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

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

***Abstract*—Tracking 6-D poses of objects from videos provides rich information to a robot in performing different tasks such as manipulation and navigation. In this article, we formulate the 6-D object pose tracking problem in the Rao–Blackwellized particle filtering framework, where the 3-D rotation and the 3-D translation of an object are decoupled. This factorization allows our approach, called PoseRBPF, to efficiently estimate the 3-D translation of an object along with the full distribution over the 3-D rotation. This is achieved by discretizing the rotation space in a fine-grained manner and training an autoencoder 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 6-D pose estimation benchmarks. We open-source our implementation at https://github.com/NVlabs/PoseRBPF.**

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

I. Introduction

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

Traditionally, the 6-D 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 3-D model of the object. Then, the 6-D object pose can be recovered using 2-D–3-D 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], [48], [62], [75], [76], So far, the focus of image-based 6-D 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 [7], [10], [31], [46]. In the context of point-cloud-based pose estimation, Kalman filtering has also been used to track 6-D 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 article, we introduce a particle-filter-based approach to estimate full posteriors over 6-D object poses. Our approach, called PoseRBPF, factorizes the posterior into the 3-D translation and the 3-D 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 3-D rotation is discretized at 5° resolution, resulting in a distribution over $72 \times 37 \times 72 = 191\,808$ bins for each particle (elevation ranges only from $-90°$ to $90°$). To achieve real-time performance, we precompute a codebook over embeddings for all discretized rotations, where embeddings come from an autoencoder 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 3-D translation to determine the center and size of the object bounding box in the image, then computes the embedding for that bounding box

Manuscript received May 27, 2020; accepted December 28, 2020. Date of publication February 25, 2021; date of current version October 1, 2021. This article was recommended for publication by Associate Editor L. Carlone and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. Part of the work was done during Xinke Deng and Fei Xia's internships at NVIDIA. *(Corresponding author: Xinke Deng.)*

Xinke Deng was with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: xdeng12@illinois.edu).

Arsalan Mousavian and Yu Xiang are with the NVIDIA, Seattle, WA 98119 USA (e-mail: arsalan.mousavian@gmail.com; yux@nvidia.com).

Fei Xia is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: xf1280@gmail.com).

Timothy Bretl is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: tbretl@illinois.edu).

Dieter Fox is with the NVIDIA, Seattle, WA 98119 USA, and also with the Allen School of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: fox@cs.washington.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2021.3056043.

Digital Object Identifier 10.1109/TRO.2021.3056043

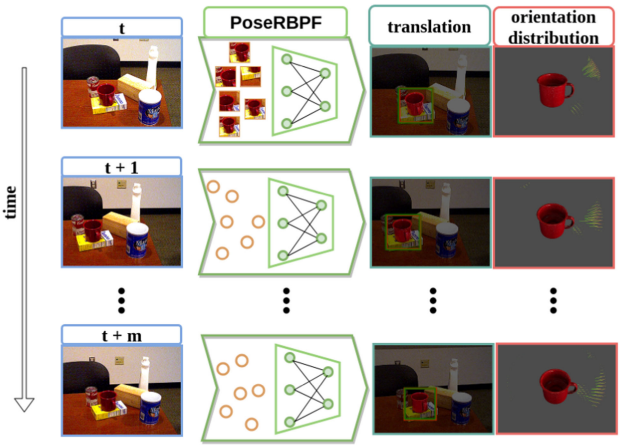


Fig. 1. Overview of our PoseRBPF framework for 6-D object pose tracking. Our method leverages a Rao–Blackwellized particle filter and an autoencoder network to estimate the 3-D translation and a full distribution of the 3-D rotation of a target object from a video sequence.

using the autoencoder, and finally updates the rotation distribution by comparing the embedding vector with the precomputed 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 6-D 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 6-D pose estimation.

Our work makes the following main contributions.

1) We introduce a novel and versatile 6-D object pose estimation framework that combines a Rao–Blackwellized particle filtering with a learned autoencoder network in an efficient and principled way.
2) Our framework is able to track full distributions over 6-D 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 article.

1) We propose an efficient modification inspired by [17], where we apply region of interest (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.
2) Apart from encoding RGB measurements using autoencoders, we propose to encode depth measurements using separate autoencoders and show that the tracking performance can be significantly improved.
3) 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.
4) 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 this article is organized as follows. After discussing the related work, we present our Rao–Blackwellized particle filtering framework for 6-D object pose tracking, followed by experimental evaluations and a conclusion.

II. Related Work

A. Six-Dimensional Object Pose Estimation

Our work is closely related to recent advances in 6-D object pose estimation using deep neural networks. The current trend is to augment state-of-the-art 2-D object detection networks with the ability to estimate 6-D object pose. For instance, Kehl *et al.* [27] extend the SSD detection network [40] to 6-D pose estimation by adding viewpoint classification to the network. Tekin *et al.* [62] utilize the YOLO architecture [51] to detect 3-D bounding box corners of objects in the images and then recover the 6-D pose by solving the perspective-n-point problem. Detecting 3-D bounding box corners or object key points for 6-D object pose estimation is also explored in [42], [49], [58], and [67]. PoseCNN [75] designs an end-to-end network for 6-D object pose estimation based on the VGG architecture [57]. Although these methods significantly improve the 6-D pose estimation accuracy over the traditional methods [3], [21], [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, Tian *et al.* [64] propose to uniformly sample rotation anchors and estimate deviations of the anchors to the target. In addition, Sundermeyer *et al.* [60], [61] introduce an implicit way of representing 3-D rotations by training an autoencoder for image reconstruction, which does not need to predefine the symmetry axes for symmetric objects. We leverage this implicit 3-D rotation representation in our work and show how to combine it with particle filtering for 6-D object pose tracking.

B. Six-Dimensional Object Pose Tracking

Another set of related work is on object tracking from videos. Early works [6], [18], [68] 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 6-D 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], [16], [28], [52], [55], [71], [72]. 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 6-D pose tracking with RGB data includes [41], [50], [65], and [66]. In [50], the pose of object is updated by optimizing the projected contour from the 3-D

model. The approach is improved in [65] with a new optimization scheme and through GPU parallelization. Tjaden *et al.* [66] improve pose tracking with a temporally consistent local color histogram. Meanwhile, deep neural networks are explored for 6-D object pose tracking. Very recently, deep neural networks have been used to predict the pose difference between consecutive frames and track the 6-D object pose accordingly [36], [41], [72]. These methods significantly improve the robustness and accuracy of tracking compared to methods that use handcrafted features [50], [65], [66]. However, object symmetries are either ignored or manually specified in these works, and 6-D 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 2-D 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.

C. Particle Filtering

The particle filtering framework has been widely applied to different tracking applications in the literature [29], [45], [54], [56], thanks to its flexibility in incorporating different observation models and motion priors. Meanwhile, it offers a rigorous probabilistic formulation to estimate uncertainty in the tracking results. Different approaches have also been proposed to track the poses of objects using particle filters [2], [8], [35], [47], [74]. However, in order to achieve good tracking performance, a particle filter requires a strong observation model. In addition, the tracking frame rate is limited by the particle sampling and evaluation efficiency. In this article, we factorize the 6-D object pose tracking problem and deploy Rao–Blackwellized particle filters [15], which have been shown to scale to complex estimation problems such as simultaneous localization and mapping problem [44], [63] and multimodel 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 6-D pose of an object at 20 frames/s.

III. Six-Dimensional Object Pose Tracking With PoseRBPF

In this section, we first state the problem of 6-D 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 6-D 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 2-D center (u, v) of the object in the first image is provided by an object detector, such as in [17] and [51] for pose tracking initialization, and the 3-D computer-aided design (CAD) model of the object is known. The rigid body transformation consists of a 3-D rotation $\mathbf{R}_k$ and a 3-D translation $\mathbf{T}_k$ of the object at time k. In this article, instead of providing a single estimation $\{\mathbf{R}_k, \mathbf{T}_k\}$, our primary goal is to estimate the posterior distribution of the 6-D 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 6-D 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 Section III-E. Each particle determines a unique RoI according to its translation, and the RoI is fed into an autoencoder network to compute a feature embedding. The observation likelihood is computed by matching the embedding with the embeddings in a precomputed codebook, which is detailed in Section 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 6-D 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 6-D pose estimation problem into 3-D rotation estimation and 3-D translation estimation. This idea is based on the observation that the 3-D 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 3-D 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 rotation conditioned on the translation and the images, and w_k^i is the importance weight. To achieve accurate pose estimation, the 3-D object rotation consisting of azimuth, elevation, and in-plane rotation is discretized into bins of size 5°, resulting in a distribution over $72 \times 37 \times 72 = 191\,808$ bins for each particle (elevation

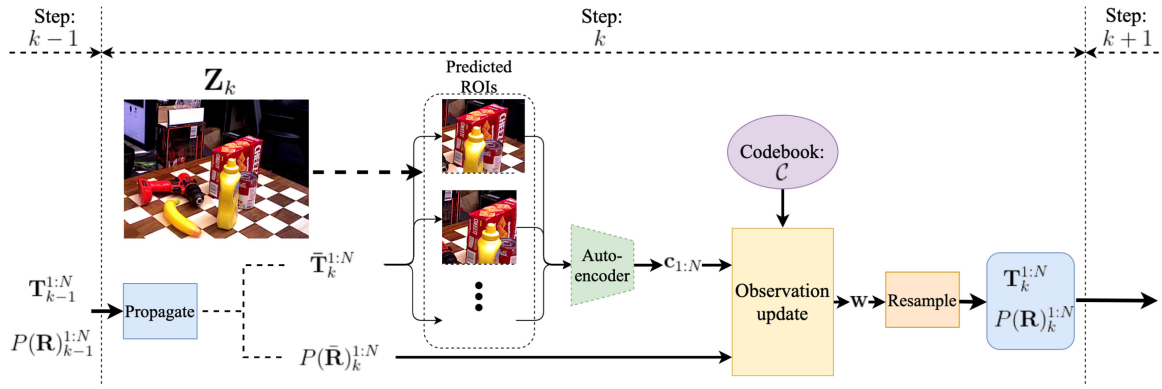


Fig. 2. Architecture of PoseRBPF. For each particle, the *rotation distribution* is estimated conditioned on translation estimation, while the translation estimation is evaluated with the corresponding RoIs.

ranges only from $-90°$ to $90°$). 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 6-D 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 3-D rotation $\mathbf{R}_k$ and the 3-D translation $\mathbf{T}_k$.

Intuitively, a 6-D object pose estimation method, such as in [27], [62], and [75], can be employed to estimate the observation likelihoods. However, these methods only provide a single estimation of the 6-D pose instead of estimating a probability distribution, i.e., there is no uncertainty in their estimation. In addition, 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 3-D model of the object. Therefore, inspired by [60], we apply an autoencoder to transform the observation $\mathbf{Z}_k$ into the same domain as the synthetic rendering of the object.

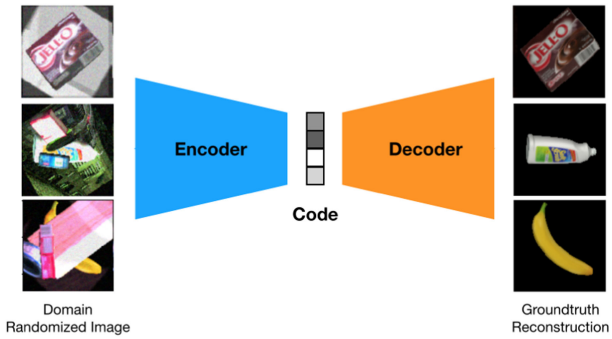


Fig. 3. Inputs and outputs of the autoencoder. Images with different lighting, background, and occlusion are feed into the network to reconstruct synthetic images of the objects from the same 6-D poses. The encoder generates a feature embedding (code) of the input image.

Then, we can compare image features in the synthetic domain to measure the likelihoods of 6-D poses efficiently.

1) Autoencoder: An autoencoder 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 autoencoder 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 autoencoder during training. In addition, the autoencoder learns a feature embedding $f(\mathbf{Z})$ of the input image.

Instead of training the autoencoder to reconstruct images with arbitrary 6-D poses, which makes the training challenging, we fix the 3-D 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 3-D model to the camera, which makes sure renderings from all the rotations well fitted to the training image size (128×128 in our experiments). The 3-D rotation $\mathbf{R}$ is uniformly sampled during training. After training, for each discretized 3-D 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

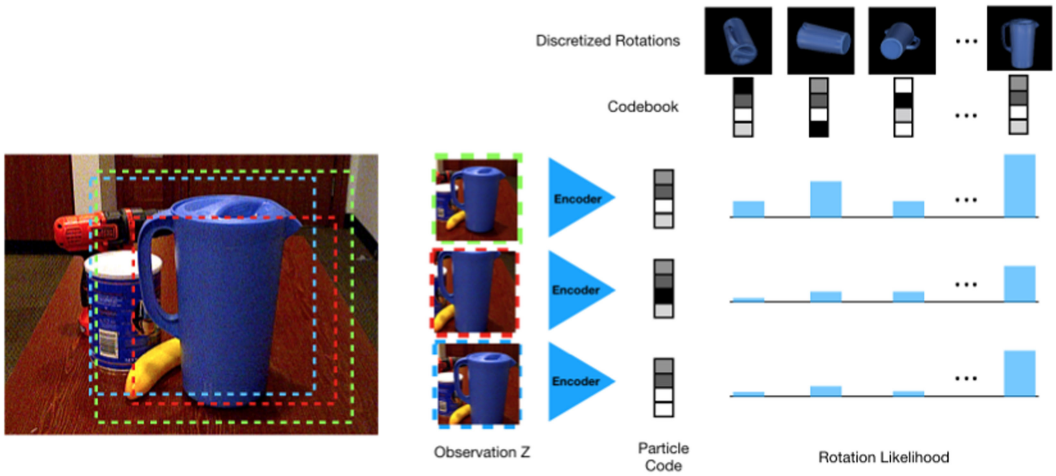


Fig. 4. Computation for the conditional rotation likelihood by codebook matching. (Left) Each particle crops the image based on its translation hypothesis. The RoI for each particle is resized, and the corresponding code is computed using the encoder. (Right) Rotation distribution $P(\mathbf{R}|\mathbf{Z},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

the feature embeddings of the discretized 3-D 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 3-D translation hypothesis $\mathbf{T}_k$, we can crop an 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 3-D 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 autoencoder, respectively. Note that each RoI is a square region in our case, which makes the RoI independent of 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:

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

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

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

Fig. 5. Visualization of reconstruction of the RoIs from autoencoder. Left is the ground truth RoI. The other two columns show the reconstruction with shifting and scale change. As 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 autoencoder a suitable choice for computing the observation likelihood.

Since the autoencoder 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 (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 (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 3-D 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 $\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$ denotes the multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$, and α is a hyperparameter 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 3-D Gaussian kernel.

F. 6-D Object Pose Tracking Framework

The tracking process can be initialized from any 2-D object detector that outputs a 2-D bounding box of the target object. Given the first frame $\mathbf{Z}_1$, we backproject the center of the 2-D bounding box to compute the (x, y) components of the 3-D 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 autoencoder to compute the corresponding code. For each particle, the rotation distribution is updated with

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

where $P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_k)$ is the rotation distribution defined in (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 (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 6-D 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 unimodal 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 multimodal 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].

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

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 predefined threshold, we determine it as a tracking failure. Algorithm 1 summarizes our Rao–Blackwellized particle filter for 6-D 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 also be encoded with an autoencoder, 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 SDF of the 3-D 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

$$\Delta_k^i(\hat{\mathbf{Z}}_k^{Di}, \mathbf{Z}_k^D, \hat{V}_k^i, \tau) = \underset{p \in \hat{V}_k^i}{\text{avg}} \left(\min \left(\frac{|\mathbf{Z}_k^D(p) - \hat{\mathbf{Z}}_k^{Di}(p)|}{\tau}, 1 \right) \right) \tag{13}$$

where τ is a predefined 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 a separate autoencoder network. For each particle $\{\mathbf{T}_k^i, P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_{1:k})\}$, we first normalize the depth measurements with

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

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, (14) normalizes depth measurements to $[0, 1]$ according to the particles. Our experiments in Section IV show that the normalization significantly improves the tracking accuracy compared to encoding the original depth values. We train a separate autoencoder 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 (12).

3) Pose Refinement With SDFs: The estimated object pose from PoseRBPF can be further refined by matching the 3-D points from the depth measurements against the 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 Section III-G1. The point cloud of the object $\mathbf{P}_{\text{obj}}$ can be computed by back-projecting the pixels in $\bar{\mathbf{V}}$

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

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

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

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

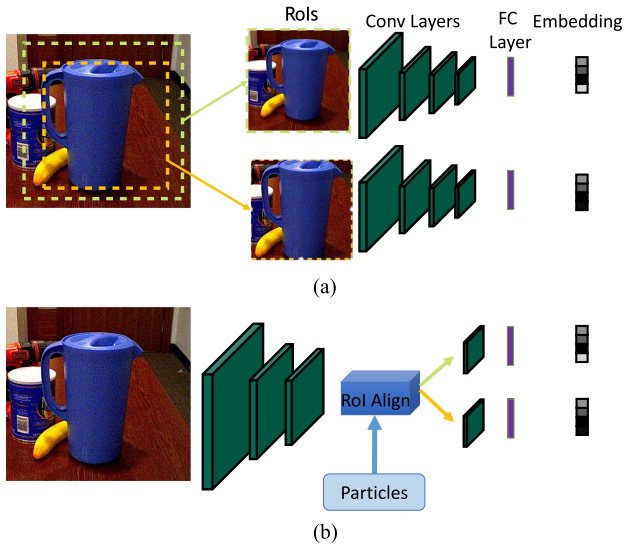


Fig. 6. Comparison between the autoencoders in the original PoseRBPF and Fast PoseRBPF. (a) In the original PoseRBPF, each particle crops its RoI directly on the input image and pass through the encoder individually. (b) In Fast PoseRBPF, each particle crops on the shared feature maps, so that the early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

where $\mathbf{p}_i$ is a 3-D point in the point cloud $\mathbf{P}_{\text{obj}}$, and $\mathbf{SDF}_{\text{obj}}(\mathbf{p}_i, \mathbf{T}, \mathbf{R})$ 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 2-D image $(\bar{u}, \bar{v})^T$ can be computed with (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 autoencoder, respectively, and β is a scaling factor to ensure that 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 2-D 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 autoencoder 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-D embeddings. We use a similar architecture but reduce the number of channels in the convolution layers by half for the depth autoencoder to avoid overfitting. For Fast PoseRBPF, the scaling factor β is set to 2. Therefore, 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 are 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 a resolution of 128×128. The depth data are first normalized according to (14) with $d = 0.4$. The background data for normalized depth are generated by averaging the RGB channels at random crops in the 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 hue, saturation, value (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 autoencoders are trained for each object separately for 150 000 iterations with batch size of 64 using the Adam optimizer with the learning rate of 0.0002. The autoencoder 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 nontextured objects.

3) Testing: During test time, the standard deviation used to compute observation likelihoods in (6) is 0.05. The codebook for each object is precomputed 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 Section III-G, the margin m is chosen as 2 cm and the threshold τ is set to 3 cm 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 6-D object pose estimation: the T-LESS dataset [24] and the YCB Video dataset [75]. The T-LESS dataset contains RGB-D sequences of 30 nontextured 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 6-D 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

$$\text{err}_{\text{vsd}} = \underset{p \in \hat{V} \cup V_{\text{gt}}}{\text{avg}} \; c(p, \hat{D}, D_{\text{gt}}, \tau)$$

$$c(p, \hat{D}, D_{\text{gt}}, \tau) = \begin{cases} d/\tau, \text{ if } p \in \hat{V} \cap V_{\text{gt}} \wedge d < \tau \\ 1, \text{ otherwise} \end{cases}$$

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 $d = |\hat{D}(p) - D_{\text{gt}}(p)|$; and τ is a constant tolerance. We report the recall of correct 6-D poses, where $\text{err}_{\text{vsd}} < 0.3$ with $\tau = 2$ cm 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

$$\text{ADD} = \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \|(\mathbf{R}\mathbf{x} + \mathbf{T}) - (\tilde{\mathbf{R}}\mathbf{x} + \tilde{\mathbf{T}})\|$$

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

where $\mathcal{M}$ denotes the set of 3-D 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.

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

TABLE II
T-LESS RESULTS: OBJECT RECALL FOR $\mathrm{err}_{\mathrm{VSD}} < 0.3$ 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**

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 results of ablation studies.

1) Original PoseRBPF Versus Fast PoseRBPF: Rows 1–4 show the comparison between the original PoseRBPF and the Fast PoseRBPF. It is shown that the Fast PoseRBPF architecture significantly improves the tracking speed by 70% for RGB and 93% for RGB-D. For 6-D 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 and 3. By including depth inputs, the observation likelihood can be better estimated and more accurate 6-D 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 autoencoder 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 autoencoder for raw depth. This negative effect results from the large variance of the raw depth, and it justifies the normalization of the depth with (14) before feeding depth into the autoencoder.

4) RGB and Depth Fusion: In addition to depth representations, we also investigate different ways to fuse RGB and depth measurements. A straightforward approach is augmenting the autoencoder for RGB inputs with an additional channel for depth. In the case, the depth measurements are fused with RGB inputs in the autoencoder, which is referred as early fusion opposite to fusing the observation likelihoods in particle filter (late fusion) proposed in Section III-G. We can see that 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 can also be seen from rows 7 and 9 that the pose estimation can be further refined by optimizing the SDF, as described in Section III-G.

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 autoencoder. 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,

TABLE III
RESULTS ON THE YCB VIDEO DATASET: COMPARISON WITH SINGLE-VIEW BASED 6-D 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**

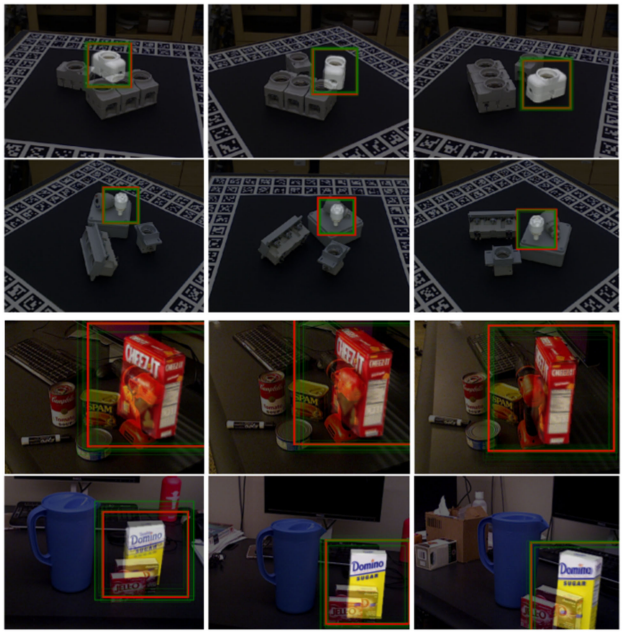


Fig. 7. Visualization of estimated poses on the T-LESS dataset (first two rows) and the YCB Video dataset (last two rows). Ground truth bounding boxes are red, green bounding boxes are particle RoIs, and the object models are superimposed on the images at the pose estimated by PoseRBPF.

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 iterative closest points (ICPs) 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 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 6-D pose estimation of PoseRBPF on several T-LESS images.

F. Results on the YCB Video Dataset

Tables III and IV show 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 6-D object pose estimation using RGB images [36], [75] and RGB-D images [36], [70], [75]. Fig. 7 illustrates some examples of the estimated 6-D 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 6-D 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 6-D 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 3-D 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

TABLE IV
RESULTS ON THE YCB VIDEO DATASET: COMPARISON WITH 6-D 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

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 ICPs and fusion approach [70]. By refining with the 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 6-D object pose tracking methods. We first compare PoseRBPF with a baseline using standard particle filters for 6-D 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. Assume that the segmentation mask of the object $\mathbf{M}_k$ is given. We can compute the average photometric error $\boldsymbol{\Delta}_k^{Ci}$ and depth error $\boldsymbol{\Delta}_k^{Di}$ as

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

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

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

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

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

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

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 6-D pose tracking and the robustness provided by the learned autoencoder networks in handling lighting variance and noise in the input images.

In addition, we compare our method with other 6-D object pose tracking methods [25], [36], [72], [73]. 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 6-D object pose tracking systems, our method still provides an useful alternative since our method tracks the full 6-D pose distribution and only requires 2-D detection centers for initialization in contrast to initial 6-D pose estimation required as in DeepIM [36].

Table V shows how the number of particles affects 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 can also be observed that the tracking performance saturates after 100 particles, and the performance for 100 particles is similar to that of 200 particles.

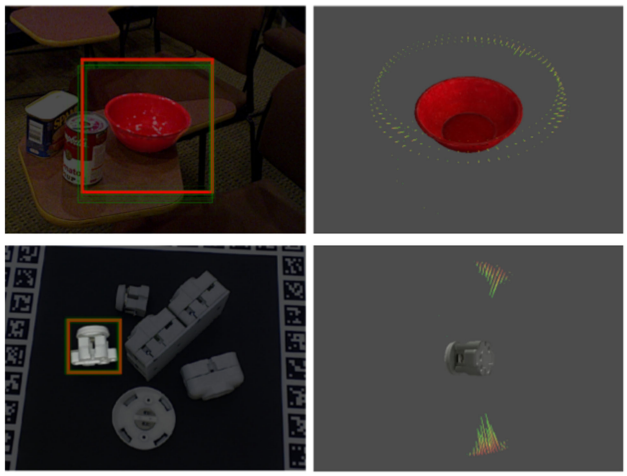


Fig. 8. Visualization of rotation distributions. For each image, the distribution over the rotation is visualized. The lines represent the probability for rotations that are higher than a threshold. The length of each line is proportional to the probability of that viewpoint. As can be seen, PoseRBPF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

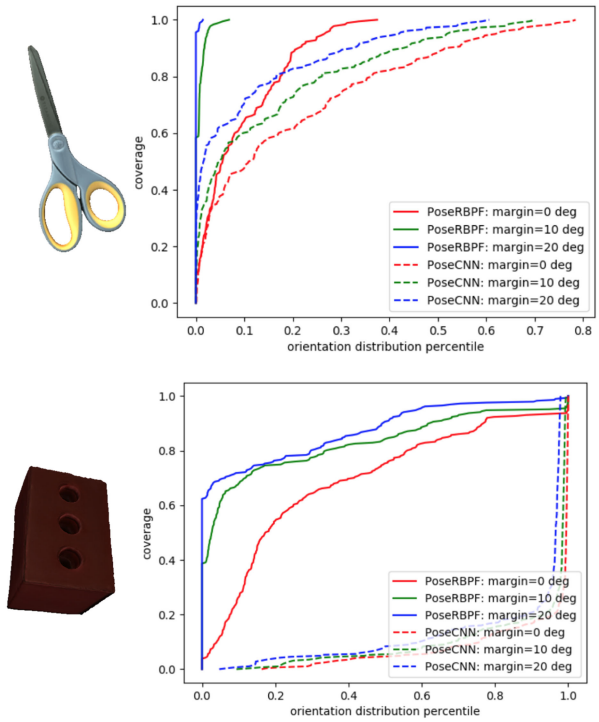


Fig. 9. Rotation coverage percentile comparison between PoseRBPF and PoseCNN for scissors and foam brick. Foam brick has 180° planar rotation and scissors is an asymmetric object.

G. Analysis of Rotation Distribution

Unlike other 6-D pose estimation methods that output a single estimate for the 3-D 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 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° 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° 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 multimodal 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 2-D detection frameworks such as in [39] and [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 2-D center of the object in the image p uniformly; the distance of the object is sampled uniformly in $[\mathbf{Z}_k^D(p) - 0.1, \mathbf{Z}_k^D(p) + 0.2]$. 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 0.015 m, 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 the 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 that 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).

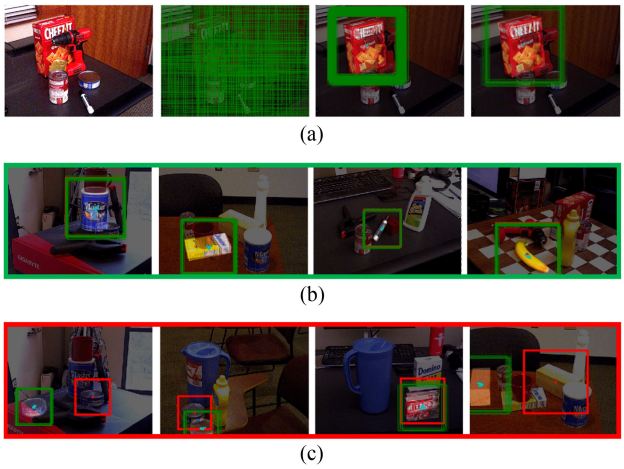


Fig. 10. Visualization of global localization on the YCB Video dataset. (a) Global localization process. We sample particles uniformly in the translation space. After evaluating all the particles, we perform fine-grained sampling around the particle with the max weight. The tracking process is triggered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

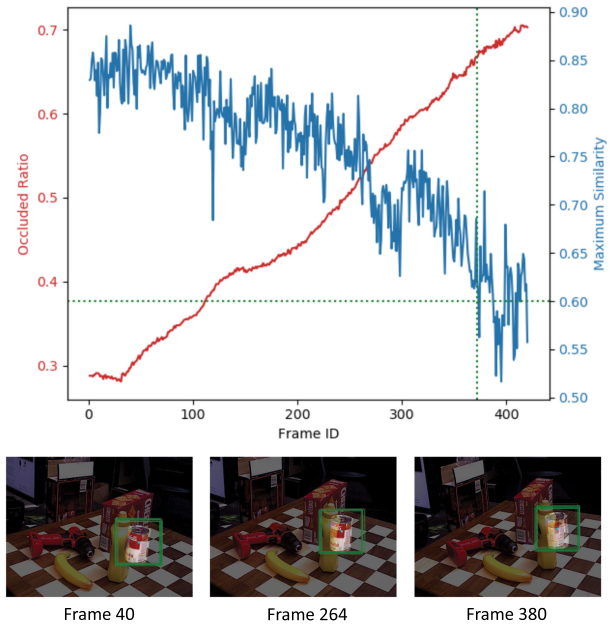


Fig. 11. Example of tracking failure due to occlusion. The plot shows that the maximum similarity (blue curve) decreases with increasing occlusion (red curve). In our implementation, the system determines tracking failure when the maximum similarity is lower than 0.6. Therefore, our system can handle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

V. Conclusion

In this article, we introduced PoseRBPF, a Rao–Blackwellized particle filter for tracking 6-D object poses. Each particle sampled 3-D translation and estimated the distribution over 3-D rotations conditioned on the image bonding box corresponding to the sampled translation. PoseRBPF compared 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 showed that PoseRBPF effectively estimates the 6-D pose of household objects and symmetric textureless 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 the YCB Video Dataset due to occlusion. We showed the ratio of the object being occluded and the maximum similarity 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 autoencoder. Training an autoencoder for multiple different objects is worth exploring. Moreover, improving orientation estimation by representing the rotation distribution with continuous functions is worth investigating.

References

[1] P. Ammirato, J. Tremblay, M.-Y. Liu, A. Berg, and D. Fox, "SymGAN: Orientation estimation without annotation for symmetric objects," in *Proc. IEEE Winter Conf. Comput. Vis.*, 2020, pp. 1657–1666.

[2] P. Azad, D. Münch, T. Asfour, and R. Dillmann, "6-DoF model-based tracking of arbitrarily shaped 3D objects," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 5204–5209.

[3] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 536–551.

[4] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *Proc. IEEE Int. Conf. Adv. Robot.*, 2015, pp. 510–517.

[5] Z. Cao, Y. Sheikh, and N. K. Banerjee, "Real-time scalable 6DOF pose estimation for textureless objects," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 2441–2448.

[6] C. Choi and H. I. Christensen, "Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 4048–4055.

[7] C. Choi and H. I. Christensen, "3D textureless object detection and tracking: An edge-based approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2012, pp. 3877–3884.

[8] C. Choi and H. 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.*, vol. 31, pp. 498–519, 2012.

[9] A. Collet, M. Martinez, and S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *Int. J. Robot. Res.*, vol. 30, no. 10, pp. 1284–1306, 2011.

[10] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "A novel representation of parts for accurate 3D object detection and tracking in monocular images," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4391–4399.

[11] P. Del Moral, "Nonlinear filtering: Interacting particle resolution," Compt. Rendus De L'académie Des Sci.-Ser. I-Math., vol. 325, no. 6, pp. 653–658, 1997.

[12] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "PoseRBPF: A Rao-Blackwellized particle filter for 6D object pose tracking," 2019.

[13] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6D object pose estimation for robot manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 3665–3671.

[14] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," in *Proc. Int. Symp. Image Signal Process. Anal.*, 2005, pp. 64–69.

[15] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, "Rao-blackwellised particle filtering for dynamic Bayesian networks," in *Proc. Conf. Uncertainty Artif. Intell.*, 2000, pp. 176–183.

[16] M. Garon and J.-F. Lalonde, "Deep 6-DOF tracking," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 11, pp. 2410–2418, Nov. 2017.

[17] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[18] C. Harris and C. Stennett, "RAPID: A video rate object tracker," in *Proc. Brit. Mech. Vis. Conf.*, 1990, pp. 15-1–15-6.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[20] S. Hinterstoisser *et al.*, "Gradient response maps for real-time detection of textureless objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 876–888, May 2012.

[21] S. Hinterstoisser *et al.*, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Proc. Asian Conf. Comput. Vis.*, 2012, pp. 548–562.

[22] T. Hodaň, SIXD Challenge2017, 2017. [Online]. Available: https://cmp.felk.cvut.cz/sixd/challenge_2017/

[23] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6D object pose estimation," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2016, pp. 606–619.

[24] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," in *Proc. IEEE Winter Conf. Comput. Vis.*, 2017, pp. 880–888.

[25] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, and S. Schaal, "Depth-based object tracking using a robust Gaussian filter," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 608–615.

[26] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 205–220.

[27] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1521–1529.

[28] W. Kehl, F. Tombari, S. Ilic, and N. Navab, "Real-time 3D model tracking in color and depth on a single CPU core," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 745–753.

[29] Z. Khan, T. Balch, and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1805–1819, Nov. 2005.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[31] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother, "6-DOF model based tracking via object coordinate regression," in *Proc. Asian Conf. Comput. Vis.*, 2014, pp. 384–399.

[32] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 954–962.

[33] A. Kundu, Y. Li, and J. Rehg, "3D-RCNN: Instance-level 3D object reconstruction via render-and-compare," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3559–3568.

[34] C. Kwok and D. Fox, "Map-based multiple model tracking of a moving object," in *RoboCup 2004: Robot Soccer World Cup VIII*. New York, NY, USA: Springer, 2004, pp. 18–33.

[35] S. Li, S. Koo, and D. Lee, "Real-time and model-free object tracking using particle filter with joint color-spatial descriptor," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2015, pp. 6079–6085.

[36] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep iterative matching for 6D pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 683–698.

[37] J. Liebelt, C. Schmid, and K. Schertler, "Viewpoint-independent object class detection using 3D feature maps," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.

[38] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[39] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár,, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.

[40] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[41] F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep model-based 6D pose refinement in RGB," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 800–815.

[42] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake, "Label fusion: A pipeline for generating ground truth labels for real RGBD data of cluttered scenes," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 3235–3242.

[43] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *J. Guid. Control Dyn.*, vol. 30, no. 4, pp. 1193–1197, 2007.

[44] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. AAAI Conf.* Artif. Intell., 2002, pp. 593–598.

[45] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "Object tracking with an adaptive color-based particle filter," in *Proc. Joint Pattern Recognit. Symp.*, 2002, pp. 353–360.

[46] Y. Oka, T. Kuroda, T. Migita, and T. Shakunaga, "Tracking 3D pose of rigid object by sparse template matching," in *Proc. Int. Conf. Image Graph.*, 2009, pp. 390–397.

[47] K. Pauwels, L. Rubio, J. Diaz, and E. Ros, "Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2347–2354.

[48] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-DOF object pose from semantic keypoints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 2011–2018.

[49] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise voting network for 6DoF pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4561–4570.

[50] V. A. Prisacariu and I. D. Reid, "PWP3D: Real-time segmentation and tracking of 3D objects," *Int. J. Comput. Vis.*, vol. 98, no. 3, pp. 335–354, 2012.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[52] C. Y. Ren, V.O. Prisacariu,I. K.Reid, and D.Murray, "3D tracking of multiple objects with identical appearance using RGB-D input," in *Proc. Int. Conf. 3D Vis.*, 2014, pp. 47–54.

[53] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *Int. J. Comput. Vis.*, vol. 66, no. 3, pp. 231–259, 2006.

[54] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-blackwellized particle filter for multiple target tracking," *Inf. Fusion*, vol. 8, no. 1, pp. 2–15, 2007.

[55] T. Schmidt, R. Newcombe, and D. Fox, "DART: Dense articulated real-time tracking," in *Proc. Robot.: Sci. Syst. Conf.*, 2014.

[56] C. Shan, Y. Wei, T. Tan, and F. Ojardias, "Real time hand tracking by combining particle filtering and mean shift," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2004, pp. 669–674.

[57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.

[58] C. Song, J. Song, and Q. Huang, "HybridPose: 6D object pose estimation under hybrid representations," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020.

[59] R. A. Srivatsan, M. N. X. Zevallos, and H. Choset, "Bingham distribution-based linear filter for online pose estimation," in *Proc. Robot.: Sci. Syst. Conf.*, 2017.

[60] M. Sundermeyer, Z.-C. Marton, M. M. D. Brucker, and R. Triebel, "Implicit 3D orientation learning for 6D object detection from RGB images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 699–715.

[61] M. Sundermeyer, M. Durner, E. Y. Puang, Z.-C. Marton, and R. Triebel, "Multi-path learning for object pose estimation across domains," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13916–13925.

[62] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6D object pose prediction," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 292–301.

[63] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* Cambridge, MA, USA: MIT Press, 2005.

[64] M. Tian, L. Pan, M. H. Ang, Jr., and G. H. Lee, "Robust 6D object pose estimation by learning RGB-D features," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 6218–6224.

[65] H. Tjaden, U.Schwanecke, and E.Schömer, "Real-time monocular segmentation and pose tracking of multiple objects," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 423–438.

[66] H. Tjaden, U. Schwanecke, and E. Schomer, "Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 124–132.

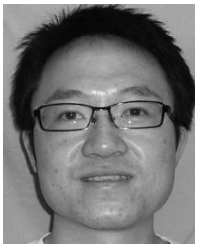
[67] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proc. Int. Conf. Robot Learn.*, 2018, pp. 306–316.
[68] L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for real-time accurate 3D camera tracking," in *Proc. Int. Symp. Mixed Augmented Reality*, 2004, pp. 48–56.
[69] J. Vidal, C.-Y. Lin, and R. Martí, "6D pose estimation using an improved method based on point pair features," in *Proc. IEEE Int. Conf. Control Autom. Robot*., 2018, pp. 405–409.
[70] C. Wang *et al.*, "Densefusion: 6D object pose estimation by iterative dense fusion," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3343–3352.
[71] C. Wang *et al.*, "6-pack: Category-level 6D pose tracker with anchor-based keypoints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 10059–10066.
[72] B. Wen, C. Mitash, B. Ren, and K. E. Bekris, "se(3)-tracknet: Data-driven 6D pose tracking by calibrating image residuals in synthetic domains," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2020.
[73] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, "Probabilistic object tracking using a range camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2013, pp. 3195–3202.
[74] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese, "Monocular multiview object tracking with 3D aspect parts," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 220–235.
[75] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Proc. Robot.: Sci. Syst.*, 2018.
[76] A. Zeng *et al.*, "Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge," in *Proc. IEEE Int. Conf. Robot. Autom*., 2017, pp. 1386–1383.

Xinke Deng received the B.E. degree in aircraft design and engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013, and the M.S. degree in aerospace engineering and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2015 and 2020, respectively.

His research interests include robot visual perception and state estimation.

Arsalan Mousavian received the B.Sc. degree from the Iran University of Science and Technology, Tehran, Iran, in 2010, the M.Sc. degree from the University of Tehran, Tehran, in 2013, both in AI and robotics, and the Ph.D. degree in computer science from George Mason University, Fairfax, VA, USA, in 2018.

He is currently a Senior Research Scientist with NVIDIA, Seattle, WA, USA. His research interests include 3-D perception methods that help robots accomplish robot manipulation tasks in the real world.



Yu Xiang (Member, IEEE) received the B.S. and M.S. degrees in computer science from Fudan University, Shanghai, China, in 2007 and 2010, respectively, and the Ph.D. in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2016.

He is currently a Senior Research Scientist with NVIDIA, Seattle, WA, USA. He was a Visiting Student Researcher with the Stanford Artificial Intelligence Laboratory, Stanford University, from 2013 to 2016. He was a Postdoctoral Researcher with Stanford University, Stanford, CA, USA, and the University of Washington, Seattle, from 2016 to 2017. His work studies how can a robot understand its 3-D environment from sensing and accomplish tasks in the physical world. His research interests include robotics and computer vision.

Fei Xia (Member, IEEE) received the bachelor's degree in automation from Tsinghua University, Beijing, China, in 2016, and the master's degree in electrical engineering in 2019 from Stanford University, Stanford, CA, USA, where he is currently working toward the Ph.D. degree under the supervision of Silvio Savarese and Leo Guibas.

His research interests include robot simulation, robot learning, and simulation-to-real transfer of robot skills.

Timothy Bretl (Member, IEEE) received the B.S. degree in engineering and the B.A. degree in mathematics from Swarthmore College, Swarthmore, PA, USA, in 1999, and the M.S. and Ph.D. degrees in aeronautics and astronautics from Stanford University, Stanford, CA, USA, in 2000 and 2005, respectively.

Subsequently, he was a Postdoctoral Fellow with the Department of Computer Science, Stanford University. Since 2006, he has been with the University of Illinois at Urbana–Champaign, Urbana, IL, USA, where he is currently an Associate Professor of Aerospace Engineering and a Research Associate Professor with 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, the William L. Everett Award for Teaching Excellence in 2016, the Rose Award for Teaching Excellence in 2016, the College of Engineering Teaching Excellence Award in 2018, and the Campus Award for Excellence in Undergraduate Teaching in 2018.

Dieter Fox (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Bonn, Bonn, Germany, in 1998.

He is currently a Professor with the Allen School of Computer Science and Engineering, University of Washington, Seattle, WA, USA, where he heads the UW Robotics and State Estimation Lab. He is also Senior Director of Robotics Research with NVIDIA. He has authored or coauthored more than 200 technical papers. He is a co-author of the textbook entitled *Probabilistic Robotics* (Cambridge, MA, USA: MIT Press, 2005). His research interests include 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.

Dr. Fox is the recipient of the IEEE RAS Pioneer Award. He was an Editor for the IEEE TRANSACTIONS ON ROBOTICS, the Program Co-Chair of the 2008 AAAI Conference on Artificial Intelligence, and the Program Chair of the 2013 Robotics: Science and Systems conference. He is a Fellow of the Association for the Advancement of Artificial Intelligence and the Association for Computing Machinery.