GRIP: Generative Robust Inference and Perception for Semantic Robot Manipulation in Adversarial Environments

Xiaotong Chen[1], Rui Chen[1], Zhiqiang Sui[1], Zhefan Ye[1], Yanqi Liu[2],
R. Iris Bahar[2] and Odest Chadwicke Jenkins[1]

***Abstract*— Recent advancements have led to a proliferation of machine learning systems used to assist humans in a wide range of tasks. However, we are still far from accurate, reliable, and resource-efficient operations of these systems. For robot perception, convolutional neural networks (CNNs) for object detection and pose estimation are recently coming into widespread use. However, neural networks are known to suffer from overfitting during the training process and are less robust under unforeseen conditions (which makes them especially vulnerable to *adversarial scenarios*). In this work, we propose *Generative Robust Inference and Perception (GRIP)* as a two-stage object detection and pose estimation system that aims to combine the relative strengths of discriminative CNNs and generative inference methods to achieve robust estimation. Our results show that a second stage of sample-based generative inference is able to recover from false object detections by CNNs, and produce robust estimations in adversarial conditions. We demonstrate the efficacy of *GRIP* robustness through comparison with state-of-the-art learning-based pose estimators and pick-and-place manipulation in dark and cluttered environments.**

I. INTRODUCTION

Taking advantage of the renaissance in deep neural networks, machine learning has achieved great progress in object detection and segmentation and image recognition. These deep learning methods are also prevalent in robotics for problems, including manipulation in clutter [1] and learning of manipulation actions [2]. For 6D object pose estimation, learning-based Convolutional Neural Networks (CNNs) have achieved promising accuracy and real-time inference speed [3], [4], [5]. Notably, these successes rely on well-designed models and adequate training resources. The robustness and generalization capability of CNNs heavily depend on the training data, which represents a certain range of conditions that could be faced by robots. However, due to the complex and dynamic nature of the real world, robots are subject to unforeseen environmental conditions, which are not present in the training data.

More specifically, CNNs recognition systems introduce vulnerability to errors (both benign and malicious) due to the effects of overfitting during the training process. Distorted objects and/or objects captured under poor lighting conditions could be enough to defeat the recognition abilities of a CNN [6]. Such perception errors can lead to (potentially disastrous) outcomes for embodied systems acting in the real world. These challenges for **robust perception** become that much more challenging when an adversary can modify the environment to exploit the vulnerabilities of a CNN. For instance, in the context of object recognition for a robotic system, a possible malicious attack (through simple modifications of an environment) has the potential to drastically alter and even manipulate a robot's final behavior. Fig. 1 shows such a robot manipulation task under dark scene.

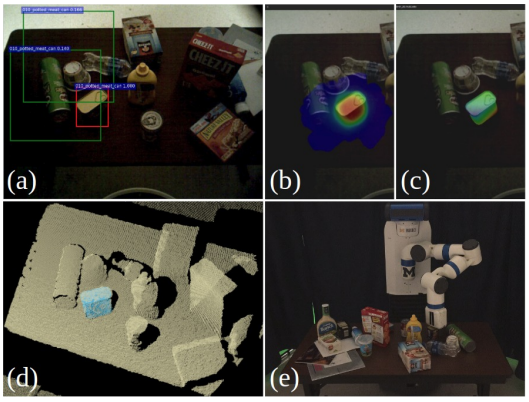


Fig. 1: Our *GRIP* system perceiving and grasping an object in adversarially darkened lighting. *GRIP* uses two stages of (a) PyramidCNN object detection bounding boxes with confidence score greater than 0.1 (green boxes), shown along with the ground truth (red box), and (b) sample-based generative inference. The (c) resulting estimate and (d) its localized pose (highlighted in cyan) enables (e) the Michigan Progress Fetch robot to accurately grasp the potted meat can object.

Generative-discriminative algorithms [7], [8] offer a promising avenue for robust perception. Such methods combine inference by deep learning (or other discriminative techniques) with sampling and probabilistic inference models to achieve robust and adaptive perception in adversarial environments. The value proposition for generative-discriminiative inference is to get the best out of existing approaches to computational perception and robotic manipulation while avoiding their shortcomings. We want the robustness of belief space planning [9], [10] without its computational intractability. The recall power of neural networks without excessive overfitting [2]. The efficiency of deterministic inference without its fragility to uncertainty [11], [12]. Generative-discriminative algorithms may be especially advantageous when exposed to adversarial attack, building on foundational ideas in this space [13], [14], [15],

[1] X. Chen, R. Chen, Z. Sui, Z. Ye and O.C. Jenkins are with the Department of Electrical Engineering and Computer Science, Robotics Institute, University of Michigan, Ann Arbor, MI, USA, 48109-2121 [cxt|richen|zsui|zhefanye|ocj]@umich.edu

[2] Y. Liu and R. Bahar are with the Department of Computer Science and the School of Engineering, Brown University, Providence, RI, USA, 02912-1910 [yanqi_liu|iris_bahar]@brown.edu

[16], [17]. Furthermore, we expect our approach will be more generally applicable to guard against broad categories of attack with a clear pathway for explanability of the resulting perceptual estimates.

In this paper, we present *Generative Robust Inference and Perception (GRIP)* as a two-stage method to explore generative-discriminiative inference for object recognition and pose estimation in adversarial environments. Within *GRIP*, we represent the first stage of inference as a CNN-based recognition distribution. The CNN recognition distribution is used within a second stage of generative multi-hypothesis optimization. This optimization is implemented as a particle filter with a static state process. We show that our *GRIP* method produces comparable and improved performance with respect to state-of-the-art pose estimation systems (PoseCNN [3] and DOPE [4]) under *adversarial scenarios* with varied lighting and cluttered occlusion. Moreover, we demonstrate the compatibility of *GRIP* with goal-directed sequential manipulation in object pick-and-place tasks with a Michigan Progress Fetch robot.

II. BACKGROUND

A. Motivation

To get the best of both worlds, we consider the state-of-the-art as the relative strengths and weaknesses of deep learning and generative inference for robust perception. We are particularly interested in complementary properties of these methods for making perceptual decisions, where the weaknesses of one can be addressed by the strengths of the other. Despite the strengths of CNNs, they have several shortcomings that leave them vulnerable to adversarial action, such as their *opacity* in understanding how its decisions are made, *fragility* for generalizing beyond overfit training examples, and *inflexibility* for recovering when false decisions are produced. For these methods, Goodfellow *et al.* [18] demonstrated that adversarial examples are misclassified both in the case of different architectures or different subsets of the training data. These weaknesses for CNNs play to the strengths of robustness for generative probabilistic inference, which are inherently: *explainable, general, and resilient* through the process of generating, evaluating, and maintaining a distribution of many hypotheses representing possible decisions. However, this robustness comes at the cost of computational efficiency. Probabilistic inference, in contrast to CNNs, is often computationally intractable with complexity that grows exponentially with the number of variables. *GRIP* aims to overcomes these limitations by combining the strengths of deep learning and probabilistic inference through a two-stage algorithm, illustrated in Fig. 2 and discussed later in Section IV. The remainder of this background section will provide a broader overview of related existing works.

B. Perception for Manipulation

Perception is a critical step for robotic manipulation in unstructured environments. Ciocarlie *et al.* [19] proposed an architecture for reliable grasping and manipulation, where non-touching, isolated objects are estimated by clustering the surface normal of RGBD sensor data. The MOPED framework [17] has been proposed for object detection and pose estimation using iterative clustering estimation from multi-view features. A bottom-up approach is taken in [20] using RANSAC and Iterative Closest Point registration (ICP), relying solely on geometric information. Narayanan *et al.* [13] integrated global search with discriminatively trained algorithms to balance robustness and efficiency, which works on multi-object identification, assuming known objects.

For manipulation in dense cluttered environments, ten Pas and Platt [21] showed success in detecting grasp affordances from 3D point clouds. In [22], they sample grasp pose candidates based on their geometric plausibility, from which feasible grasp poses are selected by a CNN. Regarding manipulation with known object geometry models, [23], [24], [25] proposed generative sampling approaches to scene estimation for object poses and physical support relations. However, these methods used object detection bounding boxes with hard thresholding as the prior for generative sampling, which might cause false negatives.

C. Object Detection and Pose Estimation

Learning-based approaches have been used as modules in object pose estimation systems, or directly built end-to-end approaches. Sui *et al.* [7] proposed a sample-based two-stage framework to sequential manipulation tasks, where object detection results are used as prior of sample initialization. Mitash *et al.* [26] developed a two-stage approach, which ran stochastic sampling of congruent sets [27] to get object poses based on the semantic map from a segmentation network. Regarding end-to-end systems, PoseCNN [3] was proposed by constructing a neural-network that learned segmentation, object 3D translation, and 3D rotation separately. This work also contributed an object dataset, called YCB-Video-Dataset, for benchmarking robotics pose estimation and manipulation approaches. DOPE [4] outperformed PoseCNN in estimation robustness in dark, occluded scenes by training the network on a synthetic dataset from domain-randomization and photo-realistic simulation. DenseFusion [5], utilized two networks to extract RGB and depth features separately.

In this paper, we focus on the pose estimation problem in *adversarial scenarios*. Liu *et al.* [8] provided insight into handling adversarial clutter, yet provided limited evaluations of its approach or comparisons with state-of-the-art methods. We believe that the performance of CNNs relies highly on the consistency of the testing environment to the training set, and that the same is true for the two-stage methods in [7] and [26] since they rely on high-quality CNN output from their first stages. Our main contribution is the development of a two-stage pose estimation system that is robust under adversarial scenarios and able to recover from false detections from its own first stage.

III. PROBLEM FORMULATION

Given an RGB-D observation (Z_r, Z_d) from the robot sensor and 3D geometry models of a known object set,

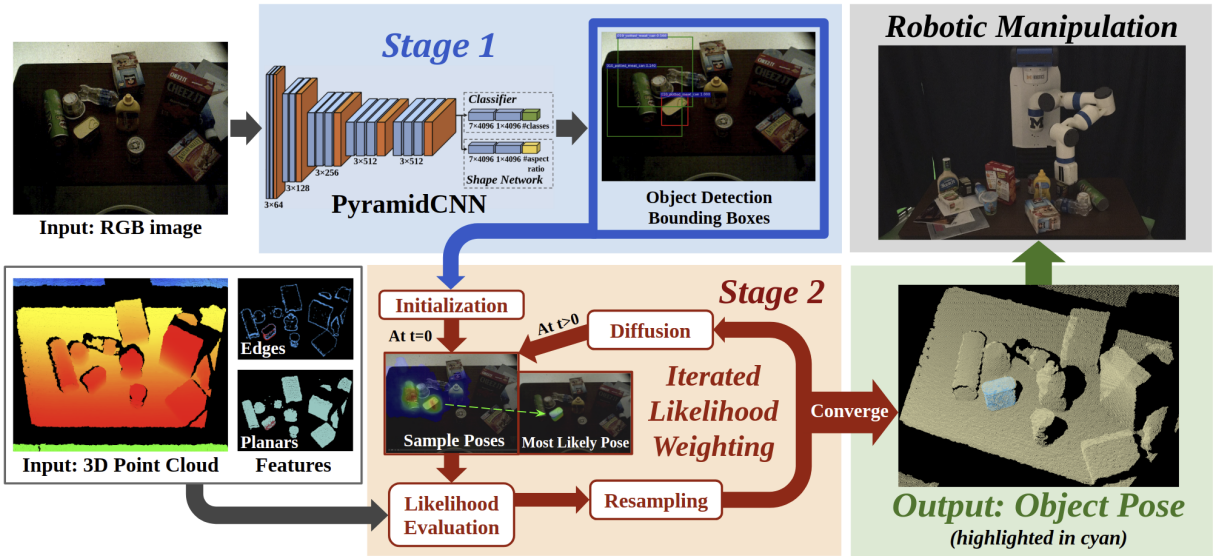


Fig. 2: Overview of *GRIP*. The robot operating in a dark and cluttered environment is to grasp the meat can from its RGBD observation. Stage 1 takes the RGB image and generates object bounding boxes with confidence scores. Stage 2 takes the depth image and performs sample-based generative inference to estimate the pose for each object in the scene. The samples in Stage 2 are initialized according to bounding boxes from Stage 1. From this estimate, the robot performs manipulation on the meat can object.

our aim is to estimate the conditional joint distribution $P(q,b|o,Z_r,Z_d)$ for each object class o, where q is the six DoF object pose and b is the object bounding box in the RGB image. The problem can be formulated as:

$$P(q,b|o,Z_r,Z_d) \tag{1}$$
$$= P(q|b,o,Z_r,Z_d)P(b|o,Z_r,Z_d) \tag{2}$$
$$= \underbrace{P(q|b,o,Z_d)}_{\text{pose estimation}}\underbrace{P(b|o,Z_r)}_{\text{detection}} \tag{3}$$

Equations (1) and (2) are derived using chain rule statistics and Equation (3) represents the factoring of object detection and pose estimation. Here, we assume that pose estimation is conditionally independent of RGB observation, while object detection is conditionally independent of depth observation.

Ideally, we could use Markov Chain Monte Carlo (MCMC) [28] to estimate the distribution of Equation (1). However, the state space of the entire states is so large that it is intractable to directly compute. End-to-end neural network methods can also be used to calculate the distribution [3], [4], [5]. These results place a heavy reliance on proper coverage of the input space in the training set. This data reliance makes such methods vulnerable to unforeseen environment changes. SUM [7] implements a combination of Equation (1) to filter over hard detections provided by a CNN, thereby enabling it to filter out false positive CNN detections. The limitation of SUM is its inability to recover from false negatives that are eliminated from consideration in object proposal and detection stages. On the other hand, our *GRIP* paradigm is able to compensate for data deficiency by employing a generative sampling method in the second stage.

IV. METHOD

We propose a two-stage paradigm to combine object detection and pose estimation, as shown in Fig. 2. In the first stage of inference, PyramidCNN performs object detection and generates a prior distribution $P(b|o,Z_r)$ of 2D bounding boxes for each object label o. In the second stage, we perform generative multi-hypothesis optimization to estimate the joint distribution $P(q,b|o,Z_{(r,d)})$ for each object label o using the first stage output as prior. The second stage is implemented as an iterated likelihood weighting filter [29]:

$$\underbrace{P(q_0,b_0|o,Z_{(r,d)})}_{\text{Sample Initialization}} = P(q_0|b_0)P(b_0|o,Z_r) \tag{4}$$

$$P(q_t,b_t|o,Z_{(r,d)}) = \eta \underbrace{P(Z_{(r,d)}|q_t,b_t,o)}_{\text{Likelihood}} \underbrace{\overline{P}(q_t,b_t|o,Z_{(r,d)})}_{\text{Proposal}} \tag{5}$$

$$\overline{P}(q_t,b_t|o,Z_{(r,d)}) = \int\int \underbrace{P(q_t,b_t|q_{t-1},b_{t-1})}_{\text{Diffusion}} \cdot P(q_{t-1},b_{t-1}|o,Z_{(r,d)})dq_{t-1}db_{t-1} \tag{6}$$

where η is the normalizing factor. In Equation (4), initial pose q_0 is generated from bounding boxes b_0, which are sampled from the prior distribution generated by the first stage. After the second stage, we get a probability distribution of pose estimation as shown in Equation (1). We consider the best estimate as the one with highest probability. Equivalently, best pose q^* satisfies,

$$q^*,b^* = \underset{q,b}{\text{argmax}}\ P(q,b|o,Z_r,Z_d) \tag{7}$$

A. Object Detection

The goal of the first stage is to provide a probability distribution map for an object class o in a given input image. To achieve this, we exploit the discriminative power of CNNs. Inspired by region proposal networks (RPN) in [30], our PyramidCNN serves as a proposal method for the second stage. We choose VGG-16 networks [31] to extract features,

which are directed to two fully convolutional networks (FCN) [32]: a classifier learning the object labels and a shape network learning the bounding box aspect ratios. The structure of PyramidCNN is detailed in Fig. 2.

The input to our networks is a pyramid of images at different scales. This enables the networks to detect objects with different sizes and appearing at various distances. Thus, the output contains a pyramid of heatmaps representing bounding boxes associated with confidence scores, positions, aspect ratios, and sizes for each object class. Different from end-to-end learning systems, we do not apply any threshold to the confidence scores in order to avoid any false negatives generated by the first stage.

B. Pose Estimation

The purpose of the second stage is to estimate the object pose by performing iterated likelihood weighting, which offers us robustness and versatility over the search space. This is critical in our context since the manipulation task heavily depends on the accuracy of pose estimations. We expect the second stage to perform robustly even with inaccurate detection from the first stage.

1) Initial Samples: We use a set of weighted samples $\{q^{(i)}, w^{(i)}\}_{i=1}^{M}$ to represent the belief of object pose, where each 6D sample pose $q^{(i)}$ corresponds to a weight $w^{(i)}$. Given an object class o, its pose q, and the corresponding geometry model, we can render a 3D point cloud observation $\boldsymbol{r}$ using the z-buffer of a 3D graphics engine. Essentially, these rendered point clouds are what would be observed if the object had the hypothesized poses, which we refer to as *rendered samples* hereafter. The samples are initialized according to the first stage output. As mentioned in Section IV-A, our CNN produces a density pyramid that is essentially a list of bounding boxes with confidence scores. We perform importance sampling over the confidence scores and initialize our samples uniformly within the 3D workspaces indicated by sampled bounding boxes as shown in Equation (4). More samples are spawned within bounding boxes with higher confidence scores.

2) Likelihood Function: The weight of each sample is calculated by the likelihood function, which evaluates the compatibility of a sample with observations as shown in Equation (5). The likelihood function consists of several parts, including bounding boxes weight, raw pixel-wise inlier ratio, and feature-based inlier ratio. We first define the raw pixel-wise inlier function as:

$$\text{Inlier}(p, p^{'}) = \mathbf{I}\left(||p - p^{'}||_2 < \varepsilon\right), \tag{8}$$

where $p, p^{'} \in \mathbb{R}^3$ refer to a point in observation point cloud $\boldsymbol{z}$ and a point in rendered point cloud from sample pose respectively. $\mathbf{I}$ is the indicator function. A rendered point is considered as an inlier if it is within a certain sensor resolution range ε from an observed point. The point-wise inlier ratio of a rendered sample is then defined as:

$$I = \frac{1}{|\boldsymbol{r}|} \sum_{(u,v)\in z} \text{Inlier}(\boldsymbol{r}_{(u,v)}, \boldsymbol{z}_{(u,v)}), \tag{9}$$

where (u,v) refers to 2D image indices in the rendered sample point cloud $\boldsymbol{r}$ and observation point cloud $\boldsymbol{z}$. $|\cdot|$ refers to point cloud size.

Besides raw point-wise inliers, we extract geometry feature point clouds from both rendered samples and observation point clouds and compute feature inlier ratios. Hereby, we enhance the robustness of the likelihood function by considering contextual geometric information from 3D point clouds. This term prunes wrong poses that agree with the observation only in individual points but neglect higher-level geometric information such as depth discontinuity and sharp object surfaces. We apply feature point extraction introduced by Zhang *et al.* [33] based on local surface smoothness:

$$c_{(u,v)} = \frac{||\sum_{(u',v')\in \mathrm{N}(u,v)} \left(\boldsymbol{p}_{(u',v')} - \boldsymbol{p}_{(u,v)}\right)||}{|\mathrm{N}(u,v)| \cdot ||\boldsymbol{p}_{(u,v)}||} \tag{10}$$

$c_{(u,v)}$ is calculated by adding all displacement vectors from $\boldsymbol{p}_{(u,v)}$ to each of its neighbor points $\mathrm{N}(u,v)$. The point cloud $\boldsymbol{p}$ here can be either rendered sample $\boldsymbol{r}$ or observation $\boldsymbol{z}$. The value is then normalized by the size of $\mathrm{N}(u,v)$ and the length of vector $\boldsymbol{p}_{(u,v)}$. Intuitively, c describes the depth changing rate within a certain local range, which has larger values in areas with acute depth changes and smaller values where object surfaces are consistent. We extract two features, edge points and planar points, by selecting point sets with largest and smallest c values respectively. To balance feature point density in areas with different observation quality, we set a maximum number of edge points and planar points to be extracted from a certain local area. Essentially, a point at (u,v) can be selected as an edge or a planar point only if its c value is larger or smaller than a threshold and if the number of selected points has not exceeded the limit. We find that the algorithm is insensitive to our feature extraction parameters. Finally, we apply feature extraction on both rendered sample and scene observation point cloud to get sample features and observation features. We use the same inlier calculation in Equations (8) and (9) to calculate feature inlier ratios.

The weight w of each hypothesis q is defined as

$$W(q) = \underbrace{\alpha_{box} w_{box} + \alpha_b I_b}_{\text{network terms}} + \underbrace{\alpha_r I_r + \alpha_e I_e + \alpha_p I_p}_{\text{geometric terms}} \tag{11}$$

where w_{box} is the confidence score of the bounding box. I_r is the ratio of pixel-wise inliers in the whole rendered sample point cloud. I_b is the inlier ratio in the portion of rendered sample that is within the bounding box (I_b is 0 if no rendered sample point falls into the bounding box). I_e and I_p are inlier ratios in sample edges and sample planars with respect to observation features. The coefficients α_* represent the importance of each likelihood term and sum up to 1. The first two terms, w_{box} and I_b, *network terms*, are heavily determined by the bounding boxes and describe the consistency between pose sample and detection result. The last three terms, *geometric terms*, weigh how much the current hypothesis explains itself in the scene geometry.

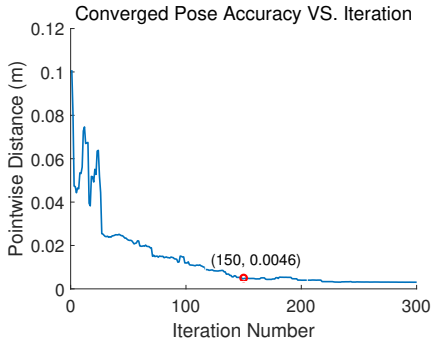


Fig. 3: Plot of pose accuracy vs. iteration numbers of all converged trials. The point-wise distance is calculated using ADD and ADD-S metrics [3] and not used in pose estimation. After about 150 iterations, the point-wise distance reaches below 0.005m.

3) Update Process: To produce object pose estimations, we follow the procedure of iterated likelihood weighting by first assigning a new weight to each sample. Resampling is done with replacement according to sample weights. During the diffusion process shown in Equation (6), each pose $q_t^{(i)}$ is diffused in the space subject to zero-mean Gaussian noises $\mathcal{N}_{T,t}(0,\sigma_{T,t}^2)$ and $\mathcal{N}_{R,t}(0,\sigma_{R,t}^2)$ with time-varying variances for translation and rotation respectively. The standard deviations $\sigma_{T,t}$ and $\sigma_{R,t}$ at iteration t are decayed according to $W(q_t^*)$, the weight of best pose estimation q_t^* at that iteration. Bounding boxes $b_t^{(i)}$ are diffused uniformly within the image. The algorithm terminates when $W(q_t^*)$ reaches a threshold $\bar{w}$, or the iteration limit is reached. Finally, we assume the pose weights of objects in the scene will be much higher than those for non-existing objects.

V. EXPERIMENTS

A. Implementation

We use PyTorch for our CNN implementation based on a VGG16 model pre-trained on ImageNet [34]. (more architectures are tested in [8]). The shape network branch of our CNN predicts 7 different aspect ratios. The size of a training image is 224×224 and contains a single object. The aspect ratio of an object in the training image can be inferred from the width and height of the object. We apply a softmax at the end of the network to generate probability distribution of object classification and aspect ratio. We use cross entropy as the loss function in training.

Our second stage pose estimation relies on the OpenGL graphics engine to render depth images with 3D geometry models and hypothesis poses on Nvidia GTX1080/RTX2070 graphics cards. During the iterated likelihood weighting process, we allocate 625 samples for each iteration and run the algorithm for 400 iterations in total, with ε set to 0.005m. The sample size is limited by the buffer size of our rendering engine, while the iteration limit was set since our pose estimation converges after approximately 150 iterations (see, e.g., Fig. 3) in less than 10s (∼60ms per iteration). Point distance threshold ε was set to approximated distance between adjacent points in 3D point clouds.

In the feature extraction mentioned in Section IV-B.2, we extract up to 5 edge points and 2 planar points from each 5×5 pixel non-overlap sliding window. Given a 3D point cloud $\boldsymbol{p}$, we consider $\boldsymbol{p}_{(u,v)}$ as an *edge point* if $ln(c_{(u,v)}) \geq -5.5$ (see Equation (10)), or a *planar point* otherwise. These hyper-parameters are determined experimentally for clear indication of object boundaries as well as surfaces. The likelihood coefficients are set to $\alpha_{box} = 0.1, \alpha_b = 0.1, \alpha_r = 0.3, \alpha_e = 0.25, \alpha_p = 0.25$. Through experiments, we find that the system performance is sensitive to the total category weight allocated to network terms and geometric terms, rather than the allocation within each category. If the first stage produces accurate detection evaluated by mean average precision (mAP), one can take advantage of it by allocating more weight to network terms. Otherwise, one should reduce the weight of network terms to attenuate the negative impact of underperforming first stage. Since our first stage produces low-mAP detection, we allocate only 20% of the weight on network terms. We allocate the remaining 80% to geometric terms since these terms are robust to adversarial scenarios and unreliable first stage detection. Further weight allocation within each category is done approximately evenly.

During diffusion, standard deviations of the Gaussian noises are decayed by a common factor λ_t, which drops exponentially from 1.0 to 0.0 as $W(q_t^*)$ increases from 0.6 to 1.0. In other words, the standard deviations at iteration t are given by $\sigma_{*,t} = \lambda_t \sigma_{*,0}$, where

$$\lambda_t = \mathbf{I}_{W(q_t^*)\geq 0.6} \cdot \left(\frac{1-W(q_t^*)}{1-0.6}\right)^5 + \mathbf{I}_{W(q_t^*)<0.6} \cdot 1 \qquad (12)$$

Initial standard deviations are $\sigma_{T,0} = 0.07m$ and $\sigma_{R,0} = 0.3rad$ for translation and rotation respectively. The threshold $\bar{w}$ for convergence is set to 0.9.

B. Dataset and Baselines

We use the YCB video dataset [3] as the training data for our first stage PyramidCNN. The YCB video dataset consists of 133,827 frames of 21 objects under normal conditions with balanced and adequate lighting but no occlusion. To test the performance of our two-stage method with baseline methods, PoseCNN [3] and DOPE [4], we collect a testing dataset (i.e., adversarial YCB dataset) from 40 scenes with 15 out of 21 objects from YCB video dataset under adversarial scenarios. In each scene, we place 5-7 different objects on a table and collect seven frames: one in normal lighting, one in darkness, two with different single light sources, and three with different cluttered object placements (see Fig. 4). The dark setting and two single-lighting settings cause bias in image pixels values from the training set and thus undermine network prediction. We refer to these settings as *varied lighting* for simplicity. In addition, object clutter causes occlusions as well as natural information loss and challenges the robustness of pose estimation algorithms. All the scene images and 3D point clouds are gathered by the RGB-D sensor on our Fetch robot. Ground truth bounding boxes and 6D poses are manually labeled.

Fig. 4: Testing dataset with YCB objects under adversarial settings. The base-setting data is collected with regular lighting without occlusions. The dark-setting data is collected with lights off in the room. The single lighting data is collected with a flash light. Object poses are the same in previous three settings. Data in three occlusion scenes is collected with the same objects randomly stacked.

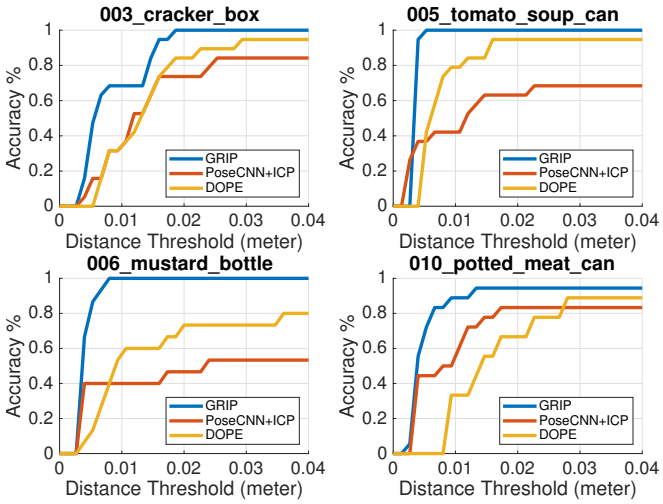
C. Evaluation

1) Comparing accuracy with PoseCNN and DOPE with 4 YCB objects: We compare our pose estimation accuracy with PoseCNN (with ICP refinement) and DOPE on the YCB dataset under different adversarial settings. We use pre-trained models from the authors' Github page for PoseCNN[1] and DOPE[2] and train our first stage PyramidCNN using 2500 frames from the original YCB video dataset. Since DOPE is trained with 5 of 21 objects from the YCB Video Dataset, we first compare all three methods on 4 of them: 003_cracker_box, 005_tomato_soup_can, 006_mustard_bottle and 010_potted_meat_can. The fifth object, 004_sugar_box, was unavailable from the market when this experiment was set up. We use ADD and ADD-S metrics [3] to calculate pose error for asymmetric and symmetric objects (marked with asterisks in Table. I) respectively. In manipulation tasks, the bearable pose estimation error is bounded by the clearance that objects have when placed in the robot end effector. Based on the sizes of Fetch robot gripper and target objects, we choose 0.04m as the maximum error tolerance. We then plot accuracy-threshold curves within a range of [0.00m, 0.04m] in Fig. 5 and calculate AUC (Area Under accuracy-threshold Curve) as the evaluation metric. *GRIP* outperforms the other two methods under most error thresholds, especially lower ones, and thereby facilitates robotic manipulation tasks. See Fig. 6 for a qualitative comparison of all three methods with different adversarial settings.

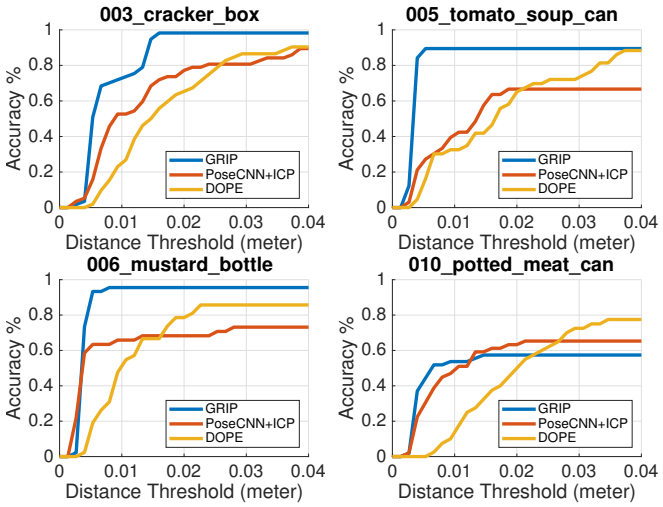
2) Comparing accuracy with PoseCNN with 15 YCB objects: Next, we perform an extensive comparison of our method with PoseCNN (with ICP) on 15 of the 21 YCB objects. Table I and Fig. 7 show our overall results and detailed accuracy evaluations for each object.

GRIP outperforms PoseCNN+ICP for most objects under all 3 settings. All methods have worse performances under varied lighting and occlusions as opposed to the basic setting.

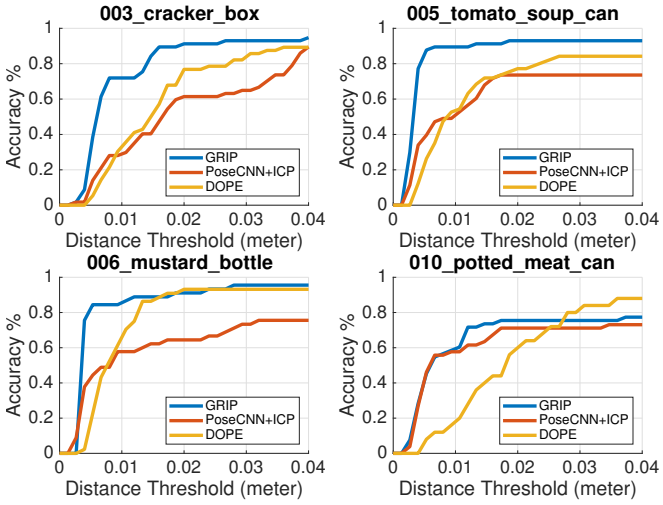
[1]https://rse-lab.cs.washington.edu/projects/posecnn/
[2]https://github.com/NVlabs/Deep_Object_Pose



(a) Base settings.



(b) Varied Light settings.



(c) Occlusion settings

Fig. 5: The comparison between DOPE, PoseCNN+ICP and our *GRIP* two-stage method on pose estimation accuracy of 4 objects mentioned in Sec. V-C.

Area Under Accuracy-Threshold Curve	Base			Varied Lighting			Occlusions		
	DOPE	PoseCNN*	**GRIP**	DOPE	PoseCNN*	**GRIP**	DOPE	PoseCNN*	**GRIP**
003_cracker_box*	0.6384	0.5925	**0.7878**	0.5509	0.6225	**0.7923**	0.5703	0.4850	**0.7442**
005_tomato_soup_can*	0.7691	0.5535	**0.9015**	0.5326	0.5181	**0.8104**	0.6372	0.6013	**0.8347**
006_mustard_bottle*	0.5720	0.4280	**0.8860**	0.6290	0.6310	**0.8552**	0.7295	0.5864	**0.8208**
007_tuna_fish_can*	\	0.3763	**0.7670**	\	0.3616	**0.7849**	\	0.3915	**0.6220**
010_potted_meat_can*	0.5556	0.6756	**0.8226**	0.4347	**0.5273**	0.5006	0.5045	0.5962	**0.6342**
011_banana	\	0.3922	**0.5467**	\	0.3449	**0.5591**	\	0.2137	**0.2750**
019_pitcher_base	\	**0.2442**	0.1774	\	**0.2490**	0.1659	\	**0.3341**	0.1874
021_bleach_cleanser	\	0.3302	**0.5671**	\	0.3111	**0.5523**	\	0.2204	**0.4635**
024_bowl*	\	0.3190	**0.8674**	\	0.3397	**0.7109**	\	0.2345	**0.6185**
025_mug	\	**0.3491**	0.2201	\	**0.3176**	0.2170	\	**0.2216**	0.2094
037_scissors	\	**0.5450**	0.3192	\	**0.5812**	0.1647	\	**0.3548**	0.1783
040_large_marker*	\	0.2071	**0.7537**	\	0.4094	**0.6750**	\	0.2736	**0.6711**
051_large_clamp	\	0.2405	**0.4927**	\	**0.2061**	0.1642	\	0.0645	**0.2551**
052_extra_large_clamp	\	**0.4000**	0.1742	\	0.1460	**0.1742**	\	**0.2147**	0.1441
061_foam_brick*	\	0.8094	**0.8333**	\	0.6380	**0.8011**	\	0.5419	**0.8297**
Overall	\	0.4308	**0.6078**	\	0.4136	**0.5285**	\	0.3556	**0.4992**

TABLE I: Overall Performance (Area Under accuracy-threshold Curve) of 15 YCB Objects on DOPE, PoseCNN with ICP and our *GRIP* method. Symmetric objects are marked with stars and evaluated using ADD-S; asymmetric objects are evaluated using ADD.

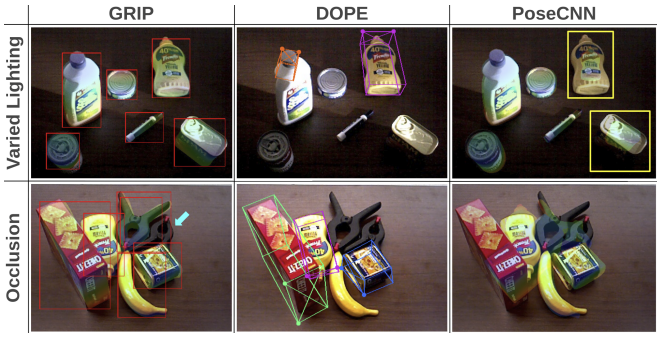


Fig. 6: Comparison of *GRIP*, DOPE and PoseCNN under adversarial scenarios. In varied lighting condition, DOPE only detects 006_mustard_bottle correctly while PoseCNN makes inaccurate depth estimation (marked yellow). In occlusion condition, DOPE misses half of the objects while PoseCNN fails to detect object poses in clutter. *GRIP* correctly detects all objects under both scenes except 051_large_clamp under occlusion setting (cyan arrow) where the sampling converges to object 052_extra_large_clamp because of their geometric similarity.

We can infer the strengths and weaknesses of each method from its performance variance among different objects. For example, PoseCNN with ICP performs better on symmetric objects such as 003_cracker_box and 061_foam_brick as opposed to others such as 021_bleach_cleanser. Symmetric objects contain repetitive features which are more likely to be captured by learning-based systems. *GRIP* performs better on objects that are well recognizable under a depth camera. Large and compact objects such as 006_mustard_bottle and 024_bowl naturally generate dense and continuous 3D point cloud observations that effectively capture their geometry. Objects with thin or articulated parts, such as 037_scissors, 052_extra_large_clamp, and 025_mug, produce sparse point clouds around their handle-like parts that do not effectively reveal the scene geometry, especially object orientations. Hence, our *GRIP* algorithm best suits scenarios where rich depth sensory data are available due to detectable object dimensions and surface materials or high-definition depth sensors. Finally, distinguishing near-identical objects remains challenging. For instance, 051_large_clamp and 052_extra_large_clamp have identical colors and shapes and differ only insignificantly in sizes. This results in poor estimation accuracy by all methods.

D. Robotic Manipulation

GRIP has been successfully used as the perception module in real robot manipulation tasks, such as a grocery packing task shown in the video[3].

VI. CONCLUSIONS

We have introduced *GRIP* as a two-stage method for robust 6D object pose estimation suited to adversarial settings. *GRIP* demonstrated similar and improved performance with respect to state-of-the-art neural network pose estimators considering the adversarial YCB dataset. The key insight of *GRIP* is to avoid hard thresholding, which introduces false positives and false negatives, until a final pose estimate is required. Avoiding hard thresholds increases the possibility of finding the real pose in adversarial environments. In addition, a generative second stage inherently provides an avenue for explainable perception, without requiring deciphering network weights. Also, this generative process readily extends to tracking over multiple instances of time through the inclusion of a proper process model. The results presented are also amenable to improvement due to the limited types of features considered. These benefits come at the cost of assuming only one instance of each object is present in the scene. For future work, we aim to investigate these limitations through exploring features amenable to robust inference with multiple object instances in greater clutter and sampling techniques for faster convergence.

[3]https://www.youtube.com/watch?v=W0KvzMhIJxo

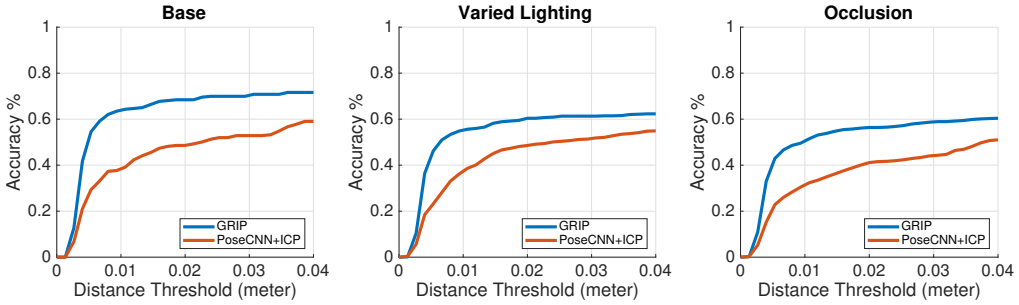


Fig. 7: Overall pose estimation accuracy of 15 YCB objects using PoseCNN and our *GRIP* method.

REFERENCES

[1] Marcus Gualtieri, Andreas ten Pas, and Robert Platt. Pick and place without geometric object models. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7433–7440. IEEE, 2018.

[2] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[3] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[4] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[5] 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. *arXiv preprint arXiv:1901.04780*, 2019.

[6] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 1:1, 2017.

[7] Zhiqiang Sui, Zheming Zhou, Zhen Zeng, and Odest Chadwicke Jenkins. Sum: Sequential scene understanding and manipulation. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3281–3288. IEEE, 2017.

[8] Yanqi Liu, Alessandro Costantini, R Bahar, Zhiqiang Sui, Zhefan Ye, Shiyang Lu, and Odest Chadwicke Jenkins. Robust object estimation using generative-discriminative inference for secure robotics applications. In *Proceedings of the International Conference on Computer-Aided Design*, page 75. ACM, 2018.

[9] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[10] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[11] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[12] Shiwali Mohan, Aaron H Mininger, James R Kirk, and John E Laird. Acquiring grounded representations of words with situated interactive instruction. In *Advances in Cognitive Systems*. Citeseer, 2012.

[13] Venkatraman Narayanan and Maxim Likhachev. Discriminatively-guided deliberative perception for pose estimation of multiple 3d object instances. In *Robotics: Science and Systems*, 2016.

[14] Venkatraman Narayanan and Maxim Likhachev. Perch: Perception via search for multi-object recognition and localization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5052–5059. IEEE, 2016.

[15] Ziyuan Liu, Dong Chen, Kai M Wurm, and Georg von Wichert. Table-top scene analysis using knowledge-supervised mcmc. *Robotics and Computer-Integrated Manufacturing*, 33:110–123, 2015.

[16] Dominik Joho, Gian Diego Tipaldi, Nikolas Engelhard, Cyrill Stachniss, and Wolfram Burgard. Nonparametric bayesian models for unsupervised scene analysis and reconstruction. *Robotics*, page 161, 2013.

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

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[19] Matei Ciocarlie, Kaijen Hsiao, Edward Gil Jones, Sachin Chitta, Radu Bogdan Rusu, and Ioan A Şucan. Towards reliable grasping and manipulation in household environments. In *Experimental Robotics*, pages 241–252. Springer, 2014.

[20] Chavdar Papazov, Sami Haddadin, Sven Parusel, Kai Krieger, and Darius Burschka. Rigid 3d geometry matching for grasping of known objects in cluttered scenes. *The International Journal of Robotics Research*, 31(4):538–553, 2012.

[21] Andreas Ten Pas and Robert Platt. Localizing handle-like grasp affordances in 3d point clouds. In *Experimental Robotics*, pages 623–638. Springer, 2016.

[22] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

[23] Zhiqiang Sui, Odest Chadwicke Jenkins, and Karthik Desingh. Axiomatic particle filtering for goal-directed robotic manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4429–4436. IEEE, 2015.

[24] Karthik Desingh, Odest Chadwicke Jenkins, Lionel Reveret, and Zhiqiang Sui. Physically plausible scene estimation for manipulation in clutter. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 1073–1080. IEEE, 2016.

[25] Zhen Zeng, Zheming Zhou, Zhiqiang Sui, and Odest Chadwicke Jenkins. Semantic robot programming for goal-directed manipulation in cluttered scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7462–7469. IEEE, 2018.

[26] Chaitanya Mitash, Abdeslam Boularias, and Kostas Bekris. Robust 6D object pose estimation with stochastic congruent sets. *arXiv preprint arXiv:1805.06324*, 2018.

[27] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*, volume 33, pages 205–215. Wiley Online Library, 2014.

[28] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[29] Stephen J. Mckenna and Hammadi Nait-Charif. Tracking human motion using auxiliary particle filters and iterated likelihood weighting. *Image Vision Comput.*, 25:852–862, 2007.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

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

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[33] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.

[34] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.