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

Xinke Deng*†, Arsalan Mousavian*, Yu Xiang*, Fei Xia*‡, Timothy Bretl†, Dieter Fox*§
*NVIDIA †University of Illinois at Urbana-Champaign
‡Stanford University §University of Washington

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

I. INTRODUCTION

Estimating the 6D pose of objects from camera images, i.e., 3D rotation and 3D translation of an object with respect to the camera, is an important problem in robotic applications. For instance, in robotic manipulation, 6D pose estimation of objects provides critical information to the robot for planning and executing grasps. In robotic navigation tasks, localizing objects in 3D provides useful information for planing and obstacle avoidance. Due to its significance, various efforts have been devoted to tackling the 6D pose estimation problem from both the robotics community [7, 4, 43, 40] and the computer vision community [32, 21, 12].

Traditionally, the 6D pose of an object is estimated using local-feature or template matching techniques, where features extracted from an image are matched against features or viewpoint templates generated for the 3D model of the object. The 6D object pose can then be recovered using 2D-3D correspondences of these local features or by selecting the best matching viewpoint onto the object [7, 11, 12]. More recently, machine learning techniques have been employed to detect key points or learn better image features for matching [2, 18]. Thanks to advances in deep learning, convolutional neural networks have recently been shown to significantly boost the estimation accuracy and robustness [15, 44, 30, 38, 43],

So far, the focus of image-based 6D pose estimation has been on the *accuracy of single image estimates*; most techniques ignore temporal information and provide only a single hypothesis for an object pose. In robotics, however, temporal data and information about the *uncertainty* of estimates can also be very important for tasks such as grasp planning or active sensing. Temporal tracking in video data can improve

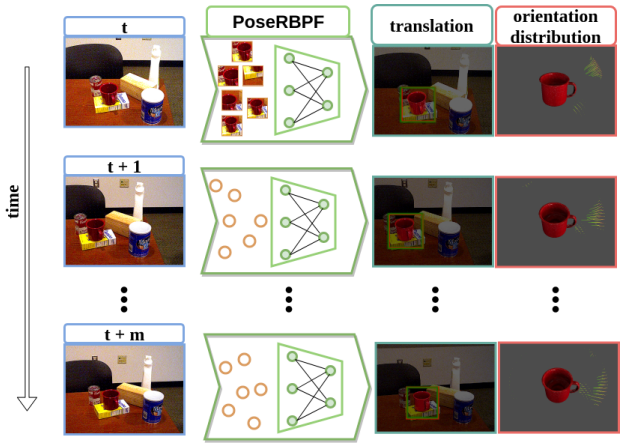

Fig. 1. Overview of our PoseRBPF framework for 6D object pose tracking. Our method leverages a Rao-Blackwellized particle filter and an auto-encoder network to estimate the 3D translation and a full distribution of the 3D rotation of a target object from a video sequence.

pose estimation [28, 5, 17, 8]. In the context of point-cloud based pose estimation, Kalman filtering has also been used to track 6D poses, where Bingham distributions have been shown to be well suited for orientation estimation [36]. However, unimodal estimates are not sufficient to adequately represent the complex uncertainties arising from occlusions and possible object symmetries.

In this work, we introduce a particle filter-based approach to estimate full posteriors over 6D object poses. Our approach, called PoseRBPF, factorizes the posterior into the 3D translation and the 3D rotation of the object, and uses a Rao-Blackwellized particle filter that samples object poses and estimates discretized distributions over rotations for each particle. To achieve accurate estimates, the 3D rotation is discretized at 5 degree resolution, resulting in a distribution over $72 \times 37 \times 72 = 191,808$ bins for each particle (elevation ranges only from -90 to 90 degree). To achieve real time performance, we pre-compute a codebook over embeddings for all discretized rotations, where embeddings come from an auto-encoder network trained to encode the visual appearance of an object from arbitrary viewpoints at a certain scale (inspired by [37]). For each particle, PoseRBPF first uses the 3D translation to determine the center and size of the object bounding box in the image, then determines the embedding for that bounding box, and finally updates the rotation distribution by comparing the embedding value with the pre-computed entries in the

codebook using cosine distance. The weight of each particle is given by the normalization factor of the rotation distribution. Motion updates are performed efficiently by sampling from a motion model over poses and a convolution over the rotations. Fig. 1 illustrates our PoseRBPF framework for 6D object pose tracking. Experiments on the YCB-Video dataset [43] and the T-Less dataset [14] show that PoseRBPFs are able to represent uncertainties arising from various types of object symmetries and can provide more accurate 6D pose estimation.

Our work makes the following main contributions:

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

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

II. Related Work

Our work is closely related to recent advances in 6D object pose estimation using deep neural networks. The current trend is to augment state-of-the-art 2D object detection networks with the ability to estimate 6D object pose. For instance, [15] extend the SSD detection network [24] to 6D pose estimation by adding viewpoint classification to the network. [38] utilize the YOLO architecture [31] to detect 3D bounding box corners of objects in the images, and then recover the 6D pose by solving the PnP problem. PoseCNN [43] designs an end-to-end network for 6D object pose estimation based on the VGG architecture [35]. Although these methods significantly improve the 6D pose estimation accuracy over the traditional methods [12, 2, 18], they still face difficulty in dealing with symmetric objects, where most methods manually specify the symmetry axis for each such object. In contrast, [37] introduce an implicit way of representing 3D rotations by training an auto-encoder for image reconstruction, which does not need to pre-define the symmetry axes for symmetric objects. We leverage this implicit 3D rotation representation in our work, and show how to combine it with particle filtering for 6D object pose tracking.

The particle filtering framework has been widely applied to different tracking applications in the literature [27, 34, 16, 33], 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 [1, 6, 29, 42, 20]. However, in order to achieve good tracking performance, a particle filter requires a strong observation model. Also, the tracking frame rate is limited by the particle sampling efficiency. In this work, we factorize the 6D object pose tracking problem and deploy Rao-Blackwellized particle filters [10], which have been shown to scale to complex estimation problems such as SLAM [39, 26] and multi-model target tracking [19, 33]. We also employ a deep neural network as an observation model that provides robust estimates for object orientations even under occlusions and symmetries. Our design allows us to evaluate all possible orientations in parallel using an efficient GPU implementation. As a result, our method can track the distribution of the 6D pose of an object at 20fps.

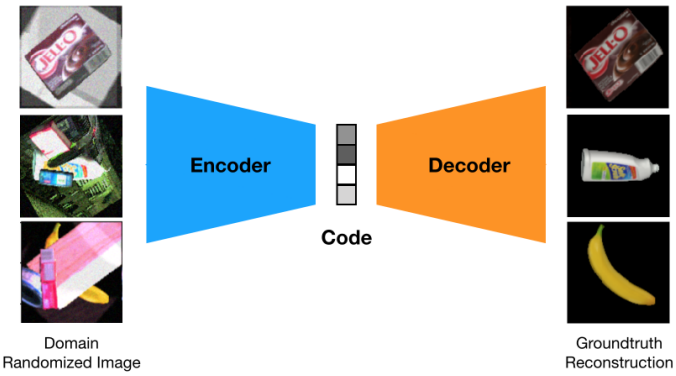

Fig. 2. Illustration of the inputs and outputs of the auto-encoder. Images with different lighting, background and occlusion are feed into the network to reconstruct synthetic images of the objects from the same 6D poses. The encoder generates a feature embedding (code) of the input image.

III. 6D Object Pose Tracking with PoseRBPF

The goal of 6D object pose tracking of an object is to estimate the 3D rotation $\mathbf{R}$ and the 3D translation $\mathbf{T}$ of the object for every frame in an image stream. In this section, we first formulate the 6D object tracking problem in a particle filtering framework, and then describe how to utilize a deep neural network to compute the likelihoods of the particles and to achieve an efficient sampling strategy for tracking.

A. Rao-Blackwellized Particle Filter Formulation

At time step k, given observations $\mathbf{Z}_{1:k}$ up to time k, our primary goal is to estimate the posterior distribution of the 6D pose of an object $P(\mathbf{R}_k, \mathbf{T}_k | \mathbf{Z}_{1:k})$, where $\mathbf{R}_k$ and $\mathbf{T}_k$ denote the 3D rotation and 3D translation of the object at time k, respectively. Using a vanilla particle filter to sample over this 6D space is not feasible, especially when there is large uncertainty over the orientation of the object. Such uncertainties occur frequently when objects are heavily occluded or have symmetries that result in multiple orientation hypotheses. We thus propose to factorize the 6D pose estimation problem into 3D rotation estimation and 3D translation estimation. This idea is based on the observation that the 3D translation can be estimated from the location and the size of the object in the image. The translation estimation provides the center and scale of the object in the image, based on which the 3D rotation can be estimated from the appearance of the object inside the bounding box. Specifically, we decompose the posterior into:

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

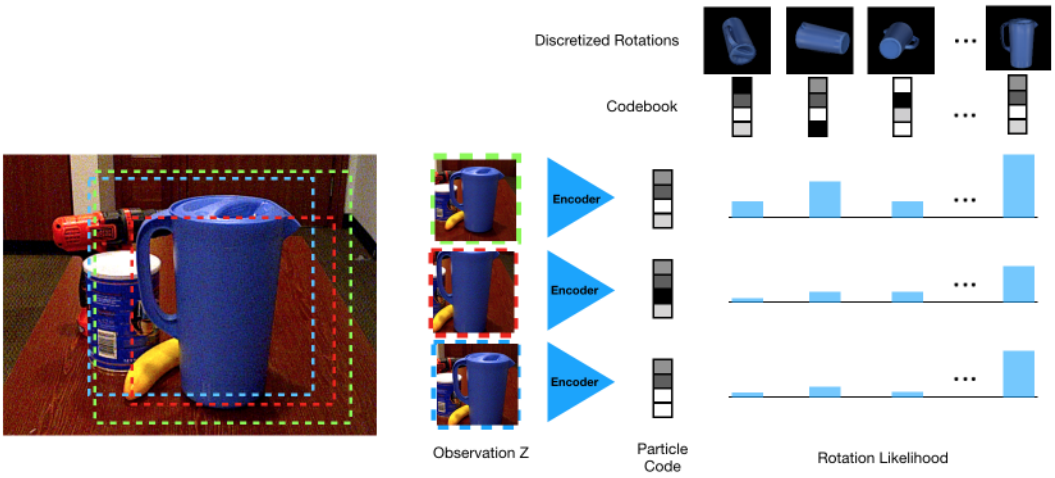

Fig. 3. Illustration of the computation for the conditional rotation likelihood by codebook matching. Left) Each particle crops the image based on its translation hypothesis. The RoI for each particle is resized and the corresponding code is computed using the encoder. Right) The rotation distribution $P(\mathbf{R}|\mathbf{Z},\mathbf{T})$ is computed from the distance between the code for each hypothesis and those in the codebook.

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 [10, 39], 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 orientation conditioned on the translation and the images, and w_k^i is the importance weight. To achieve accurate pose estimation, the 3D object orientation consisting of azimuth, elevation, and in-plane rotation is discretized into bins of size 5 degree, resulting in a distribution over $72\times37\times72 = 191,808$ bins for each particle (elevation ranges only from -90 to 90 degrees). At every time step k, the particles are propagated through a motion model to generate a new set of particles $\mathcal{X}_{k+1}$, from which we can estimate the 6D pose distribution.

B. Observation Likelihoods

The observation likelihoods of the two posteriors $P(\mathbf{Z}_k|\mathbf{T}_k)$ and $P(\mathbf{Z}_k|\mathbf{T}_k,\mathbf{R}_k)$ measure the compatibility of the observation $\mathbf{Z}_k$ with the object pose at the 3D rotation $\mathbf{R}_k$ and the 3D translation $\mathbf{T}_k$. According to the Bayes Rule

$$P(\mathbf{Z}_k|\mathbf{T}_k,\mathbf{R}_k) \propto P(\mathbf{R}_k|\mathbf{T}_k,\mathbf{Z}_k)P(\mathbf{Z}_k|\mathbf{T}_k), \qquad (2)$$

it is sufficient to estimate the likelihood $P(\mathbf{Z}_k|\mathbf{T}_k,\mathbf{R}_k)$ by computing $P(\mathbf{R}_k|\mathbf{T}_k,\mathbf{Z}_k)$ and $P(\mathbf{Z}_k|\mathbf{T}_k)$. Intuitively, a 6D object pose estimation method, such as [15, 38, 43], can be employed to estimate the observation likelihoods. However, these methods only provide a single estimation of the 6D pose instead of estimating a probability distribution, i.e., there is no uncertainty in their estimation. Also, these methods are computationally expensive if we would like to evaluate a large number of samples in the particle filtering.

Ideally, if we can synthetically generate an image of the object with the pose $(\mathbf{R}_k,\mathbf{T}_k)$ into the same scene as the observation $\mathbf{Z}_k$, we can compare the synthetic image with the input image $\mathbf{Z}_k$ to measure the likelihoods. However, this is not feasible since it is very difficult to synthesize the same lighting, background or even occlusions between objects as in the input video frame. In contrast, it is straightforward to render a synthetic image of the object using constant lighting, blank background and no occlusion, given the 3D model of the object. Therefore, inspired by [37], we apply an auto-encoder to transform the observation $\mathbf{Z}_k$ into the same domain as the synthetic rendering of the object. Then we can compare image features in the synthetic domain to measure the likelihoods of 6D poses efficiently.

1) Auto-encoder: An auto-encoder is trained to map an image $\mathbf{Z}$ of the target object with pose $(\mathbf{R},\mathbf{T})$ to a synthetic image $\mathbf{Z}'$ of the object rendered from the same pose, where the synthetic image $\mathbf{Z}'$ is rendered using constant lighting, and there is no background and occlusion in the synthetic image. In this way, the auto-encoder is forced to map images with different lighting, background and occlusion to the common synthetic domain. Fig. 2 illustrates the input and output of the auto-encoder during training. In addition, the auto-encoder learns a feature embedding $f(\mathbf{Z})$ of the input image.

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

Fig. 4. Visualization of reconstruction of the RoIs from auto-encoder. Left is the groundtruth RoI. The other two column show the reconstruction with shifting and scale change. As it is shown the reconstruction quality degrades with deviations from groundtruth RoI. This property makes auto-encoder a suitable choice for computing the observation likelihoods.

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

$$\begin{bmatrix} u_k \\ v_k \end{bmatrix} = \begin{bmatrix} f_x \frac{x_k}{z_k} + p_x \\ f_y \frac{y_k}{z_k} + p_y \end{bmatrix}, \tag{3}$$

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

The RoI is feed into the encoder to compute the feature embedding $f(\mathbf{Z}_k(\mathbf{T}_k))$. 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 rotation likelihood:

$$P(\mathbf{R}_c^j|\mathbf{Z}_k, \mathbf{T}_k) \propto \phi\Big(\frac{f(\mathbf{Z}_k(\mathbf{T}_k)) \cdot f(\mathbf{Z}(\mathbf{R}_c^j, \mathbf{T}_0))}{\|f(\mathbf{Z}_k(\mathbf{T}_k))\| \cdot \|f(\mathbf{Z}(\mathbf{R}_c^j, \mathbf{T}_0))\|}\Big), \tag{4}$$

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. In this way, we can obtain a probabilistic likelihood distribution of all the rotations in the codebook given a translation. Fig. 3 illustrates the computation of the rotation likelihoods by the cookbook matching.

Since the auto-encoder is trained with the object being at the center of the image and at a certain scale, i.e., with the canonical translation $\mathbf{T}_0$, any change in scale or deviation of the object from the center of the image results in poor reconstructions (see Fig. 4). 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. We utilize this property to compute the translation likelihood $P(\mathbf{Z}_k|\mathbf{T}_k)$. Intuitively, if the translation $\mathbf{T}_k$ is correct, the similarity scores in Eq. (4) for rotation $\mathbf{R}^i$ that is close to the ground truth rotation would be high. Specifically, $P(\mathbf{Z}_k|\mathbf{T}_k)$ is computed as the sum of the probability density $P(\mathbf{R}_c^j|\mathbf{T}_k, \mathbf{Z}_k)$ for all the discrete rotations.

C. Motion Priors

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

$$P(\mathbf{T}_k|\mathbf{T}_{k-1}, \mathbf{T}_{k-2}) = \mathcal{N}\left(\mathbf{T}_{k-1} + \alpha(\mathbf{T}_{k-1} - \mathbf{T}_{k-2}), \mathbf{\Sigma}_\mathbf{T}\right), \tag{5}$$

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

$$P(\mathbf{R}_k|\mathbf{R}_{k-1}) = \mathcal{N}\left(\mathbf{R}_{k-1}, \mathbf{\Sigma}_\mathbf{R}\right), \tag{6}$$

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

D. 6D Object Pose Tracking Framework

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

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

$$P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_{1:k}) \propto P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_k) P(\mathbf{R}_k|\mathbf{R}_{k-1}), \tag{7}$$

where $P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_k)$ is the rotation distribution defined in Eq. (4), and $P(\mathbf{R}_k|\mathbf{R}_{k-1})$ is the motion prior. Finally, we compute the posterior of the translation $P(\mathbf{T}_k^i|\mathbf{Z}_{1:k})$ with

$$P(\mathbf{T}_k^i|\mathbf{Z}_{1:k}) \propto \sum_{\mathbf{R}_k} P(\mathbf{Z}_k|\mathbf{T}_k^i, \mathbf{R}_k) P(\mathbf{R}_k|\mathbf{T}_{1:k-1}^i, \mathbf{Z}_{1:k-1}), \tag{8}$$

and use it as the weight w^i of this particle. The systematic resampling method [9] is used to resample the particles according to the weights $w^{1:N}$. Our 6D object pose tracking framework is shown in Fig. 5.

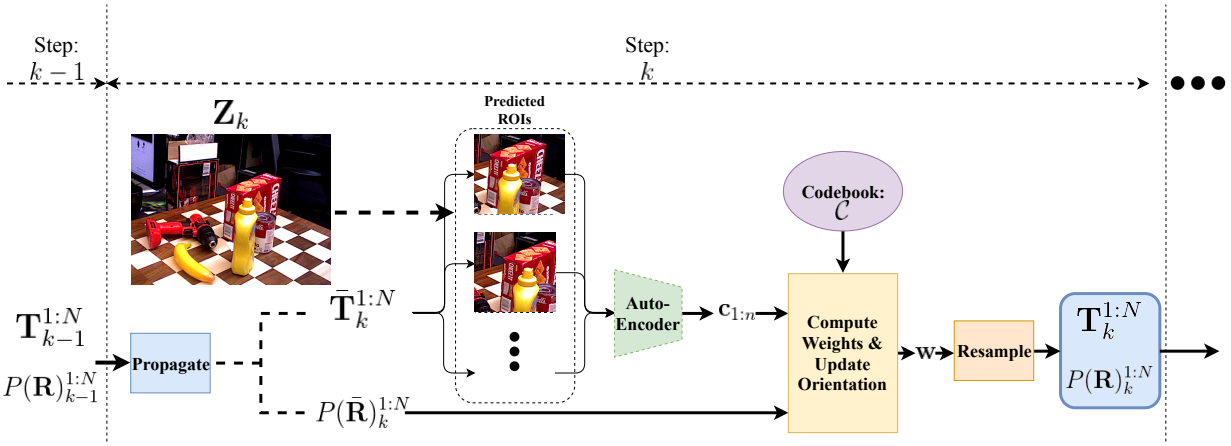

Fig. 5. We propose PoseRBPF, a Rao-Blackwellized particle filter for 6D object pose tracking. For each particle, the *orientation distribution* is estimated conditioned on translation estimation, while the translation estimation is evaluated with the corresponding RoIs.

Some robotic tasks require the expectation of the 6D pose of the object from the particle filter for decision making. The expectation can be represented as $(\mathbf{T}_k^E, \mathbf{R}_k^E)$. The translation expectation can be computed simply by averaging the translation estimations $\mathbf{T}_k^{1:N}$ for all the N particles due to the uni-modal nature of translation in the object tracking task. Computing the rotation expectation $\mathbf{R}_k^E$ is less obvious since the distribution $P(\mathbf{R}_k)$ might be multi-modal and simply performing weighted averaging over all the discrete rotations is not meaningful. To compute the rotation expectation, we first summarize the rotation distribution for all the particles by taking the maximum probability for every discrete rotation, resulting in rotation distribution $P(\mathbf{R}^E)_k$. 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 [25].

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 maximal score is lower than a pre-defined threshold, we determine it is a tracking failure. Algorithm 1 summarizes our Rao-Blackwellized particle filter for 6D object pose tracking.

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

Algorithm 1: 6D Object Pose Tracking with PoseRBPF

E. RGB-D Extension of PoseRBPF

PoseRBPF can be extended to use depth measurements for computing the observation likelihoods. With the RGB image $\mathbf{Z}_k^C$ and the additional depth measurements $\mathbf{Z}_k^D$, the observation likelihood in Eq. (2) can be rewritten as:

$$\begin{aligned} P(\mathbf{Z}_k|\mathbf{T}_k, \mathbf{R}_k) &= P(\mathbf{Z}_k^C, \mathbf{Z}_k^D|\mathbf{T}_k, \mathbf{R}_k) \\ &\propto P(\mathbf{R}_k|\mathbf{T}_k, \mathbf{Z}_k^C) P(\mathbf{Z}_k^C|\mathbf{T}_k) P(\mathbf{Z}_k^D|\mathbf{T}_k). \end{aligned} \tag{9}$$

Note that the auto-encoder only uses the RGB image. Therefore, $P(\mathbf{R}_k|\mathbf{T}_k, \mathbf{Z}_k^C, \mathbf{Z}_k^D) = P(\mathbf{R}_k|\mathbf{T}_k, \mathbf{Z}_k^C)$. To compute the likelihood with the depth image $P(\mathbf{Z}_k^D|\mathbf{T}_k^i)$ for a translation hypothesis $\mathbf{T}_k^i$, we first render the object with pose $(\mathbf{T}_k^i, \mathbf{R}_k^*)$, where $\mathbf{R}_k^* = \arg\max_{\mathbf{R}_k} P(\mathbf{R}_k|\mathbf{T}_k^i, \mathbf{Z}_k^C)$ from the color image. By comparing the rendered depth image $\hat{\mathbf{Z}}_k^{Di}$ with the depth measurements $\mathbf{Z}_k^D$, we first 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, the rendered pixel p with depth less than $\mathbf{Z}_k^D(p) + m$ is determined as visible. With the estimated visibility mask, the *visible depth discrepancy* between the two depth maps is computed as:

$$\Delta_k^i(\hat{\mathbf{Z}}_k^{Di}, \mathbf{Z}_k^D, \hat{V}_k^i, \tau) = \underset{p \in \hat{V}_k^i}{\text{avg}} \Big(\min \big(\frac{|\mathbf{Z}_k^D(p) - \hat{\mathbf{Z}}_k^{Di}(p)|}{\tau}, 1 \big) \Big), \tag{10}$$

where τ is a pre-defined threshold for each object. For every particle, we compute its *depth score* $s_d^i = v_k^i(1 - \Delta_k^i)$, 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 $P(\mathbf{Z}_k^D|\mathbf{T}_k^i)$ as $\phi'(s_d^i)$, where $\phi'(\cdot)$ is a gaussian probability density function centered at the maximum depth score among all the particles.

TABLE I
EFFECT OF THE NUMBER OF PARTICLES ON FRAME RATE IN TRACKING.

Number of particles	50	100	200	400
Frame rate (RGB)	20.3	11.5	6.1	3.1
Frame rate (RGB-D)	14.8	9.5	5.0	2.8

IV. EXPERIMENTS

A. Datasets

We evaluated our method on two datasets: the YCB Video dataset [43] and the T-LESS dataset [14].

YCB Video dataset: The YCB video dataset contains RGB-D video sequences of 21 objects from the YCB Object and Model Set [3]. It contains textured and textureless household objects put in different arrangements. Objects are annotated with 6D object poses and two metrics are used for quantitative evaluation. The first metric is ADD, which is the average distance between the corresponding 3D points on the object at groundtruth pose vs the predicted pose. The second metric is ADD-S, which is the average distance between the *closest point* between the 3D model of the object at groundtruth and the model of the object at the predicted pose. ADD-S is designed for symmetric objects, since it focuses on shape matching, rather than exact pose matching.

T-LESS: This dataset contains RGB-D sequences of 30 non-textured industrial objects. Evaluation is done on 20 test scenes. The dataset is challenging because the objects do not have texture and they have various forms of symmetries and occlusions. We follow the evaluation pipeline in SIXD challenge and used Visible Surface Discrepancy err_{vsd} [13] to evaluate the quality of the pose estimation. Visual surface discrepancy is computed as mean average of the distance between the visible points. The metric is the recall of correct 6D poses where $err_{vsd} < 0.3$ with tolerance $20mm$ and visibility of more than 10%.

B. Implementation Details

The auto-encoder is trained for each object separately for $150,000$ iterations with batch size of 64 using the Adam optimizer with learning rate of 0.0002. The auto-encoder is optimized with the L2 loss on the N pixels with largest reconstruction errors. Larger Ns are more suitable for textured objects to capture more details. We use $N = 2000$ for textured objects and $N = 1000$ for non-textured objects. The training data is generated by rendering the object at random rotation and superimposed at random crops of the MS-COCO dataset [22] at resolution 128×128. In addition to the target object, three additional objects are sampled at random locations and scales to generate training data with occlusions. The target object is positioned at the center of the image and jittered with 5 pixels, the object is sampled uniformly at scales between 0.975 and 1.025 with random lighting. Color is randomized in HSV space and we also add Gaussian noise to pixel values 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 architecture of the network is described in [37]. It consists of four 5×5 convolutional layers and four 5×5 deconvolutional layers for the encoder and the decoder, respectively. The standard deviations used to compute observation likelihoods in Eq. (4) are selected between 0.03 and 0.1. The codebook for each object is pre-computed offline and loaded during test time. Computation of observation likelihood is done efficiently on a GPU. Table I shows the frame rate at which PoseRBPF can process images.

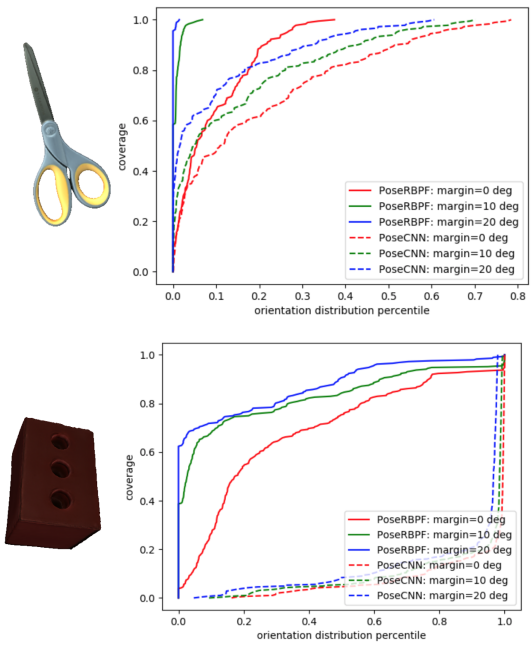

Fig. 6. Rotation Coverage Percentile comparison between PoseRBPF and PoseCNN for scissors and foam brick. Foam brick has 180° planar rotation and scissors is an asymmetric object.

C. Results on YCB Video Dataset

Table II shows the pose estimation results on the YCB video dataset, where we compare with the state-of-the-art methods for pose estimation using RGB images [43, 40] and RGB-D images [43, 41]. We initialize PoseRBPF using PoseCNN at the first frame or after the object was heavily occluded. On average, this happened only 1.03 times per sequence. As can be seen, our method significantly improves the accuracy of 6D pose estimation when using 200 particles. Note that our method handles symmetric objects such as 024_bowl, 061_foam_brick much better. One of the objects on which PoseRBPF performs poorly is 036_wood_block, which is caused by the difference in texture of the 3D model of the wooden block and the texture of the wooden block used in the real images. In addition, the physical dimensions of the wooden block are different between real images and the model contained in this dataset. Another observation is that with the increase in the number of particles, the accuracy improves significantly because with more samples the variations in scale and translation of an object are covered much better.

It has been shown in the context of robot localization that adding samples drawn according to the most recent

TABLE II
RESULTS ON YCB VIDEO DATASET

	RGB										RGB-D					
	PoseCNN [43]		DOPE [40]		PoseRBPF 50 particles		PoseRBPF 200 particles		PoseRBPF++ 200 particles		PoseCNN+ICP [43]		DenseFusion [41]		PoseRBPF 200 particles	
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
002_master_chef_can	50.9	84.0	-	-	56.1	75.6	58.0	77.1	**63.3**	**87.5**	69.0	95.8	-	**96.4**	**90.5**	95.1
003_cracker_box	51.7	76.9	55.9	69.8	73.4	85.2	76.8	87.0	**77.8**	**87.6**	80.7	91.8	-	**95.5**	**88.2**	93.0
004_sugar_box	68.6	84.3	75.7	87.1	73.9	86.5	75.9	87.6	**79.6**	**89.4**	**97.2**	**98.2**	-	97.5	92.9	95.5
005_tomato_soup_can	66.0	80.9	**76.1**	**85.1**	71.1	82.0	74.9	84.5	73.0	83.6	81.6	94.5	-	**94.6**	**90.0**	93.8
006_mustard_bottle	79.9	90.2	81.9	90.9	80.0	90.1	82.5	91.0	**84.7**	**92.0**	**97.0**	**98.4**	-	97.2	91.9	96.3
007_tuna_fish_can	**70.4**	**87.9**	-	-	56.1	73.8	59.0	79.0	64.2	82.7	83.1	97.1	-	96.6	**91.1**	95.3
008_pudding_box	**62.9**	**79.0**	-	-	54.8	69.2	57.2	72.1	64.5	77.2	**96.6**	**97.9**	-	96.5	85.8	92.0
009_gelatin_box	75.2	87.1	-	-	83.1	89.7	**88.8**	**93.1**	83.0	90.8	**98.2**	**98.8**	-	98.1	96.3	97.5
010_potted_meat_can	**59.6**	**78.5**	39.4	52.4	47.0	61.3	49.3	62.0	51.8	66.9	**83.8**	**92.8**	-	91.3	68.7	77.9
011_banana	**72.3**	**85.9**	-	-	22.8	64.1	24.8	61.5	18.4	66.9	**91.6**	**96.9**	-	96.6	74.2	86.9
019_pitcher_base	52.5	76.8	-	-	74.0	87.5	**75.3**	**88.4**	63.7	82.1	**96.7**	**97.8**	-	97.1	86.8	94.2
021_bleach_cleanser	50.5	71.9	-	-	51.6	66.7	54.5	69.3	**60.5**	**74.2**	**92.3**	**96.8**	-	95.8	86.0	93.0
024_bowl	6.5	69.7	-	-	26.4	**88.2**	**36.1**	86.0	28.4	85.6	17.5	78.3	-	88.2	**25.5**	**94.2**
025_mug	57.7	78.0	-	-	67.3	83.7	70.9	85.4	**77.9**	**89.0**	81.4	95.1	-	**97.1**	**90.9**	**97.1**
035_power_drill	55.1	72.8	-	-	64.4	80.6	70.9	**85.0**	**71.8**	84.3	96.9	98.0	-	96.0	93.9	**96.1**
036_wood_block	**31.8**	**65.8**	-	-	0.0	0.0	2.8	33.3	2.3	31.4	**79.2**	**90.5**	-	89.7	20.1	89.1
037_scissors	35.8	56.2	-	-	20.6	30.9	21.7	33.0	**38.7**	**59.1**	**78.4**	92.2	-	**95.2**	76.1	85.6
040_large_marker	58.0	71.4	-	-	45.7	54.1	48.7	59.3	**67.1**	**76.4**	85.4	97.2	-	**97.5**	**92.0**	97.1
051_large_clamp	25.0	49.9	-	-	27.0	73.2	**47.3**	**76.9**	38.3	59.3	**52.6**	75.4	-	72.9	48.5	**94.8**
052_extra_large_clamp	15.8	47.0	-	-	50.4	68.7	**26.5**	**69.5**	32.3	44.3	28.7	65.3	-	69.8	**40.3**	**90.1**
061_foam_brick	40.4	87.8	-	-	75.8	88.4	78.2	89.7	**84.1**	**92.6**	48.3	**97.1**	-	92.5	**81.1**	95.7
ALL	53.7	75.9	-	-	57.1	74.8	59.9	77.5	**62.1**	**78.4**	79.3	93.0	-	93.1	**80.8**	**93.3**

Fig. 7. Visualization of estimated poses on the YCB Video dataset (left) and T-LESS dataset (right). 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.

observation can improve the localization performance [39]. Here, we applied such a technique by sampling 50% of the particles around PoseCNN predictions and the other 50% from the particles of the previous time step. Our results show that such a hybrid version, PoseRBPF++, further improves the pose estimation accuracy of our approach. Fig. 7 illustrates the 6D pose estimation on YCB Video dataset. Depth measurements contain useful information to improve the pose estimation accuracy. By comparing the depth of the object rendered at the estimated pose and the depth image (explained in Sec III-E), our method achieves the state-of-the-art performance. The comparison between RGB and RGB-D versions of PoseRPBF shows using depth information with the same number of particles improves the accuracy of estimated poses significantly. Note that depth information is only used during inference and the encoder takes only the RGB images.

D. Results on T-LESS Dataset

Table III presents our results on the T-LESS dataset. T-LESS is a challenging dataset because objects do not have texture and objects are occluded frequently in different frames. We compared our method with [37] which uses a similar auto-encoder, but does not use any temporal information. We evaluated both using ground truth bounding boxes and the detection output from RetinaNet [23] that is used in [37]. Our tracker uses 100 particles, and is reinitialized whenever the observation likelihood drops below a threshold. The results show that the recall for correct object poses doubles by tracking the object pose rather than just predicting object pose from single images in the RGB case. With additional depth images, the recall can be further improved by around 76%, and PoseRBPF outperforms refining [37] with ICP by 28%. 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, recall increases significantly for all the methods and the particle filter consistently outperforms [37], which shows the importance of temporal tracking for object pose estimation.

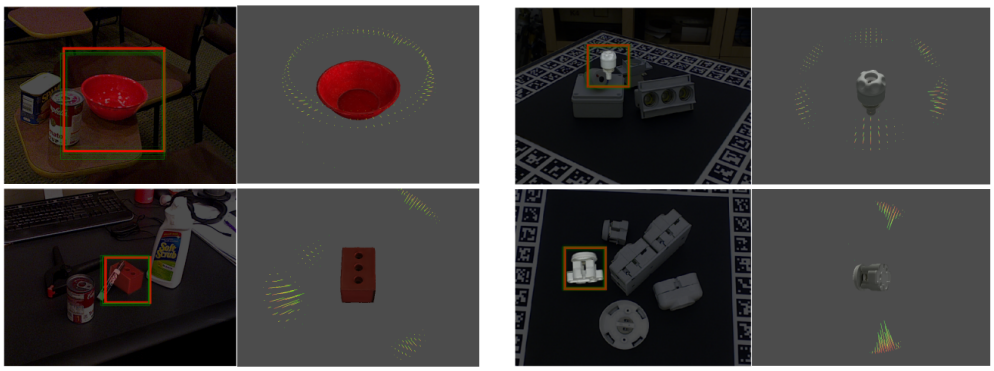

Fig. 8. Visualization of rotation distributions. For each image, the distribution over the rotation is visualized. The lines represent the probability for rotations that are higher than a threshold. The length of each line is proportional to the probability of that viewpoint. As can be seen, PoseRBPF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

TABLE III
T-LESS RESULTS: OBJECT RECALL FOR $e_{vsd} < 0.3$ ON ALL PRIMESENSE TEST SCENES

Object	Without GT 2D BBs					With GT 2D BBs	
	RGB			RGB-D			
	SSD [37]	RetinaNet [37]	RetinaNet PoseRBPF	RetinaNet [37] + ICP	RetinaNet PoseRBPF	[37]	PoseRBPF
1	5.65	8.87	**27.60**	22.32	**61.30**	12.33	**80.90**
2	5.46	13.22	**26.60**	29.49	**63.10**	11.23	**85.80**
3	7.05	12.47	**37.70**	38.26	**74.30**	13.11	**85.60**
4	4.61	6.56	**23.90**	23.07	**64.50**	12.71	**62.00**
5	36.45	34.80	**54.40**	76.10	**86.70**	66.70	**89.80**
6	23.15	20.24	**73.00**	67.64	**71.50**	52.30	**97.80**
7	15.97	16.21	**51.60**	73.88	**88.00**	36.58	**91.20**
8	10.86	19.74	**37.90**	67.02	**84.00**	22.05	**95.60**
9	19.59	36.21	**41.60**	78.24	**86.00**	46.49	**77.10**
10	10.47	11.55	**41.50**	**77.65**	74.30	14.31	**85.30**
11	4.35	6.31	**38.30**	35.89	**62.60**	15.01	**89.50**
12	7.80	8.15	**39.60**	49.30	**71.00**	31.34	**91.20**
13	3.30	4.91	**20.40**	42.50	**42.10**	13.60	**89.30**
14	2.85	4.61	**32.00**	30.53	**50.10**	45.32	**70.20**
15	7.90	26.71	**41.60**	**83.73**	76.60	50.00	**96.60**
16	13.06	21.73	**39.10**	67.42	**83.80**	36.09	**97.00**
17	41.70	**64.84**	40.00	**86.17**	78.40	81.11	**87.00**
18	47.17	14.30	**47.90**	**84.34**	81.10	52.62	**89.70**
19	15.95	22.46	**40.60**	50.54	**61.80**	50.75	**83.20**
20	2.17	5.27	**29.60**	14.75	**55.00**	37.75	**70.00**
21	19.77	17.93	**47.20**	40.31	**72.70**	50.89	**84.40**
22	11.01	18.63	**36.60**	35.23	**63.80**	47.60	**77.70**
23	7.98	18.63	**42.00**	42.52	**82.40**	35.18	**85.90**
24	4.74	4.23	**48.20**	59.54	**83.20**	11.24	**91.80**
25	21.91	18.76	**39.50**	70.89	**77.70**	37.12	**88.70**
26	10.04	12.62	**47.80**	66.20	**85.00**	28.33	**90.90**
27	7.42	21.13	**41.30**	**73.51**	68.00	21.86	**79.10**
28	21.78	23.07	**49.50**	61.20	**79.30**	42.58	**72.10**
29	15.33	26.65	**60.50**	73.04	**86.30**	57.01	**96.00**
30	34.63	29.58	**52.70**	**92.90**	80.10	70.42	**77.00**
Mean	14.67	18.35	**41.67**	57.14	**73.16**	36.79	**85.28**

Fig. 7 shows the 6D pose estimation of PoseRBPF on several T-LESS images.

E. Analysis of Rotation Distribution

Unlike other 6D pose estimation methods that output a single estimate for the 3D rotation of an object, PoseRBPF tracks full distributions over object rotations. Fig. 8 shows example distributions for some objects. 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's capture rotation uncertainty, we compared PoseRBPF estimates to those of PoseCNN assuming a Gaussian uncertainty with mean at the PoseCNN estimate. Fig. 6 shows this comparison for the scissors and foam brick objects. Here, the x-axis ranges over percentiles of the rotation distributions, and the y-axis shows how often the ground truth pose is within 0, 10, or 20 degrees of one of the rotations contained in the corresponding percentile. For instance, for the scissors, the red, solid line indicates that 80% of the time, the ground truth rotation is within 20 degrees of an rotation taken from the top 20% of the PoseRBPF distribution. If we take the top 20% rotations estimated by PoseCNN assuming a Gaussian uncertainty, this number drops to about 60%, as indicated by the lower dashed, red line. The importance of maintaining multi-modal uncertainties becomes even more prominent for the foam brick, which has a $180°$ symmetry. Here, PoseRBPF achieves high coverage, whereas PoseCNN fails to generate good rotation estimates even when moving further from the generated estimate.

V. CONCLUSION

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

References

[1] Pedram Azad, David Münch, Tamim Asfour, and Rüdiger Dillmann. 6-DoF model-based tracking of arbitrarily shaped 3D objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5204–5209, 2011.

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *European Conference on Computer Vision (ECCV)*, pages 536–551, 2014.

[3] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In IEEE, editor, *IEEE International Conference on Advanced Robotics (ICAR)*, July 2015.

[4] Zhe Cao, Yaser Sheikh, and Natasha Kholgade Banerjee. Real-time scalable 6DOF pose estimation for textureless objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2441–2448, 2016.

[5] Changhyun Choi and Henrik I Christensen. 3D textureless object detection and tracking: An edge-based approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3877–3884, 2012.

[6] Changhyun Choi and Henrik I Christensen. Robust 3D visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *The International Journal of Robotics Research (IJRR)*, 31(4):498–519, 2012.

[7] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research (IJRR)*, 30(10):1284–1306, 2011.

[8] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. A novel representation of parts for accurate 3D object detection and tracking in monocular images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4391–4399, 2015.

[9] Randal Douc and Olivier Cappé. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pages 64–69, 2005.

[10] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-Blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 176–183, 2000.

[11] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(5):876–888, 2012.

[12] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conference on Computer Vision (ACCV)*, pages 548–562, 2012.

[13] A Hodan, J Matas, and S. Obdrzalek. On evaluation of 6d object pose estimation. In *European Conference on Computer Vision (ECCV)*, 2016.

[14] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888, 2017.

[15] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making rgb-based 3d detection and 6D pose estimation great again. In *International Conference on Computer Vision (ICCV)*, pages 22–29, 2017.

[16] Zia Khan, Tucker Balch, and Frank Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(11):1805–1819, 2005.

[17] Alexander Krull, Frank Michel, Eric Brachmann, Stefan Gumhold, Stephan Ihrke, and Carsten Rother. 6-DOF model based tracking via object coordinate regression. In *Asian Conference on Computer Vision (ACCV)*, pages 384–399, 2014.

[18] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 954–962, 2015.

[19] C.T. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276, 2004.

[20] Shile Li, Seongyong Koo, and Dongheui Lee. Real-time and model-free object tracking using particle filter with joint color-spatial descriptor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6079–6085, 2015.

[21] Jörg Liebelt, Cordelia Schmid, and Klaus Schertler. Independent object class detection using 3D feature maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[23] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He,

and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[25] F Landis Markley, Yang Cheng, John Lucas Crassidis, and Yaakov Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007.

[26] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. 2002.

[27] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. Object tracking with an adaptive color-based particle filter. In *Joint Pattern Recognition Symposium*, pages 353–360, 2002.

[28] Yuki Oka, Toshiyuki Kuroda, Tsuyoshi Migita, and Takeshi Shakunaga. Tracking 3D pose of rigid object by sparse template matching. In *International Conference on Image and Graphics*, pages 390–397, 2009.

[29] Karl Pauwels, Leonardo Rubio, Javier Diaz, and Eduardo Ros. Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2347–2354, 2013.

[30] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-DOF object pose from semantic keypoints. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2011–2018, 2017.

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[32] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision (IJCV)*, 66(3):231–259, 2006.

[33] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Rao-Blackwellized particle filter for multiple target tracking. *Information Fusion*, 8(1):2–15, 2007.

[34] Caifeng Shan, Yucheng Wei, Tieniu Tan, and Frédéric Ojardias. Real time hand tracking by combining particle filtering and mean shift. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 669–674, 2004.

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

[36] R.A. Srivatsan, M. Xu, N. Zevallos, and H. Choset. Bingham distribution-based linear filter for online pose estimation. *Robotics: Science and Systems (RSS)*, 2017.

[37] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3D orientation learning for 6D object detection from RGB images. In *European Conference on Computer Vision (ECCV)*, pages 712–729, 2018.

[38] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6D object pose prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 292–301, 2018.

[39] S Thrun, W Burgard, and D Fox. *Probabilistic Robotics*. MIT Press, 2005.

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

[41] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. 2019.

[42] Yu Xiang, Changkyu Song, Roozbeh Mottaghi, and Silvio Savarese. Monocular multiview object tracking with 3d aspect parts. In *European Conference on Computer Vision (ECCV)*, pages 220–235, 2014.

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

[44] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1386–1383, 2017.