Self-supervised 6D Object Pose Estimation for Robot Manipulation

Xinke Deng[1,2], Yu Xiang[1], Arsalan Mousavian[1], Clemens Eppner[1], Timothy Bretl[2], Dieter Fox[1,3]

Abstract— To teach robots skills, it is crucial to obtain data with supervision. Since annotating real world data is time-consuming and expensive, enabling robots to learn in a self-supervised way is important. In this work, we introduce a robot system for self-supervised 6D object pose estimation. Starting from modules trained in simulation, our system is able to label real world images with accurate 6D object poses for self-supervised learning. In addition, the robot interacts with objects in the environment to change the object configuration by grasping or pushing objects. In this way, our system is able to continuously collect data and improve its pose estimation modules. We show that the self-supervised learning improves object segmentation and 6D pose estimation performance, and consequently enables the system to grasp objects more reliably. A video showing the experiments can be found at https://youtu.be/W1Y0Mmh1Gd8.

I. INTRODUCTION

For robots to gain skills such as manipulation, learning-based methods have received more attention recently due to their capability in handling the complexity in robot perception. For instance, a number of 6D object pose estimation methods using deep neural networks are introduced for model-based grasping [1], [2]. Learning techniques also demonstrate impressive performance in grasping unknown objects [3]–[5]. It is well known that training deep neural networks requires a significant amount of data. How to obtain training data becomes the bottleneck in applying learning techniques to different robotic problems these days.

It is very appealing to conduct training in simulation, since synthetic data generated from simulators are free, come with ground truth annotations, and are easy to scale up. However, due to the domain gap, models trained exclusively with synthetic data cannot be guaranteed to work well in the real world. In contrast, training can be performed using data directly collected in the real world. However, annotating real world data is time consuming and labor intensive.

Recently, the concept of self-supervised learning is becoming attractive, where robots autonomously annotate real world data for training. Self-supervised learning has great potential to improve the robustness of robotic systems and achieve life-long learning for robots. The main challenge in self-supervision is how to annotate the real world data automatically, and meanwhile, guarantee the accuracy of the annotations. Previous works have explored self-supervised learning for model-free grasping [4], pixel-wise object segmentation [6], and object detection with bounding boxes [7]. However, to the best of our knowledge, no work has been
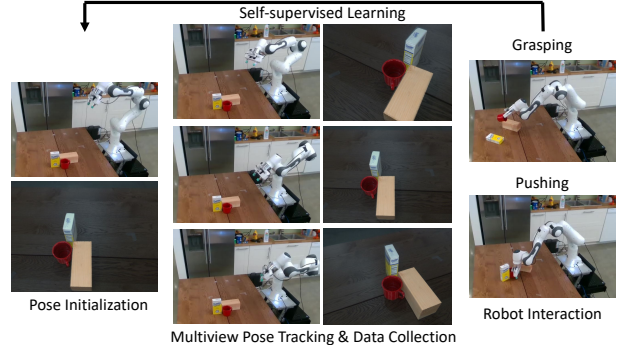
Fig. 1: Our self-supervised learning process for robot manipulation.

proposed for directly annotating 6D object pose in a self-supervised fashion. This is mainly because accurately estimating 6D object pose and automatically evaluating the estimation is challenging.

In this work, by leveraging the recent advances in 6D object pose estimation with deep neural networks [1], [8], [9], we present a novel self-supervised 6D object pose estimation system for manipulation (Fig. 1). An active vision system is set up by mounting an RGB-D camera onto the hand of a robot manipulator. Given a set of objects in the workspace, we control the robot arm to collect images from different views. Meanwhile, the 6D poses of these objects are accurately initialized and then tracked based on the PoseRBPF framework [9], where we use the forward kinematics of the robot arm as a motion prior in the particle filtering. Consequently, our system is able to collect images of multiple objects from different views with accurate 6D pose annotations. We can use this data to perform self-supervised training of the deep neural networks in the system for object segmentation and pose estimation. After each iteration of the multi-view pose tracking, the robot manipulator grasps or pushes one object in the scene to generate a new scene, and then repeats the pose tracking process. In this way, our system is able to continuously generate new data for self-supervised learning, and improve its pose estimation and grasping performance in a life-long learning fashion.

We conduct experiments to demonstrate how the self-supervised learning improves the system. We first show that by using the self-annotated images to fine-tune the semantic segmentation network, we can significantly improve the segmentation accuracy compared to the original network trained with synthetic data only. Semantic segmentation suffers from the domain gap severely for objects with complex material properties, such as shininess. Fine-tuning the network can bridge the domain gap. Second, we show that fine-tuning the auto-encoders used in PoseRBPF [9] can improve their
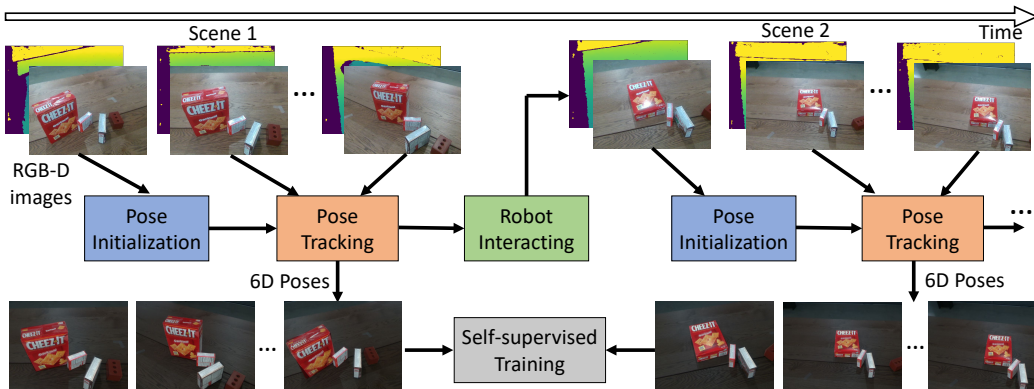
Fig. 2: Overview of our self-supervised 6D object pose estimation system.

reconstruction quality, and consequently improve the pose estimation accuracy. Finally, we conduct robot grasping experiments to illustrate that our fine-tuned system can achieve a high success rate in model-based grasping based on 6D object pose estimation.

This paper is organized as follows: After discussing related work, we introduce our system for self-supervised 6D object pose estimation, followed by experiments and a conclusion.

II. RELATED WORK

6D Object Pose Estimation. Given the 3D models of objects, the goal of 6D pose estimation is to estimate the 3D translation and the 3D rotation of these objects from input images. Traditionally, the problem is tackled by local feature matching [10]–[12] or template matching [13]–[15]. Recently, learning-based methods receive more attention due to their ability to generalize to different backgrounds, lighting, and occlusions [16]–[18]. Deep neural networks further boost the performance by learning more discriminative features and classifiers [1], [2], [8], [9], [19]. However, deep learning methods also require more data for training.

A few recent methods show that it is possible to train exclusively with synthetic data and apply the trained network to the real world [8], [20]. But for objects with mismatch between simulation and real world, these methods tend to fail, such as mismatch in texture materials or lighting. Meanwhile, some efforts are devoted to collecting large-scale real-world dataset for 6D pose estimation [1], [21]. These methods use a moving camera around a static scene so that the pose labels for a video sequence can be generated with labeling one frame and tracking the camera ego-motion [1] or labeling the 3D reconstructed scene [21]. Although tracking camera motion simplifies the labeling process, these methods require human to arrange the scene, collect video sequences of objects, and initialize the systems, which is difficult to scale to a large number of scenes. In this work, we employ robots to automatically arrange scenes and generate accurate 6D pose annotations without pose initialization from human.

Self-supervised Learning. Due to its ability to learn from unlabeled data, self-supervised learning has been studied in different sub-fields in AI, such as in robotics [4], [22], computer vision [23], [24], machine learning [25] and natural language processing [26]. The majority of these methods

focus on learning from fixed datasets. In robotics, since the robot can interact with the real world, we can plan and control how data is collected [27], [28]. This degree of freedom in data collection significantly boosts the self-supervised learning efficiency.

Previous works on self-supervised learning in robot perception mainly focus on learning object segmentation [6], object detection [7], and dense pixel-wise correspondences [29], [30], since it is relatively easy to obtain annotations in 2D image space. In this work, we explore self-supervised learning for 6D object pose estimation. Our system benefits from the recent advances in 6D object pose estimation using deep neural networks [1], [8], [9]. By incorporating these techniques into the interactive robot perception framework [31], we can obtain accurate 6D poses of objects to improve the system.

III. SELF-SUPERVISED 6D OBJECT POSE ESTIMATION SYSTEM

A. System Overview

In our system, a RGB-D camera is mounted to the hand of a robot manipulator, where images from different views can be collected by moving the robot arm (Fig. 1). The system consists of a pose initialization module, a pose tracking module, a robot interaction module and a self-supervised training module (Fig. 2). 1) Given a set of objects in the robot's workspace, the pose initialization module estimates the 6D poses of all objects from a single RGB-D image. To ensure accurate pose initialization, we evaluate the estimated poses and re-initialize the unsatisfied objects until all the poses are accurate. 2) Then the pose tracking module tracks the 6D poses of these objects when the robot arm moves to different predefined way points. This step also collects images with 6D object poses from different views, and saves the data in an online fashion. Both the pose initialization and pose tracking are based on the PoseRBPF framework using particle filtering [9]. 3) After the data is saved, the robot interaction module controls the manipulator to either grasp or push one object in the scene, thereby generating a new scene. In this way, we can continuously collect data with different arrangements of objects. 4) The self-supervised training module fine-tunes the deep neural networks in the system with the saved data to boost its performance.
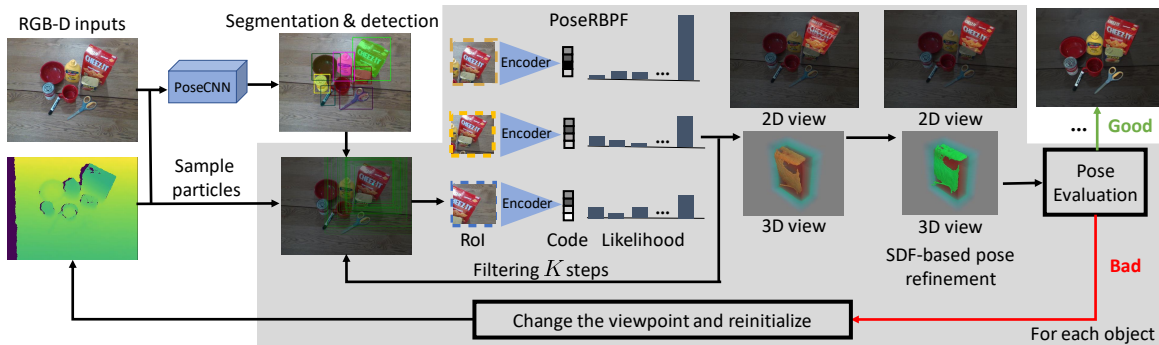
Fig. 3: Pose initialization process. This module takes an RGB-D image as input and estimates the 6D poses of all the objects in the image. The process iterates until all object poses are accurate according to the pose evaluation component.

B. The Pose Initialization Module

Given a set of objects in a scene, the task of the pose initialization module is to accurately estimate the 6D poses of all objects from a single RGB-D image. Based on this estimate we can start pose tracking (Fig. 3). Pose estimation from a single image is a challenging problem. With a new scene, the camera is moved automatically to a viewpoint with high elevation angle, so the pose estimation is less affected by the occlusions between objects.

1) Object Segmentation and Detection: The first step is to detect the objects in the input image. In general, any state-of-the-art object detector can be used, such as YOLO [32], SSD [33] or Faster R-CNN [34]. Instead of using these top-down methods that are built on bounding boxes, we resort to PoseCNN [1]. PoseCNN first labels every pixel into an object class and then applies Hough voting to find the centers and bounding boxes of the objects. PoseCNN generates segmentation masks of the objects which can be used to segment the point clouds of the objects in pose refinement [35]. In addition, the object centers from PoseCNN are more accurate than bounding box centers from the top-down methods thanks to the Hough voting strategy.

2) 3D Rotation and 3D Translation Estimation: To estimate the 3D rotation $\mathbf{R}$ and 3D translation $\mathbf{t}$ of objects in the scene, we utilize the PoseRBPF framework [9]. PoseRBPF is a Rao-Blackwellized particle filter combined with a learned auto-encoder network [8] for 6D object pose estimation. In PoseRBPF, each particle consists of a 3D translation and a full distribution over $\mathbf{SO}(3)$. A 3D translation can be represented by (u, v, z), where (u, v) is the object center in the image and z is the distance of object center from the camera in 3D.

For each detected object, the 3D translations of the particles in PoseRBPF are sampled according to the detected object center $(\hat{u}, \hat{v})$ and the observed depth image $\mathbf{D}$: $(u_i, v_i) \sim \mathcal{N}((\hat{u}, \hat{v}), \sigma_u, \sigma_v)$, $z_i \sim \mathcal{U}(\mathbf{D}(u_i, v_i) - \frac{d}{2}, \mathbf{D}(u_i, v_i) + \frac{d}{2})$, where (u_i, v_i, z_i) denotes the translation of the ith particle, σ_u, σ_v denote the standard deviations of the Gaussian distributions for object center, and d denotes the range of a uniform distribution where z_i is sampled from. $\mathbf{D}(u_i, v_i)$ is the depth at pixel location (u_i, v_i). Each particle determines a Region of Interest (RoI) of the target object (Fig. 3). The RoI is fed into the auto-encoder to compute a code which is used

to measure the likelihood of the particle in PoseRBPF. After the initial sampling of the particles, the system performs K filtering steps with the same RGB-D image to ensure the convergence of particles, from which we can extract the expectation of 3D rotation and 3D translation.

3) 6D Pose Refinement: Since the number of particles in PoseRBPF is limited due to time budget and discretization of 3D rotation in each particle, we perform a continuous optimization using 3D points from the depth image to refine the poses. Finding correct pixels corresponding to the object is crucial in the refinement process. We estimate the segmentation mask of the object $\mathbf{\Omega}$ by considering both the predicted segmentation mask $\hat{\mathbf{\Omega}}$ from PoseCNN and the predicted 6D pose $(\bar{\mathbf{t}}, \bar{\mathbf{R}})$ from PoseRBPF. We first render the object according to the estimated pose. Denoting the rendered depth image and segmentation mask of the object as $\bar{\mathbf{D}}$ and $\bar{\mathbf{\Omega}}$. The final segmentation mask $\mathbf{\Omega}_{\text{obj}}$ is estimated with $\mathbf{\Omega}_{\text{obj}} = \hat{\mathbf{\Omega}} \cap \bar{\mathbf{\Omega}} \cap \mathbf{\Omega}_{\mathbf{D}}$. Here, $\mathbf{\Omega}_{\mathbf{D}}$ represents the pixels where the rendered depth matches with the measured depth within a margin m: $\mathbf{\Omega}_{\mathbf{D}} = \{\forall(u, v), |\mathbf{D}(u, v) - \bar{\mathbf{D}}(u, v)| < m\}$. Then the point cloud of the object $\mathbf{P}_{\text{obj}}$ can be computed by back-projecting the pixels in $\mathbf{\Omega}_{\mathbf{D}}$:

$$\mathbf{P}_{\text{obj}} = \{\mathbf{D}(u, v)\mathbf{K}^{-1}(u, v, 1)^T, (u, v) \in \mathbf{\Omega}_{obj}\}, \quad (1)$$

where $\mathbf{K}$ represents the intrinsic matrix of the camera.

After computing the 3D points on the object, we optimize the pose by matching these points against the Signed Distance Function (SDF) of the object model as in [36]. The optimization problem we solve is

$$(\hat{\mathbf{t}}, \hat{\mathbf{R}}) = \arg\min_{\mathbf{t}, \mathbf{R}} \sum_{\mathbf{p}_i \in \mathbf{P}_{\text{obj}}} |\mathbf{SDF}_{\text{obj}}(\mathbf{p}_i, \mathbf{t}, \mathbf{R})| + \lambda \frac{1}{2}\|\mathbf{t} - \bar{\mathbf{t}}\|^2,$$
$$(2)$$

where $\mathbf{p}_i$ is a 3D point in the point cloud $\mathbf{P}_{\text{obj}}$, $\mathbf{SDF}_{\text{obj}}(\mathbf{p}_i, \mathbf{t}, \mathbf{R})$ denotes the signed distance value by transforming the point $\mathbf{p}_i$ from the camera coordinate into the object model coordinate using pose $(\mathbf{t}, \mathbf{R})$, and λ is a weight to balance the regularization term on the translation. Fig. 3 illustrates the matching between the point cloud against the SDF before and after the optimization.

4) Pose Evaluation: Due to noise in both RGB and depth measurements, failure in pose estimation is inevitable. To detect failures and label object poses accurately, we propose a pose evaluation process (Fig. 4) so that objects
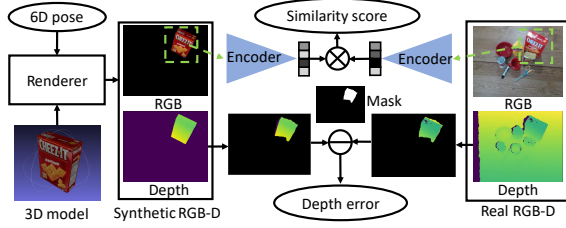
Fig. 4: The pose evaluation component compares the rendered RGB-D image with the real RGB-D image.

Fig. 5: Examples of 100 precomputed grasps for 10 YCB objects.

with erroneous pose estimation will be reinitialized. We first render an object according to its estimated pose to generate a synthetic RGB-D image. Then we compute two metrics by comparing the synthetic image with the real image. For RGB, we compute the code of the color rendering $\hat{\mathbf{c}}$ and the code of the color measurement $\mathbf{c}$ by passing the two corresponding RoIs through the auto-encoder in PoseRBPF. The cosine distance $s = (\hat{\mathbf{c}} \cdot \mathbf{c})/(\|\hat{\mathbf{c}}\| \cdot \|\mathbf{c}\|)$ between the two codes is then computed to indicate the pose confidence in the color space. For depth, we compare the rendered depth image $\hat{\mathbf{D}}$ and the measured depth image $\mathbf{D}$ using the mask $\mathbf{\Omega}_{\text{obj}}$ from Eq. (1). The mean depth error is computed to measure the pose accuracy in the depth space: $e = \sum_{(u,v) \in \mathbf{\Omega}_{\text{obj}}} |\hat{\mathbf{D}}(u,v) - \mathbf{D}(u,v)|$. Finally, we declare pose estimation failure if $s < s^*$ or $e > e^*$, where s^* and e^* are predefined thresholds. In this case the system re-initializes the pose of the failed object.

C. The Pose Tracking Module

Once object poses in the scene are initialized, the robot moves the camera to different views around the objects for data collection. Meanwhile, a continuous sequence of video frames is collected in real time, and the poses of the objects in the sequence are tracked with the PoseRBPF framework. The pose tracking module generates accurate pose annotations online for self-supervised training and enables the robot to interact with the objects continuously.

In order to track all objects accurately, we first propagate all particles according to the robot's forward kinematics. Denoting the camera pose transformation from the forward kinematics as $(\Delta\mathbf{t}, \Delta\mathbf{R}) \in \mathbf{SE}(3)$, the particles are propagated according to the following motion prior:

$$P(\mathbf{t}_k|\mathbf{t}_{k-1}, \Delta\mathbf{t}, \Delta\mathbf{R}) = \mathcal{N}(\Delta\mathbf{R}\mathbf{t}_{k-1} + \Delta\mathbf{t}, \Sigma_\mathbf{t}), \quad (3)$$

$$P(\mathbf{R}_k|\mathbf{R}_{k-1}, \Delta\mathbf{t}, \Delta\mathbf{R}) = \mathcal{N}(\Delta\mathbf{R}\mathbf{R}_{k-1}, \Sigma_\mathbf{R}), \quad (4)$$

where $\Sigma_\mathbf{t}$ and $\Sigma_\mathbf{R}$ denote the covariance matrix of the Gaussian distributions of the translation and the rotation, respectively. Since the rotation in PoseRBPF is represented as a *full discrete distribution* over $\mathbf{SO}(3)$, the distributions need to be shifted entirely. We first represent $\Delta\mathbf{R}$ as $(\Delta\phi, \Delta\theta, \Delta\psi)$ according to the discretization of the rotation distribution, where $\Delta\phi$, $\Delta\theta$, $\Delta\psi$ represent the changes in pitch, yaw and roll, respectively. Then the rotation distribution is adjusted by performing bilinear interpolation of the shifted grids $\{\phi + \Delta\phi, \theta + \Delta\theta, \psi + \Delta\psi\}$.

Due to the noises from the joint encoders of the robot or the bias in the hand-eye calibration, simply applying forward
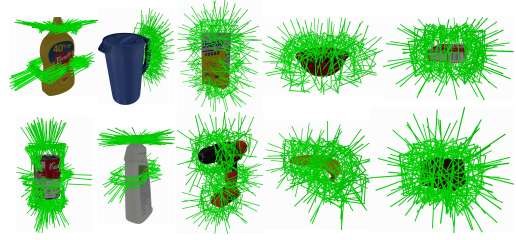
kinematics might often lead to inaccurate object poses. Similar to the pose initialization process, we perform 1 step filtering with the propagated particles using the current RGB-D image for all the objects. To ensure the quality of the generated data for self-supervised learning, we only save data when the two pose evaluation metrics are above certain thresholds (Sec. III-B.4). Once the system decides to save the data, the object poses are further refined with the SDF-based pose refinement algorithm as described in Sec. III-B.3.

D. The Robot Interaction Module

After a scene is captured, the robot physically interacts with objects in the scene to change their configuration. This way the system can continuously capture new data with different poses of the objects. Two types of interactions are used. The robot either pushes an object in a random direction or grasps an object and places it at a different location (Fig. 1).

Pushing is defined by choosing a direction and random radius with respect to the object center and executing the motion. To ensure that objects stay in the workspace of the robot, objects are pushed either toward the center of mass of other objects in the clutter or the center of the robot workspace. For grasping, a set of diverse grasps for each object is computed offline, where we use 100 parallel-jaw grasps sampled from the robust grasps of [37] (see Fig. 5 for examples). During online execution, using the estimated pose of the object, the pre-computed grasps are transformed from object to camera coordinates. Then the robot selects any grasp that is kinematically feasible and collision-free with other objects using the RRT* planning algorithm in MoveIt! [38]. After grasping the object, the robot first rotates its gripper to change the orientation of the object. Then a random placement location is chosen within the robot workspace and the robot places the object.

E. The Self-Supervised Training Module

With the collected images and estimated object poses, we can fine-tune the neural networks in our system to improve the pose estimation performance. Initially, both the network for segmentation and detection in PoseCNN and the auto-encoders used in PoseRBPF (see Fig. 3) are trained only with synthetic data. As a result, the system cannot even segment some textureless objects in our experiments such as a red foam brick. Therefore, we employ the curriculum learning idea [39] to bootstrap the system sequentially. We start with simple scenes that contain only one object. If the network cannot segment the object, we perform a global initialization

TABLE I: F1 score of semantic segmentation for 20 YCB objects on the test set. The best F1 score for each object are bold, and numbers below 60% are highlighted red.

Model	Synth.	+20% Real	+40% Real	+60% Real	+80% Real	+100% Real
master_chef_can	69.3	88.7	92.8	91.9	91.6	**93.9**
cracker_box	84.7	92.8	93.0	**93.4**	93.0	**93.4**
sugar_box	83.0	92.0	92.0	92.4	92.5	**92.6**
tomato_soup_can	83.6	90.2	90.5	90.8	91.2	**91.4**
mustard_bottle	83.9	92.5	93.3	93.3	93.9	**94.2**
tuna_fish_can	42.3	90.1	90.2	**91.7**	**91.7**	91.6
pudding_box	61.6	85.7	84.6	85.9	**87.1**	87.0
gelatin_box	66.6	83.6	83.2	83.7	82.0	**84.6**
potted_meat_can	62.9	84.1	85.4	86.4	86.6	**88.6**
banana	79.8	87.3	88.2	89.0	89.0	**89.3**
pitcher_base	51.5	86.3	84.3	88.0	**89.7**	89.6
bleach_cleanser	57.9	89.3	92.1	93.3	90.2	**93.4**
bowl	69.8	90.4	92.5	93.2	94.5	**95.4**
mug	69.2	90.3	90.9	91.4	**92.0**	91.0
power_drill	66.1	84.4	87.3	88.0	87.5	**88.5**
wood_block	64.2	82.6	80.2	85.1	86.0	86.1
scissors	36.3	71.9	75.8	77.4	77.5	**78.9**
large_marker	55.5	73.8	75.6	76.2	75.4	**77.1**
extra_large_clamp	15.5	76.3	76.0	**79.1**	77.1	**79.1**
foam_brick	12.2	86.5	86.7	87.6	86.6	**88.9**
MEAN	60.8	85.9	86.7	87.9	87.8	**88.7**

in PoseRBPF, where particles are uniformly sampled in the image. This process collects training data with single objects in images. We can then fine-tune the segmentation network to make it segment all objects. Fine-tuning with single objects also enables the network to segment objects in cluttered scenes. In the next stage, we move to cluttered scenes with multiple objects, and collect training data with different object poses and occlusions to fine-tune the segmentation network and the auto-encoders. During fine-tuning, we mix synthetic and real data to prevent the networks from overfitting. We can iterate between data collection and fine-tuning. Because when the networks become better, the system is more efficient in collecting new data.

IV. EXPERIMENTS

A. Data Collection

We conduct experiments with 20 objects in the YCB Object and Model Set [40]. During around 12 robot hours, our system collected 497 scenes, 6,541 RGB-D images and 22,851 object instances in these images with accurate 6D poses. In this process, human intervention is required in two cases: 1) whenever a new subset of the 20 YCB objects needs to be presented to the robot, and 2) to rearrange objects if the system cannot initialize the object poses for a while. Compared to human collected videos for pose annotation, our system saves a significant amount of human labor, and has the ability to scale up to a large number of scenes. For comparison, the YCB-Video dataset [1] and the LabelFusion dataset [21] have 92 scenes and 138 scenes, respectively. We split the collected data into a training set (265 scenes, 3,590 images) and a test set (232 scenes, 2,951 images).

B. Evaluation Metrics

We evaluate how the system improves by fine-tuning the neural networks with the collected data. For semantic segmentation, we compute the F1 score by comparing the
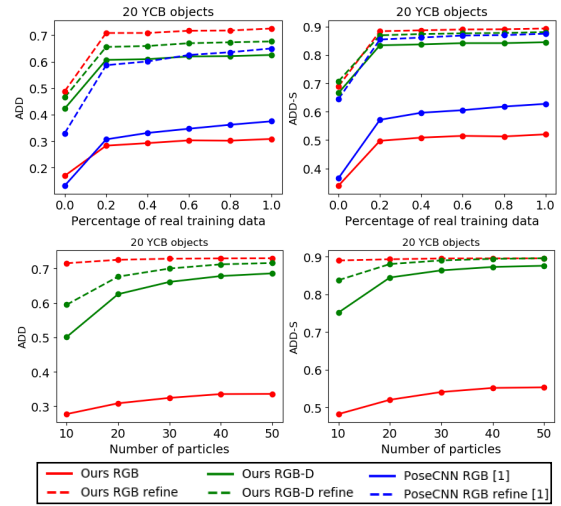

Fig. 6: Evaluation of 6D object pose estimation on the test set.

predicted labels with the ground truth labels pixel-wisely. For evaluating 6D poses, we use ADD and ADD-S metric as defined in [1], [14]: $\text{ADD} = \frac{1}{m}\sum_{\mathbf{x}\in\mathcal{M}}\|(\mathbf{R}\mathbf{x}+\mathbf{t})-(\tilde{\mathbf{R}}\mathbf{x}+\tilde{\mathbf{t}})\|$, $\text{ADD-S} = \frac{1}{m}\sum_{\mathbf{x}_1\in\mathcal{M}}\min_{\mathbf{x}_2\in\mathcal{M}}\|(\mathbf{R}\mathbf{x}_1+\mathbf{t})-(\tilde{\mathbf{R}}\mathbf{x}_2+\tilde{\mathbf{t}})\|$, where $\mathcal{M}$ denotes the set of 3D model points and m is the number of points. $(\mathbf{R},\mathbf{t})$ and $(\tilde{\mathbf{R}},\tilde{\mathbf{t}})$ are the ground truth pose and estimated pose, respectively. To evaluate the grasping performance, we compute the success rate, and the speed for initializing pose estimations as well as grasping.

C. Implementation Details

We implement our system on a 7-DoF Franka Panda manipulator with an Intel RealSense D415 camera mounted on its parallel-jaw gripper. The deep neural networks run on two NVIDIA Titan Xp GPUs. In the pose initialization module, the noise for sampling particles σ_u, σ_v, and σ_z are set to $20\,\text{pixels}$, $20\,\text{pixels}$, and $0.1\,\text{m}$ respectively. In pose refinement, the margin for computing the visibility mask is set to $0.02\,\text{m}$, and the weight λ is set to 0.001 In pose evaluation, thresholds for deciding estimation failures are 0.5 and $0.03\,\text{m}$ for s^* and e^*. In pose tracking module, process noise $\Sigma_{\mathbf{t}}$ and $\Sigma_{\mathbf{R}}$ are set to $0.015\,\text{m}$ and $0.05\,\text{rad}$.

D. Semantic Segmentation

By fine-tuning the segmentation network with self-collected images, we can bridge the domain gap and significantly improve the segmentation accuracy. Table I shows the F1 scores of 20 YCB objects on the test set. The performance of the networks fine-tuned with different percentages of real training data is presented. By using $20\,\%$ training data, the overall F1 score is increased by $25\,\%$. Fine-tuning with more data consistently improves the performance. In Table I, we highlight objects on which synthetic training performs poorly. Most of them are textureless (pitcher_base, wood_block, extra_large_clamp and foam_brick). Objects with metal parts generate reflections which are not well modeled in simulation (tuna_fish_can, potted_meat_can and scissors). For pudding_box and power_drill, the textures of the real objects are different from the 3D models due to
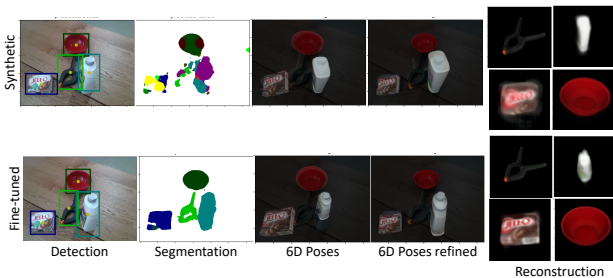
Fig. 7: An example where synthetic networks fail in segmentation and pose estimation, but the fine-tuned networks succeed.

product updates. large_marker is difficult since it is small in images. All these issues make synthetic training difficult to transfer to the real world, and our self-supervised learning system successfully adapts to the new environment.

E. 6D Object Pose Estimation

In this experiment, we evaluate 6D object pose estimation using single RGB-D images as input. If the system can estimate object poses quickly and accurately from just a single image, it would significantly speed up manipulation tasks. We evaluate our pose initialization method (Fig. 3) and the original PoseCNN [1]. Both methods use the same network for segmentation and detection. The main difference is in rotation and translation estimation. PoseCNN [1] is trained to directly regress to object distance and 3D rotation, while our method uses the auto-encoders and particle filter to estimate translation and rotation.

Fig. 6 presents the 6D pose estimation accuracy in terms of ADD and ADD-S using different percentages of real images for fine-tuning or number of particles. We can see that 1) both methods improve consistently with more real training data, and the gap between synthetic models and fine-tuned ones is obvious. 2) Using depth is important to achieve high accuracy, and our SDF-based pose refinement algorithm significantly boosts the accuracy. 3) The performance of our method improves consistently with more particles in PoseRBPF. However the improvement becomes less obvious when performing refinement. 4) It is interesting that our RGB based particle filter with SDF refinement achieves the best performance, which is even better than the RGB-D particle filter with refinement. This is because the auto-encoders are trained only with RGB images. Using RGB images in the particle filter obtains more accurate rotation, and then SDF refinement can fix errors in translation, and adjust rotation locally. Fig 7 shows an example on how the fine-tuned segmentation network and autoencoders improve segmentation and pose estimation.

F. Real Robot Grasping Experiments

We evaluate the performance of our system on a pick-and-place task. The goal is to pick up all objects in the scene and place them into a bin. The distance between camera and clutter is around $0.5\,\mathrm{m}$ and the initial elevation angle of the camera is $55°$. The system first initializes all objects in the scene. Every time before grasping the target object, all
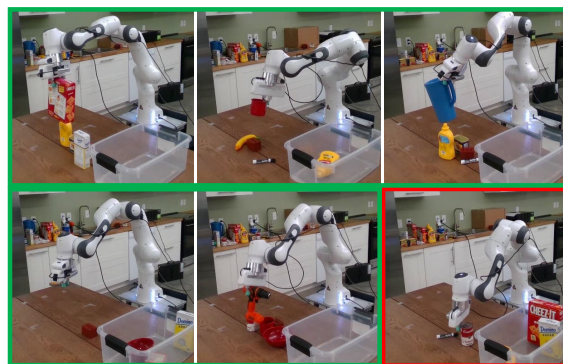

Fig. 8: Grasping performed in real cluttered environments. Here we show 5 successful and 1 failed trials on grasping YCB objects.

TABLE II: Statistics for 30 grasping trials in clutter.

Grasp Trials	Success rate [%]	Avg. initialization time per object [s]	Avg. grasp duration [s]
synthetic data only	46.70	10.04 (std: 15.98)	21.27
synthetic + real data	**86.70**	**4.48** (std: 5.71)	**15.47**

objects are evaluated and those with a detected failure are re-initialized based on the target object. The estimated poses are used for grasping the objects. After picking up an object, the robot follows a predefined trajectory to place it.

We created six different cluttered environments. Every environment consists of five different YCB objects. In total, 14 graspable YCB objects are selected, and 30 grasping trials are executed. Grasping examples are shown in Fig. 8. We evaluate three metrics: 1) grasp success rate, 2) average time elapsed for accurately initializing each object, and 3) average time elapsed for initializing and grasping each object. The results are shown in Table II. It is possible that the pose of the objects cannot be successfully initialized. We allow $45\,\mathrm{s}$ for initializing each object before calling timeout and labelling the trial as a failure. As we see, by fine-tuning with additional self-annotated data, the system's performance on grasping can be significantly improved: the success rate is improved by $85.7\,\%$, the duration for initialization is reduced by $55.3\,\%$, the overall duration for grasping an object is reduced by $27.3\,\%$. There were four cases when grasps failed. Three of them were due to errors in pose estimations; in the other one MoveIt! failed to generate a feasible trajectory.

V. CONCLUSION

We introduce a novel self-supervised 6D object pose estimation system for robot manipulation. Our system is able to automatically collect real world images with accurate 6D object poses to fine-tune its neural networks for pose estimation. In the learning stage, the robot interacts with objects in a scene by grasping or pushing them to create new scenes, so new data can be collected continuously. We show that, starting from neural networks trained with synthetic images only, our system consistently improves the pose estimation accuracy through self-supervised learning. The grasping success rate and grasping time are also significantly improved. For future work, we plan to investigate how to improve grasp selection and grasp execution with self-supervised learning in our system.

REFERENCES

[1] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Robotics: Science and Systems (RSS)*, 2018.

[2] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6D object pose estimation by iterative dense fusion," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3343–3352.

[3] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems (RSS)*, 2017.

[4] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[5] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational grasp generation for object manipulation," in *International Conference on Computer Vision (ICCV)*, 2019.

[6] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. 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)*. IEEE, 2017, pp. 1386–1383.

[7] C. Mitash, K. E. Bekris, and A. Boularias, "A self-supervised learning system for object detection using physics simulation and multi-view pose estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 545–551.

[8] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D orientation learning for 6d object detection from RGB images," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.

[9] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "PoseRBPF: A rao-blackwellized particle filter for 6D object pose tracking," in *Robotics: Science and Systems (RSS)*, 2019.

[10] D. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision (ICCV)*, 1999.

[11] 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," *International Journal of Computer Vision (IJCV)*, vol. 66, no. 3, pp. 231–259, 2006.

[12] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research (IJRR)*, vol. 30, no. 10, pp. 1284–1306, 2011.

[13] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 34, no. 5, pp. 876–888, 2011.

[14] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Asian Conference on Computer Vision (ACCV)*. Springer, 2012, pp. 548–562.

[15] Z. Cao, Y. Sheikh, and N. K. Banerjee, "Real-time scalable 6DOF pose estimation for textureless objects," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2441–2448.

[16] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 536–551.

[17] L. Bo, X. Ren, and D. Fox, "Learning hierarchical sparse features for RGB-(D) object recognition," *The International Journal of Robotics Research (IJRR)*, vol. 33, no. 4, pp. 581–599, 2014.

[18] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 954–962.

[19] 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 *International Conference on Computer Vision (ICCV)*, 2017, pp. 1521–1529.

[20] 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 *Conference on Robot Learning (CoRL)*, 2018.

[21] 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 *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[22] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 5074–5082.

[23] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1422–1430.

[24] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 577–593.

[25] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *International Conference on Machine Learning (ICML)*, 2007, pp. 759–766.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[27] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2786–2793.

[28] D. Pathak, Y. Shentu, D. Chen, P. Agrawal, T. Darrell, S. Levine, and J. Malik, "Learning instance segmentation by interaction," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2042–2045.

[29] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 420–427, 2016.

[30] P. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," in *Conference on Robot Learning (CoRL)*, 2018, pp. 373–385.

[31] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, 2017.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37.

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 91–99.

[35] J. M. Wong, V. Kee, T. Le, S. Wagner, G.-L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, and D. M. Johnson, "SegICP: Integrated deep semantic segmentation and pose estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5784–5789.

[36] T. Schmidt, R. A. Newcombe, and D. Fox, "DART: Dense articulated real-time tracking." in *Robotics: Science and Systems (RSS)*, vol. 2, no. 1. Berkeley, CA, 2014.

[37] C. Eppner, A. Mousavian, and D. Fox, "A billion ways to grasps - an evaluation of grasp sampling schemes on a dense, physics-based grasp data set," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, Hanoi, Vietnam, 2019.

[38] S. Chitta, I. Sucan, and S. Cousins, "MoveIt!" *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[39] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *International Conference on Machine Learning (ICML)*. ACM, 2009, pp. 41–48.

[40] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *International Conference on Advanced Robotics (ICAR)*. IEEE, 2015, pp. 510–517.