4	75.4	66.3	79.6	65.5	81.9	59.2	74.3	78.0	90.2	**87.8**	**95.5**	86.7	93.7	87.5	95.2

TABLE V
EFFECT OF THE NUMBER OF PARTICLES ON TRACKING SPEED AND ACCURACY ON THE YCB VIDEO DATASET.

	RGB			RGB-D		
#particles	FPS	ADD	ADD-S	FPS	ADD	ADD-S
50	20.3	57.1	74.8	17.5	76.94	92.35
100	11.5	60.4	75.4	12.3	86.72	93.69
200	6.1	59.9	75.5	7.6	87.00	93.73

experiment highlights the viewpoint accuracy. In this setting, the particle filter significantly outperforms [60] and [1], which shows the importance of temporal tracking for object pose estimation. Fig. 7 shows the 6D pose estimation of PoseRBPF on several T-LESS images.

F. Results on the YCB Video Dataset

Tables III and IV shows the pose estimation results on the YCB Video dataset. In Table III, we compare with the state-of-the-art single-view based methods for 6D object pose estimation using RGB images [75, 36] and RGB-D images [75, 70, 36]. Fig. 7 illustrates some examples of the estimated 6D poses on the YCB Video dataset. We initialize PoseRBPF using PoseCNN detection [75] at the first frame or after the object is heavily occluded. On average, this happened only 1.03 times per sequence. In the experiments, 100 particles are used to track the 6D pose. For tracking with RGB inputs, our method handles symmetric objects such as 024_bowl, 061_foam_brick much better than the methods directly regressing the orientations [75]. By performing further image-based refinement upon the pose from PoseCNN, DeepIM [36] achieves more accurate 6D pose estimation than our method.

It has been shown in the context of robot localization that adding samples drawn according to the most recent observation can improve the localization performance [63]. Here, we applied such a technique by sampling 50% of the particles

around PoseCNN translation predictions and the other 50% with the motion model. Our results show that such a hybrid version (Ours++) improves the pose estimation accuracy of our approach thanks to the more accurate proposal distributions. One of the objects on which PoseRBPF performs poorly is the wooden block, which is caused by the difference in texture of the 3D model of the wooden block and the texture of the wooden block used in the real images. In addition, the physical dimensions of the wooden block are different between real images and the model contained in this dataset.

As shown in the results on the T-LESS dataset, depth measurements contain useful information to improve the pose estimation accuracy. This improvement is consistent on the YCB Video dataset. It is also worth to note that depth measurements also help bridge the gap between synthetic training data and real testing data which result in much better tracking performance on objects such as the wooden block. By comparing the depth of the rendered object with the depth measurements and encoding depth measurements, our filtering approach achieves better accuracy than [75] with Iterative Closest Points (ICP) and fusion approach [70]. By refining with SDF, the accuracy can be further improved, and our approach achieves the state-of-the-art performance.

In Table IV, we compare our method with 6D object pose tracking methods. We first compare PoseRBPF with a baseline using standard particle filters for 6D object pose tracking. In this baseline, each particle is represented with a translation hypothesis $\mathbf{T}_k^i$ and a rotation hypothesis $\mathbf{R}_k^i$, and we render the RGB image $\hat{\mathbf{Z}}_k^{Ci}$, the depth image $\hat{\mathbf{Z}}_k^{Di}$, and the mask for the object $\hat{\mathbf{M}}_k^i$ accordingly. Assuming the segmentation mask of the object $\mathbf{M}_k$ is given. We can compute the average photometric error $\mathbf{\Delta}_k^{Ci}$ and depth error $\mathbf{\Delta}_k^{Di}$ as:

$$\mathbf{\Delta}_k^{Ci} = \underset{p \in (\hat{\mathbf{M}}_k^i \cap \mathbf{M}_k)}{\text{avg}} (|\hat{\mathbf{Z}}_k^{Ci}(p) - \mathbf{Z}_k^C(p)|),$$

$$\mathbf{\Delta}_k^{Di} = \underset{p \in (\hat{\mathbf{M}}_k^i \cap \mathbf{M}_k)}{\text{avg}} (|\hat{\mathbf{Z}}_k^{Di}(p) - \mathbf{Z}_k^D(p)|).$$

We can also compute the ratio between the intersection and the union between the estimated segmentation mask $\hat{\mathbf{M}}_k^i$ and the measured segmentation mask $\mathbf{M}_k$ and denote it as m_k^i. The observation likelihood for RGB and RGB-D tracking can be computed as:

$$P_{\text{RGB}}(\mathbf{Z}_k|\mathbf{T}_k^i, \mathbf{R}_k^i) = \phi_c(\mathbf{\Delta}_k^{Ci})\phi_m(m_k^i),$$
$$P_{\text{RGBD}}(\mathbf{Z}_k|\mathbf{T}_k^i, \mathbf{R}_k^i) = \phi_c(\mathbf{\Delta}_k^{Ci})\phi_d(\mathbf{\Delta}_k^{Di})\phi_m(m_k^i),$$

where $\phi_c(\cdot)$ and $\phi_d(\cdot)$ are Gaussian functions centered at 0, and $\phi_m(\cdot)$ is a gaussion function centered at 1. In the experiments, we use the ground truth segmentation masks and 100 particles in the baselines. As shown in Table IV, PoseRBPF performs significantly better than the baselines using standard particle filters in both RGB and RGB-D scenarios. The comparisons demonstrate the superior sample efficiency of PoseRBPF over the standard particle filter in 6D pose tracking, and the robustness provided by the learned auto-encoder networks in handling lighting variance and noise in the input images.

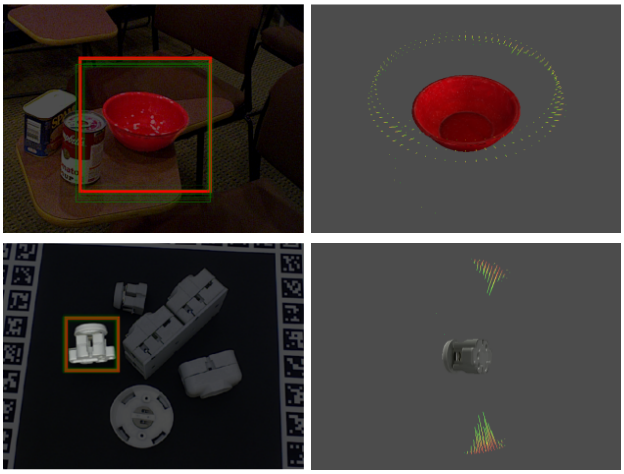


Fig. 8. Visualization of rotation distributions. For each image, the distribution over the rotation is visualized. The lines represent the probability for rotations that are higher than a threshold. The length of each line is proportional to the probability of that viewpoint. As can be seen, PoseRBPF naturally represents uncertainties due to various kinds of symmetries, including rotational symmetry of the bowl, mirror symmetry of the T-LESS object 12.

In addition, we compare our method with other 6D object pose tracking methods [36, 25, 73, 72]. Our method achieves comparable accuracy to the recent state-of-the-art method [72] for RGB-D tracking. For RGB-based tracking, our method is less accurate than refinement-based methods such as [36]. In comparison to the existing 6D object pose tracking systems, our method still provides an useful alternative since our method tracks the full 6D pose distribution and only requires 2D detection centers for initialization in contrast to initial 6D pose estimation required as in DeepIM [36].

Table V shows how the number of particles affect the tracking speed and accuracy. When the number of particles is small, with the increase in the number of particles, the accuracy improves because with more samples the variations in scale and translation of an object are covered much better. However, it also can be observed that the tracking performance saturates after 100 particles, and the performance for 100 particles is similar to that of 200 particles.

G. Analysis of Rotation Distribution

Unlike other 6D pose estimation methods that output a single estimate for the 3D rotation of an object, PoseRBPF tracks full distributions over object rotations. Fig. 8 shows example distributions of the rotation. There are two types of uncertainties in these distributions. The first source is the symmetry of the objects resulting in multiple poses with similar appearances. As expected, each cluster of the viewpoints corresponds to one of the similarity modes. The variance for each cluster corresponds to the true uncertainty of the pose. For example for the bowl, each ring of rotations corresponds to the uncertainty around the azimuth because the bowl is a rotationally symmetric object. Different rings show the uncertainty on the elevation.

To measure how well PoseRBPF can capture rotation uncertainty, we compared PoseRBPF estimates to those of PoseCNN assuming a Gaussian uncertainty with mean at the PoseCNN

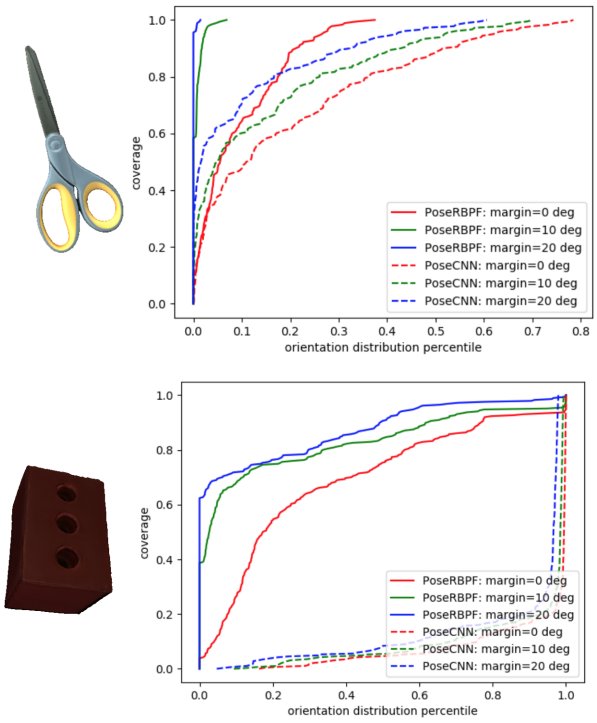
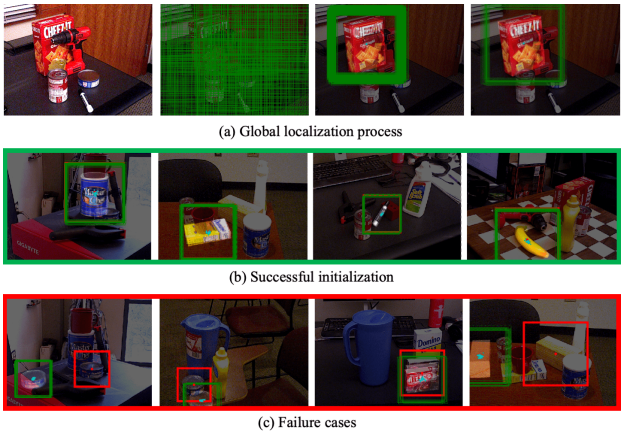


Fig. 9. Rotation Coverage Percentile comparison between PoseRBPF and PoseCNN for scissors and foam brick. Foam brick has 180° planar rotation and scissors is an asymmetric object.

estimate. Fig. 9 shows this comparison for the scissors and the foam brick in the YCB Video dataset. Here, the x-axis ranges over percentiles of the rotation distributions, and the y-axis shows how often the ground truth pose is within 0, 10, or 20 degrees of one of the rotations contained in the corresponding percentile. For instance, for the scissors, the red, solid line indicates that 80% of the time, the ground truth rotation is within 20 degrees of an rotation taken from the top 20% of the PoseRBPF distribution. If we take the top 20% rotations estimated by PoseCNN assuming a Gaussian uncertainty, this number drops to about 60%, as indicated by the lower dashed, red line. The importance of maintaining multi-modal uncertainties becomes even more prominent for the foam brick, which has a 180° symmetry. Here, PoseRBPF achieves high coverage, whereas PoseCNN fails to generate good rotation estimates even when moving further from the generated estimate.

H. Global Localization

In the previous discussion, we focused on object pose tracking, where PoseRBPF was initialized by 2D detection frameworks such as [39, 75]. However, there is no conceptual reason why PoseRBPF could not be deployed for global pose estimation, overcoming the need for a detection framework. Here, we propose a global sampling-based approach to initialize the system. We first sample translation by sampling 2D center of the object in the image p uniformly; the distance of the object is sampled uniformly in $\left[\mathbf{Z}_k^D(p) - 0.1, \mathbf{Z}_k^D(p) + 0.2\right]$. We evaluate 2400 samples, find the most likely one and sample 100 particles around it in a fine-grained manner: for object center in the image and distance, we sample with Gaussian functions with standard deviation 5 pixels and $0.015m$ respectively. When the maximum similarity score among all the particles is greater than a threshold (0.6), we start to tracking the object, otherwise we repeat the global localization process. We visualize the global localization process, successful initialization examples, and failure cases in Fig. 10. We evaluate the global localization strategy on YCB Video dataset. With the proposed global localization strategy, our tracking system can be initialized successfully in 49 out 55 testing sequences for all the objects in the YCB Video dataset, and result in ADD as 83.45 and ADD-S as 89.06. In our experiments, we observe initialization failures happen when the depth measurements on the object are missing (tuna fish can), or the object is heavily occluded (pudding box), or the texture of the object is significantly different to the model (wooden block).



(a) Global localization process

(b) Successful initialization

(c) Failure cases

Fig. 10. Visualization of global localization on YCB Video dataset. (a) illustrates the global localization process. We sample particles uniformly in the translation space. After evaluating all the particles, we perform fine-grained sampling around the particle with the max weight. The tracking process is triggered when the maximum similarity score among all the particles is above a threshold. (b) and (c) show the successful initialization and failures cases.

V. CONCLUSION AND DISCUSSION

We introduced PoseRBPF, a Rao-Blackwellized particle filter for tracking 6D object poses. Each particle samples 3D translation and estimates the distribution over 3D rotations conditioned on the image bonding box corresponding to the sampled translation. PoseRBPF compares each bounding box embedding to learned viewpoint embeddings so as to efficiently update distributions over time. We demonstrated that the tracked distributions capture both the uncertainties from the symmetry of objects and the uncertainty from object pose. Experiments on two benchmark datasets show that PoseRBPF effectively estimates the 6D pose of household objects and symmetric texture-less industrial objects.

PoseRBPF has several limitations that remain to be addressed. PoseRBPF can fail when the object is heavily occluded or the measurements are significantly different from the synthetic training data. Fig. 11 illustrates a tracking failure in YCB Video Dataset due to occlusion. We show the ratio of the object being occluded and the maximum similarity

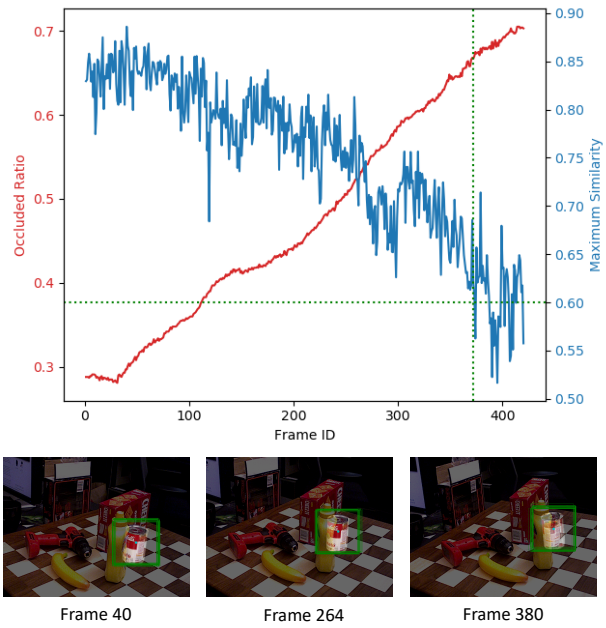


Fig. 11. Example of tracking failure due to occlusion. The plot shows that the maximum similarity (blue curve) decreases with increasing occlusion (red curve). In our implementation, the system determines tracking failure when the maximum similarity is lower than 0.6. Therefore our system can handle about 65% occlusion before failure in this example.

score of all the particles. It can be seen that the maximum similarity decreases with increasing occlusion. In this example, the system determines failure when the maximum similarity is below 0.6, which corresponds to 65% of the object being occluded. One potential approach to deal with occlusion is more accurately estimating the camera motion with visual odometry so that the particle filter depends less on the observation update. Fine-tuning the neural networks with real annotated data can be effective in bridging the domain gap between synthetic training data and real testing measurements, and it motivates our work [13] on annotating real data in a self-supervised fashion. Another limitation is that every object requires its own auto-encoder. Training an auto-encoder for multiple different objects is worth exploring. Moreover, improving orientation estimation by representing the rotation distribution with continuous functions is worth investigating.

References

[1] Phil Ammirato, Jonathan Tremblay, Ming-Yu Liu, Alexander Berg, and Dieter Fox. SymGAN: Orientation estimation without annotation for symmetric objects. In *IEEE Winter Conf. Comput. Vis. (WACV)*, 2020.

[2] Pedram Azad, David Münch, Tamim Asfour, and Rüdiger Dillmann. 6-DoF model-based tracking of arbitrarily shaped 3D objects. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011.

[3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *Eur. Conf. Comput. Vis. (ECCV)*, 2014.

[4] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron Dollar. The YCB object and model set: towards common benchmarks for manipulation research. In *IEEE Int. Conf. Adv. Robot. (ICAR)*, 2015.

[5] Zhe Cao, Yaser Sheikh, and Natasha Kholgade Banerjee. Real-time scalable 6DOF pose estimation for textureless objects. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016.

[6] Changhyun Choi and Henrik I Christensen. Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010.

[7] Changhyun Choi and Henrik I Christensen. 3D textureless object detection and tracking: An edge-based approach. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2012.

[8] Changhyun Choi and Henrik I Christensen. Robust 3D visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *Int. J. Robot. Res.*, 2012.

[9] Alvaro Collet, Manuel Martinez, and Siddhartha Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *Int. J. Robot. Res.*, 2011.

[10] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. A novel representation of parts for accurate 3D object detection and tracking in monocular images. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015.

[11] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 1997.

[12] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao-Blackwellized particle filter for 6D object pose tracking. In *Robotics: Science and Systems (RSS)*, 2019.

[13] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6D object pose estimation for robot manipulation. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020.

[14] Randal Douc and Olivier Cappé. Comparison of resampling schemes for particle filtering. In *Int. Symp. Image Signal Process. Anal.*, 2005.

[15] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-Blackwellised particle filtering for dynamic bayesian networks. In *Conf. Uncertainty in Artif. Intell.*, 2000.

[16] Mathieu Garon and Jean-François Lalonde. Deep 6-DOF tracking. *IEEE Trans. Vis. Comput. Graphics*, 2017.

[17] Ross Girshick. Fast R-CNN. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015.

[18] Chris Harris and Carl Stennett. RAPID: a video rate object tracker. In *Brit. Mech. Vis. Conf.*, 1990.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017.

[20] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Trans. Pattern Anal. Mach.*

Intell., 2012.

[21] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conf. Comput. Vis.*, 2012.

[22] Tomáš Hodaň. *SIXD Challenge 2017*. URL http://cmp.felk.cvut.cz/sixd/challenge_2017/.

[23] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. On evaluation of 6D object pose estimation. *Eur. Conf. Comput. Vis. Workshops (ECCVW)*, 2016.

[24] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. In *IEEE Winter Conf. Comput. Vis. (WACV)*, 2017.

[25] Jan Issac, Manuel Wüthrich, Cristina Garcia Cifuentes, Jeannette Bohg, Sebastian Trimpe, and Stefan Schaal. Depth-based object tracking using a robust gaussian filter. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016.

[26] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In *Eur. Conf. Comput. Vis. (ECCV)*, 2016.

[27] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making rgb-based 3d detection and 6D pose estimation great again. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017.

[28] Wadim Kehl, Federico Tombari, Slobodan Ilic, and Nassir Navab. Real-time 3D model tracking in color and depth on a single CPU core. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017.

[29] Zia Khan, Tucker Balch, and Frank Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2005.

[30] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*, 2014.

[31] Alexander Krull, Frank Michel, Eric Brachmann, Stefan Gumhold, Stephan Ihrke, and Carsten Rother. 6-DOF model based tracking via object coordinate regression. In *Asian Conf. Comput. Vis.*, 2014.

[32] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015.

[33] Abhijit Kundu, Yin Li, and James Rehg. 3D-RCNN: Instance-level 3D object reconstruction via render-and-compare. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018.

[34] Cody Kwok and Dieter Fox. Map-based multiple model tracking of a moving object. In *RoboCup 2004: Robot Soccer World Cup VIII*, 2004.

[35] Shile Li, Seongyong Koo, and Dongheui Lee. Real-time and model-free object tracking using particle filter with joint color-spatial descriptor. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2015.

[36] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: deep iterative matching for 6D pose estimation. In *Eur. Conf. Comput. Vis. (ECCV)*, 2018.

[37] Jörg Liebelt, Cordelia Schmid, and Klaus Schertler. Independent object class detection using 3D feature maps. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2008.

[38] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Eur. Conf. Comput. Vis. (ECCV)*, 2014.

[39] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

[40] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot Multibox Detector. In *Eur. Conf. Comput. Vis. (ECCV)*, 2016.

[41] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6D pose refinement in RGB. In *Eur. Conf. Comput. Vis. (ECCV)*, 2018.

[42] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. Label fusion: A pipeline for generating ground truth labels for real RGBD data of cluttered scenes. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018.

[43] F Landis Markley, Yang Cheng, John Lucas Crassidis, and Yaakov Oshman. Averaging quaternions. *J. Guidance Control Dyn.*, 2007.

[44] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI Conf. Artif. Intell.*, 2002.

[45] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. Object tracking with an adaptive color-based particle filter. In *Joint Pattern Recognit. Symp.*, 2002.

[46] Yuki Oka, Toshiyuki Kuroda, Tsuyoshi Migita, and Takeshi Shakunaga. Tracking 3D pose of rigid object by sparse template matching. In *Int. Conf. Image Graph.*, 2009.

[47] Karl Pauwels, Leonardo Rubio, Javier Diaz, and Eduardo Ros. Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013.

[48] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-DOF object pose from semantic keypoints. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

[49] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-wise voting network for 6DoF pose estimation. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019.

[50] Victor A Prisacariu and Ian D Reid. PWP3D: Real-time segmentation and tracking of 3D objects. *Int. J. Comput. Vis.*, 2012.

[51] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Int. Conf. Comput. Vis. Pattern*

Recognit. (CVPR), 2016.

[52] Carl Yuheng Ren, Victor Prisacariu, Olaf Kaehler, Ian Reid, and David Murray. 3D tracking of multiple objects with identical appearance using RGB-D input. In *Int. Conf. 3D Vis.*, 2014.

[53] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *Int. J. Comput. Vis.*, 2006.

[54] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Rao-Blackwellized particle filter for multiple target tracking. *Information Fusion*, 2007.

[55] Tanner Schmidt, Richard Newcombe, and Dieter Fox. DART: dense articulated real-time tracking. In *Robotics: Science and Systems (RSS)*, 2014.

[56] Caifeng Shan, Yucheng Wei, Tieniu Tan, and Frédéric Ojardias. Real time hand tracking by combining particle filtering and mean shift. In *IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2004.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[58] Chen Song, Jiaru Song, and Qixing Huang. HybridPose: 6D object pose estimation under hybrid representations. *arXiv preprint arXiv:2001.01869*, 2020.

[59] Rangaprasad Arun Srivatsan, Mengyun Xu, Nicolas Zevallos, and Howie Choset. Bingham distribution-based linear filter for online pose estimation. *Robotics: Science and Systems (RSS)*, 2017.

[60] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3D orientation learning for 6D object detection from RGB images. In *Eur. Conf. Comput. Vis. (ECCV)*, 2018.

[61] Martin Sundermeyer, Maximilian Durner, En Yen Puang, Zoltan-Csaba Marton, and Rudolph Triebel. Multi-path learning for object pose estimation across domains. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[62] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6D object pose prediction. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018.

[63] Sebastia Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[64] Meng Tian, Liang Pan, Marcelo H Ang Jr, and Gim Hee Lee. Robust 6D object pose estimation by learning RGB-D features. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020.

[65] Henning Tjaden, Ulrich Schwanecke, and Elmar Schömer. Real-time monocular segmentation and pose tracking of multiple objects. In *Eur. Conf. Comput. Vis. (ECCV)*, 2016.

[66] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017.

[67] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Int. Conf. Robot Learn. (CoRL)*, 2018.

[68] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3D camera tracking. In *Int. Symp. on Mixed and Augmented Reality*, 2004.

[69] Joel Vidal, Chyi-Yeu Lin, and Robert Martí. 6d pose estimation using an improved method based on point pair features. In *IEEE Int. Conf. Control Automat. Robot.*, 2018.

[70] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6D object pose estimation by iterative dense fusion. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019.

[71] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-pack: Category-level 6D pose tracker with anchor-based keypoints. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020.

[72] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E. Bekris. se(3)-tracknet: Data-driven 6D pose tracking by calibrating image residuals in synthetic domains. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2020.

[73] Manuel Wüthrich, Peter Pastor, Mrinal Kalakrishnan, Jeannette Bohg, and Stefan Schaal. Probabilistic object tracking using a range camera. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2013.

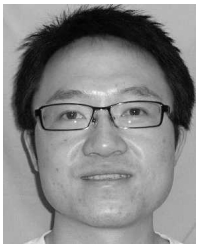
[74] Yu Xiang, Changkyu Song, Roozbeh Mottaghi, and Silvio Savarese. Monocular multiview object tracking with 3D aspect parts. In *Eur. Conf. Comput. Vis. (ECCV)*, 2014.

[75] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: a convolutional neural network for 6D object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018.

[76] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

Xinke Deng Xinke Deng received his Ph.D. in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 2020. He completed M.S. degree in aerospace engineering from the University of Illinois at Urbana-Champaign in 2015 and B.E. degree in aircraft design and engineering from Nanjing University of Aeronautics and Astronautics in 2013. His research interests focus on robot visual perception and state estimation.

Arsalan Mousavian Arsalan Mousavian is a Senior Research Scientist at NVIDIA. He received his Ph.D. in computer science from George Mason University in 2018. Prior to that, he received his M.Sc. degree from University of Tehran in 2013 and his B.Sc. degree from Iran University of Science and Technology in 2010. Arsalan's research interests are in 3D perception methods that help robots accomplish robot manipulation tasks in the real world.



Yu Xiang Yu Xiang is a Senior Research Scientist at NVIDIA. He received his Ph.D. in electrical engineering from the University of Michigan at Ann Arbor in 2016. He was a postdoctoral researcher at Stanford University and at the University of Washington from 2016 to 2017, and was a visiting student researcher in the artificial intelligence lab at Stanford University from 2013 to 2016. He received M.S. degree in computer science from Fudan University in 2010 and B.S. degree in computer science from Fudan University in 2007. His research interests focus on robotics and computer vision. His work studies how can a robot understand its 3D environment from sensing and accomplish tasks in the physical world.

Fei Xia Fei Xia is a Ph.D. Student at Stanford University. He is advised by Silvio Savarese and Leo Guibas. He obtained his bachelor's degree from Tsinghua University in 2016, and master's degree from Stanford University in 2019. His research tackles robot simulation, robot learning and simulation-to-real transfer of robot skills.

Timothy Bretl Timothy Bretl received the B.S. degree in engineering and the B.A. degree in mathematics from Swarthmore College, Swarthmore, PA, in 1999 and the M.S. and Ph.D. degrees both in aeronautics and astronautics from Stanford University, Stanford, CA, in 2000 and 2005, respectively. Subsequently, he was a Postdoctoral Fellow in the Department of Computer Science, also at Stanford University. Since 2006, he has been with the University of Illinois at Urbana-Champaign, where he is an Associate Professor of Aerospace Engineering and a Research Associate Professor in the Coordinated Science Laboratory. Dr. Bretl received the National Science Foundation Faculty Early Career Development Award in 2010. He has also received numerous teaching awards at Illinois, including the AIAA Student Chapter Teacher of the Year Award in 2015, both the William L. Everett Award for Teaching Excellence and the Rose Award for Teaching Excellence in 2016, and both the College of Engineering Teaching Excellence Award and the Campus Award for Excellence in Undergraduate Teaching in 2018.

Dieter Fox Dieter Fox received the Ph.D. degree from the University of Bonn, Germany. He is a professor in the Allen School of Computer Science & Engineering at the University of Washington, where he heads the UW Robotics and State Estimation Lab. He is also Senior Director of Robotics Research at NVIDIA. His research is in robotics and artificial intelligence, with a focus on state estimation and perception applied to problems such as mapping, object detection and tracking, manipulation, and activity recognition. He has published more than 200 technical papers and is co-author of the textbook "Probabilistic Robotics". He is a Fellow of the IEEE , AAAI, and ACM, and recipient of the IEEE RAS Pioneer Award. He was an editor of the IEEE Transactions on Robotics, program co-chair of the 2008 AAAI Conference on Artificial Intelligence, and program chair of the 2013 Robotics: Science and Systems conference.