Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects

Jonathan Tremblay **Thang To** **Balakumar Sundaralingam**[†]
Yu Xiang **Dieter Fox** **Stan Birchfield**
NVIDIA
{jtremblay,thangt,yxiang,dieterf,sbirchfield}@nvidia.com
[†]Also with Univ. of Utah, bala@cs.utah.edu

Abstract:
Using synthetic data for training deep neural networks for robotic manipulation holds the promise of an almost unlimited amount of pre-labeled training data, generated safely out of harm's way. One of the key challenges of synthetic data, to date, has been to bridge the so-called *reality gap*, so that networks trained on synthetic data operate correctly when exposed to real-world data. We explore the reality gap in the context of 6-DoF pose estimation of known objects from a single RGB image. We show that for this problem the reality gap can be successfully spanned by a simple combination of domain randomized and photorealistic data. Using synthetic data generated in this manner, we introduce a one-shot deep neural network that is able to perform competitively against a state-of-the-art network trained on a combination of real and synthetic data. To our knowledge, this is the first deep network trained only on synthetic data that is able to achieve state-of-the-art performance on 6-DoF object pose estimation. Our network also generalizes better to novel environments including extreme lighting conditions, for which we show qualitative results. Using this network we demonstrate a real-time system estimating object poses with sufficient accuracy for real-world semantic grasping of known household objects in clutter by a real robot.[1]

Keywords: computer vision, pose estimation, synthetic data, randomization

1 Introduction

In order for robots to operate safely and effectively alongside humans, they must be aware of their surroundings. One aspect of this awareness is knowledge of the 3D position and orientation of objects in the scene, often referred to as 6-DoF (degrees of freedom) pose. This knowledge is important to perform pick-and-place of objects, handoff from a person, or watch someone handle the object for imitation learning. In this work we focus on rigid, known objects for which a prior training time to learn the appearance and shape of the objects is allowed. Our goal is to infer the 3D pose of such objects, in clutter, from a single RGB image in real time for the purpose of enabling the robot to manipulate such objects.

While deep neural networks have been successfully applied to the problem of object detection in 2D [1, 2, 3], they have only recently begun to be applied to 3D object detection and pose estimation [4, 5, 6]. Unlike 2D object detection, it is prohibitive to manually label data for 3D detection. Due to this difficulty of collecting sufficiently large amounts of labeled training data, such approaches are typically trained on real data that are highly correlated with the test data (*e.g.*, same camera, same object instances, similar lighting conditions). As a result, one challenge of existing approaches is generalizing to test data that are significantly different from the training set.

Synthetic data is a promising alternative for training such deep neural networks, capable of generating an almost unlimited amount of pre-labeled training data with little effort. Synthetic data comes with its own problems, however. Chief among these is the *reality gap*, that is, the fact that networks trained on synthetic data usually do not perform well on real data without additional fine-tuning or

[1]Video and code: https://research.nvidia.com/publication/2018-09_Deep-Object-Pose .

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

other tricks. A recently proposed solution to this problem is *domain randomization* [7], in which the training data is randomized in non-realistic ways so that, at test time, real data appears to the network as simply another variation. Domain randomization has proved successful at detecting colored geometric shapes on a table [7], flying a quadcopter indoors [8], or learning visuomotor control for reaching [9] or pick-and-place [10] of a brightly colored cube.

In this paper we explore a powerful complement to domain randomization (DR), namely, using photorealistic data [11]. We show that a simple combination of DR data with such photorealistic data yields sufficient variation and complexity to train a deep neural network that is then able to operate on real data without any fine-tuning. Additionally, our synthetically-trained network generalizes well to a variety of real-world scenarios, including various backgrounds and extreme lighting conditions. Our contributions are thus as follows:

- A one-shot, deep neural network-based system that infers, in near real time, the 3D poses of known objects in clutter from a single RGB image without requiring post-alignment. This system uses a simple deep network architecture, trained entirely on simulated data, to infer the 2D image coordinates of projected 3D bounding boxes, followed by perspective-*n*-point (P*n*P) [12]. We call our system DOPE (for "deep object pose estimation").[2]
- Demonstration that combining both non-photorealistic (domain randomized) and photorealistic synthetic data for training robust deep neural networks successfully bridges the reality gap for real-world applications, achieving performance comparable with state-of-the-art networks trained on real data.
- An integrated robotic system that shows the estimated poses are of sufficient accuracy to solve real-world tasks such as pick-and-place, object handoff, and path following.

2 Approach

We propose a two-step solution to address the problem of detecting and estimating the 6-DoF pose of all instances of a set of known household objects from a single RGB image. First, a deep neural network estimates belief maps of 2D keypoints of all the objects in the image coordinate system. Secondly, peaks from these belief maps are fed to a standard perspective-*n*-point (P*n*P) algorithm [12] to estimate the 6-DoF pose of each object instance. In this section we describe these steps, along with our novel method of generating synthetic data for training the neural network.

2.1 Network architecture

Inspired by convolutional pose machines (CPMs) [13, 14], our one-shot fully convolutional deep neural network detects keypoints using a multistage architecture. The feedforward network takes as input an RGB image of size $w \times h \times 3$ and branches to produce two different outputs, namely, belief maps and vector fields. There are nine belief maps, one for each of the projected 8 vertices of the 3D bounding boxes, and one for the centroids. Similarly, there are eight vector fields indicating the direction from each of the 8 vertices to the corresponding centroid, similar to [5], to enable the detection of multiple instances of the same type of object. (In our experiments, $w = 640$, $h = 480$.)

The network operates in stages, with each stage taking into account not only the image features but also the outputs of the immediately preceding stage. Since all stages are convolutional, they leverage an increasingly larger effective receptive field as data pass through the network. This property enables the network to resolve ambiguities in the early stages due to small receptive fields by incorporating increasingly larger amounts of context in later stages.

Image features are computed by the first ten layers from VGG-19 [15] (pretrained on ImageNet), followed by two 3×3 convolution layers to reduce the feature dimension from 512 to 256, and from 256 to 128. These 128-dimensional features are fed to the first stage consisting of three $3 \times 3 \times 128$ layers and one $1 \times 1 \times 512$ layer, followed by either a $1 \times 1 \times 9$ (belief maps) or a $1 \times 1 \times 16$ (vector fields) layer. The remaining five stages are identical to the first stage, except that they receive a 153-dimensional input ($128 + 16 + 9 = 153$) and consist of five $7 \times 7 \times 128$ layers and one $1 \times 1 \times 128$ layer before the $1 \times 1 \times 9$ or $1 \times 1 \times 16$ layer. All stages are of size $w/8$ and $h/8$, with ReLU activation functions interleaved throughout.

[2] Among the various meanings of the word *dope*, we prefer to emphasize "very good."

2.2 Detection and pose estimation

After the network has processed an image, it is necessary to extract the individual objects from the belief maps. In contrast to other approaches in which complex architectures or procedures are required to individuate the objects [4, 5, 22, 6], our approach relies on a simple postprocessing step that searches for local peaks in the belief maps above a threshold, followed by a greedy assignment algorithm that associates projected vertices to detected centroids. For each vertex, this latter step compares the vector field evaluated at the vertex with the direction from the vertex to each centroid, assigning the vertex to the closest centroid within some angular threshold of the vector.

Once the vertices of each object instance have been determined, a P*n*P algorithm [12] is used to retrieve the pose of the object, similar to [6, 4]. This step uses the detected projected vertices of the bounding box, the camera intrinsics, and the object dimensions to recover the final translation and rotation of the object with respect to the camera. All detected projected vertices are used, as long as at least the minimum number (four) are detected.

2.3 Data generation

A key question addressed by this research is how to generate effective training data for the network. In contrast to 2D object detection, for which labeled bounding boxes are relatively easy to annotate, 3D object detection requires labeled data that is almost impossible to generate manually. Although it is possible to semi-automatically label data (using a tool such as LabelFusion [16]), the labor-intensive nature of the process nevertheless impedes the ability to generate training data with sufficient variation. For example, we are not aware of any real-world training data for 6-DoF object pose estimation that includes extreme lighting conditions or poses.

To overcome these limitations of real data, we turn to synthetically generated data. Specifically, we use a combination of non-photorealistic domain randomized (DR) data and photorealistic data to leverage the strengths of both. As shown later in the experimental results, these two types of data complement one another, yielding results that are much better than those achieved by either alone. Synthetic data has an additional advantage in that it avoids overfitting to a particular dataset distribution, thus producing a network that is robust to lighting changes, camera variations, and backgrounds.

Since our goal is robotic manipulation of the objects whose 6-DoF pose has been estimated, we leverage the popular YCB objects [17].[3] These objects are readily available and suitable for manipulation by compliant, low-impedance human-friendly robots such as the Baxter. Both the domain randomized and photorealistic data were created by placing the YCB object models in different virtual environments. All data were generated by a custom plugin we developed for Unreal Engine 4 (UE4) called NDDS [18]. By leveraging asynchronous, multithreaded sequential frame grabbing, the plugin generates data at a rate of 50–100 Hz, which is significantly faster than either the default UE4 screenshot function or the publicly available Sim4CV tool [19]. Fig. 1 shows examples of images generated using our plugin, illustrating the diversity and variety of both domain randomized and photorealistic data.

Domain randomization. The domain randomized images were created by placing the foreground objects within virtual environments consisting of various distractor objects in front of a random background. Images were generated by randomly varying distractors, overlaid textures, backgrounds, object poses, lighting, and noise. More specifically, the following aspects of the scene were randomized, similar to [20]: number and types of distractors, selected from a set of 3D models (cones, pyramids, spheres, cylinders, partial toroids, arrows, etc.); texture on the object of interest, as well as on distractors (solid, striped); solid colored background, photograph (from a subset of 10,000 images from the COCO dataset [21]), or procedurally generated background image with random multicolored grid pattern; 3D pose of objects and distractors, sampled uniformly; directional lights with random orientation, color, and intensity; and visibility of distractors.

Photorealistic images. The photorealistic data were generated by placing the foreground objects in 3D background scenes with physical constraints. Backgrounds were chosen from standard UE4 virtual environments such as a kitchen, sun temple, and forest. These environments were chosen for

[3] http://www.ycbbenchmarks.com/object-models

Figure 1: Example images from our domain randomized (left) and photorealistic (right) datasets used for training.

their high-fidelity modeling and quality, as well as for the variety of indoor and outdoor scenes. For this dataset we included the same subset of 21 household YCB objects as in [5]. Allowed to fall under the weight of gravity, and to collide with each other and with surfaces in the scene, these objects interact in physically plausible ways. While the objects were falling, the virtual camera system was rapidly teleported to random azimuths, elevations, and distances with respect to a fixation point to collect data. Azimuth ranged from –120° to +120° (to avoid collision with the wall, when present), elevation from 5° to 85°, and distance from 0.5 m to 1.5 m. We recently made this dataset, known as Falling Things (FAT), publicly available [11].

3 Experimental Results

Since our goal is robotic manipulation of household objects, we focused our evaluation solely on images of the YCB objects, which incidentally is also the primary focus of the leading method for 6-DoF pose estimation, namely PoseCNN [5]. We compare our DOPE method to PoseCNN on both the YCB-Video dataset introduced in [5], as well as on a dataset that we collected using a different camera, a variety of backgrounds, and extreme lighting conditions. Since PoseCNN has already been shown to outperform the single-shot method of Tekin *et al.* [6], which itself beats the more complex networks of BB8 [4] and SSD-6D [22] on the standard LINEMOD [23] and Occluded-LINEMOD [24] datasets, the comparison of our method against PoseCNN is sufficient to answer the question as to whether our simple network architecture coupled with our novel synthetic data generation procedure are able to yield results that are on par with state-of-the-art. Moreover, of the four algorithms just mentioned, PoseCNN is the only one with publicly available source code (at the time of this writing) capable of detecting YCB objects.[4]

3.1 Datasets and metric

For the first set of experiments we used the YCB-Video dataset [5], which is composed of ∼133k frames including pose annotation for all objects in all frames. The objects consist of a subset of 21 objects taken from the YCB dataset. We used the same testing set of 2,949 frames as suggested by Xiang *et al.* [5]. We also collected our own dataset of four videos captured in a variety of extreme lighting conditions, by a Logitech C960 camera. In these videos we placed a subset of 5 objects from the 21 YCB objects in YCB-Video, including multiple instances of the same objects in many image frames. To our knowledge, this is the first such dataset with multiple simultaneous instances. The 5 objects are cracker box (003), sugar box (004), tomato soup can (005), mustard bottle (006), and potted meat can (010), where the numbers in parentheses indicate the object number in the YCB

[4] https://github.com/yuxng/PoseCNN

dataset. These objects were selected for their easy availability at typical grocery stores, graspability, and visual texture. We adopt the average distance (ADD) metric [23, 5] for evaluation, which is the average 3D Euclidean distance of all model points between ground truth pose and estimated pose.

3.2 Training

For training we used ∼60k domain-randomized image frames mixed with ∼60k photorealistic image frames. Whereas we used a separate set of ∼60k images in the former case (one dataset per object), in the latter case we used the same dataset for all objects, since it consists of the 21 YCB objects randomly interacting with one another. For data augmentation, Gaussian noise ($\sigma = 2.0$), random contrast ($\sigma = 0.2$) and random brightness ($\sigma = 0.2$) were added. For PoseCNN [5], we used the publicly available weights, which were trained using an undisclosed synthetic dataset and fine-tuned on images from separate videos of the YCB-Video dataset.

To avoid the vanishing gradients problem with our network, a loss was computed at the output of each stage, using the L_2 loss for the belief maps and vector fields. The ground truth belief maps were generated by placing 2D Gaussians at the vertex locations with $\sigma = 2$ pixels. The ground truth vector fields were generated by setting pixels to the normalized x- and y-components of the vector pointing toward the object's centroid. Only the pixels within a 3-pixel radius of each ground-truth vertex were set in this manner, with all other pixels set to zero. Wherever more than one vertex resided within this radius, one of them was selected at random to be used for generating these components.

Our network was implemented using PyTorch v0.4 [25]. The VGG-19 feature extractions were taken from publicly available trained weights in `torchvision` open models. The networks were trained for 60 epochs with a batchsize of 128. Adam [26] was used as the optimizer with learning rate set at 0.0001. The system was trained on an NVIDIA DGX workstation (containing 4 NVIDIA P100 or V100 GPUs), and testing used an NVIDIA Titan X.

3.3 YCB-Video dataset

Fig. 2 shows the accuracy-threshold curves for five YCB objects using the ADD metric for our DOPE method compared with PoseCNN. For our method we include three versions depending on the training set: using domain randomized data only (∼60k synthetic images), using photorealistic data only (∼60k synthetic images), and using both (∼120k synthetic images). We also show results of PoseCNN trained on our DR+photo synthetic data, without real data.

These results reveal that our simple network architecture, trained only on synthetic data, is able to achieve results on par with the state-of-the-art PoseCNN method trained on a mixture of synthetic and real data. These results are made more significant when it is noted that the real images on which PoseCNN was trained were taken from other videos in the same YCB-Video dataset, meaning that they were captured by the same camera under similar conditions as the test data. Even so, our synthetic-only approach achieves higher area under the curve (AUC) for 4 of the 5 objects, as well as better results at thresholds less than 2 cm, which is about the limit of graspability error for the Baxter robot's parallel jaw end effector. The reason our network fails to detect many of the potted meat can instances is due to severely occluded frames in which only the top of the can is visible; since our synthetic data does not properly model the highly reflective metallic material of the top surface, the network does not recognize these pixels as belonging to the can. We leave the incorporation of such material properties to future work.

3.4 Extreme lighting dataset

To test the generalization performance of our DOPE approach compared with PoseCNN, we ran the two networks on the extreme lighting videos that we collected. Fig. 3 shows four images from these videos,[5] showing qualitatively that the poses found by DOPE trained using our combined synthetic data generation procedure are more robust and precise than those of PoseCNN. These qualitative results highlight the promise of using synthetic data to yield better generalization across cameras and lighting conditions than using real data (due to the difficulty of labeling the latter).

[5] Results on other video frames are similar to those shown; see the accompanying video.

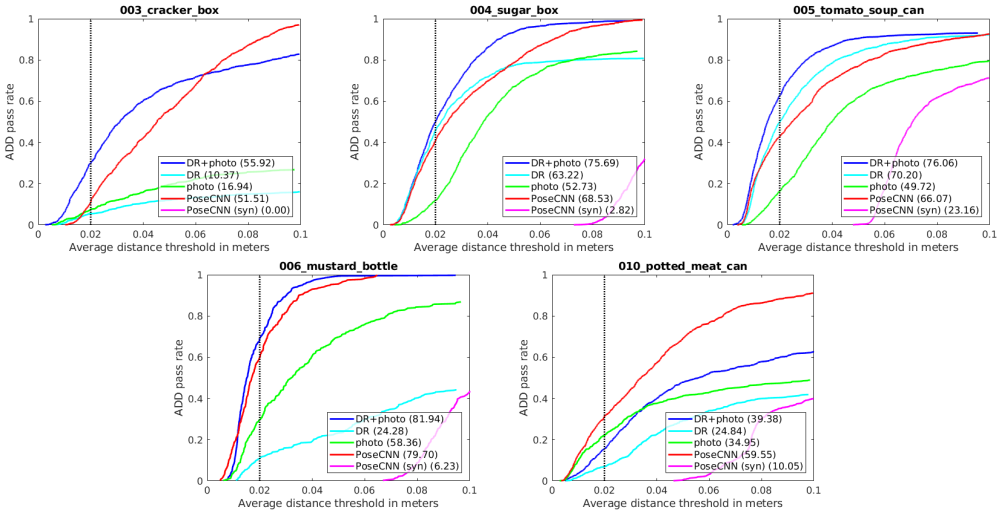


Figure 2: Accuracy-threshold curves for our DOPE method compared with PoseCNN [5] for 5 YCB objects on the YCB-Video dataset. Shown are versions of our method trained using domain-randomized data only (DR), synthetic photorealistic data only (photo), and both (DR+photo). The numbers in the legend display the area under the curve (AUC). The vertical dashed line indicates the threshold corresponding approximately to the level of accuracy necessary for grasping using our robotic manipulator (2 cm). Our method (blue curve) yields the best results for 4 out of 5 objects.

3.5 Additional experiments

To test the effect of dataset size, we trained our network on DR data only using 2k, 10k, 20k, 50k, 100k, 200k, 300k, . . ., 1M. (We arbitrarily chose the sugar box for this experiment.) The biggest performance increase occurred from 2k (0.25 AUC) to 10k (54.22 AUC), results saturated around 100k (60.6 AUC), and the highest value was achieved with 300k (66.64 AUC). This experiment was then repeated using photorealistic images instead, with the highest value achieved at 600k (62.94 AUC). Both of these results are far below the DR + photo result (77.00 AUC) reported above with 120k. These results confirm that our proposed method of mixing DR and photorealistic synthetic data achieves greater success at domain transfer than either DR or photorealistic images alone.

We also compared different mixing percentages. That is, we trained with increments of 10% of each dataset while keeping the other at 100% (*e.g.*, using 50% of the DR dataset with 100% of the photorealistic dataset). The performance was comparable for all networks as long as at least 40% of either dataset was included.

Fig. 4 shows the impact of the number of stages on both accuracy and computing time. As expected, additional stages yield higher accuracy at the cost of greater computation.

3.6 Robotic manipulation

For our purposes, the ultimate test of a pose estimation method is whether its accuracy is sufficient for robotic grasping. We attached a Logitech C960 camera to the waist of a Baxter robot, and calibrated the camera to the robot base using a standard checkerboard target visible to both the Logitech and wrist cameras. The parallel jaw gripper moves from an opening of approximately 10 cm to 6 cm, or from 8 cm to 4 cm, depending on how the fingers are attached. Either way the travel distance of the gripper is just 4 cm, meaning that the error can be no more than 2 cm on either side of the object during the grasp.

For quantitative results, we placed the 5 objects, amidst clutter, at 4 different locations on a table in front of the robot, at 3 different orientations for each of the 4 locations. The robot was instructed to move to a pregrasp point above the object, then execute a top-down grasp, yielding 12 trials per object. Of these 12 attempts, the number of successful grasps were as follows: 10 (cracker), 10 (meat), 11 (mustard), 11 (sugar), and 7 (soup). The difficulty of the soup arises because the round

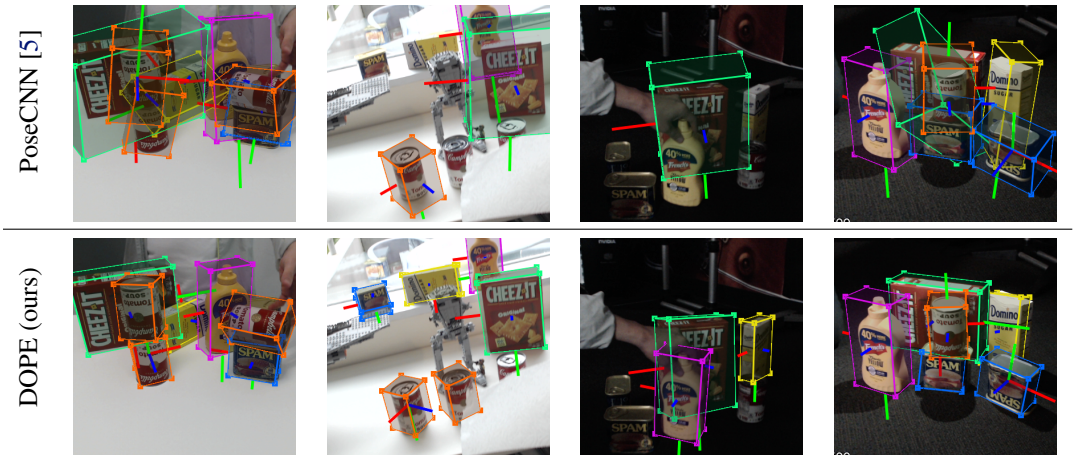
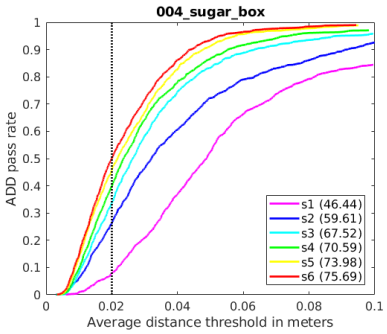


Figure 3: Pose estimation of YCB objects on data showing extreme lighting conditions. TOP: PoseCNN [5], which was trained on a mixture of synthetic data and real data from the YCB-Video dataset [5], struggles to generalize to this scenario captured with a different camera, extreme poses, severe occlusion, and extreme lighting changes. BOTTOM: Our proposed DOPE method generalizes to these extreme real-world conditions even though it was trained only on synthetic data; all objects are detected except the severely occluded soup can (2nd column) and three dark cans (3rd column).



	speed (ms)	AUC
1 stage	57	46.44
2 stages	88	59.61
3 stages	124	67.52
4 stages	165	70.59
5 stages	202	73.98
6 stages	232	75.69

Figure 4: Accuracy-threshold curves with various numbers of stages, showing the benefit of additional stages to resolve ambiguity from earlier stages. The table shows the total execution time, including object extraction and P*n*P, and performance of the system for different numbers of stages.

geometry is unforgiving to top-down grasps. When we repeated the experiment with the soup lying on its side, this number increased to 9. These experiments demonstrate that our DOPE pose estimation method is robust enough in real-world conditions to execute successful grasps, without any closed-loop servoing, in a majority of cases. Sources of error include the pose estimation algorithm, mis-calibration between the camera and robot, and imprecise robot control. Although it is difficult to pinpoint the exact cause of error in each case, anecdotally the imprecise control of the robot appears to play a disproportionate role in many cases we observed. For better control, a method like that of Levi *et al.* [27] might be appropriate.

To our knowledge, these results are the first repeatable experiments of semantic robotic grasping of household objects using the output of a real-time 6-DoF object pose estimator. Unlike the semantic grasping work of [28], our system is not limited to top-down grasps of static objects. Rather, because it estimates 6-DoF pose in real time, our system is capable of performing pick-and-place operations where one object is placed on top of another object, as shown in Fig. 5, as well as grasping the object out of a human's hand (*i.e.*, handoff from the human to the robot), and causing the object to follow the 6-DoF path of another object in real time. Unlike the approach of Tobin *et al.* [7], objects are not restricted to be on the table when grasped but can be anywhere in the visible workspace.

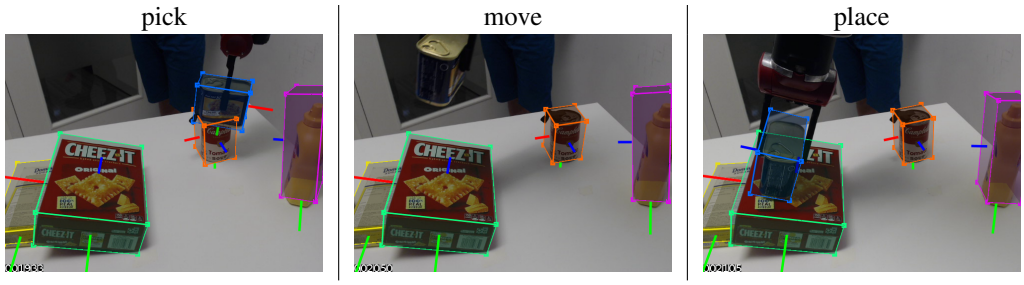


Figure 5: Robotic pick-and-place of a potted meat can on a cracker box. Note that the can is initially resting on another object rather than on the table, and that the destination box is not required to be aligned with the table, since the system estimates full 6-DoF pose of all objects. Note also that the can is aligned with the box (as desired) and within a couple centimeters of the center of the box.

4 Related Work

Object detection and 6-DoF pose estimation. The problem of object detection and 6-DoF pose estimation from RGB images has been addressed by many researchers using traditional computer vision methods, such as [29, 30, 31, 32]. Recently, several deep learning-based approaches have appeared that operate directly on RGB images [4, 5, 22, 6], showing improved performance with respect to difficulties such as occlusion. Of these, BB8 [4] and PoseCNN [5] first segment the objects, then estimate the object pose, whereas the approaches of Tekin *et al.* [6] and Kehl *et al.* [22] rely on the single-shot networks of YOLO and SSD, respectively. All of these networks were trained on real data, thus limiting their ability to generalize to new scenarios (*e.g.*, lighting conditions). Moreover, directly regressing to the pose bakes the camera intrinsics directly into the network weights, causing additional errors when applying the network to a different camera. Our approach differs from these methods in the following ways: 1) We train only on synthetic data, 2) our single-shot belief map network architecture is simpler, and 3) we demonstrate the accuracy of the system in the context of real-world robotic manipulation. Moreover, our approach does not rely on post-refinement of the estimated pose, and our use of P*n*P allows the method to operate on different camera intrinsics without retraining. For alternate approaches to related problems, see [33, 34, 35, 36, 37].

Synthetic data for training. Given the deep learning hunger for large amounts of labeled training data, a recent research trend has focused on providing synthetic datasets for training [38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 19, 48]. Most of these datasets are photorealistic, thus requiring significant modeling effort from skilled 3D artists. To address this limitation, domain randomization [7, 8] has been proposed as an inexpensive alternative that forces the network to learn to focus on essential features of the image by randomizing the training input in non-realistic ways. While domain randomization has yielded promising results on several tasks, it has not yet been shown to produce state-of-the-art results compared with real-world data. In [7, 20], for example, the authors found that fine-tuning with real data was necessary for domain randomization to compete with real data. We hypothesize that domain randomization alone is not sufficient for the network to fully understand a scene, given its non-realism and lack of context. Thus, our approach of using photorealistic data to complement domain randomization can be seen as a promising solution to this problem.

5 Conclusion

We have presented a system for detecting and estimating the 6-DoF pose of known objects using a novel architecture and data generation pipeline. The network employs multiple stages to refine ambiguous estimates of the 2D locations of projected vertices of each object's 3D bounding cuboid. These points are then used to predict the final pose using P*n*P, assuming known camera intrinsics and object dimensions. We have shown that a network trained only on synthetic data can achieve state-of-the-art performance compared with a network trained on real data, and that the resulting poses are of sufficient accuracy for robotic manipulation. Further research should be aimed at increasing the number of objects, handling symmetry, and incorporating closed-loop refinement to increase grasp success.

References

[1] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[2] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *CVPR*, 2017.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[4] M. Rad and V. Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *ICCV*, 2017.

[5] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *RSS*, 2018.

[6] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6D object pose prediction. In *CVPR*, 2018.

[7] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

[8] F. Sadeghi and S. Levine. CAD^{2}RL: Real single-image flight without a single real image. In *Robotics: Science and Systems (RSS)*, 2017.

[9] F. Zhang, J. Leitner, M. Milford, and P. Corke. Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks. In *arXiv 1709.05746*, 2017.

[10] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *CoRL*, 2017.

[11] J. Tremblay, T. To, and S. Birchfield. Falling things: A synthetic dataset for 3D object detection and pose estimation. In *CVPR Workshop on Real World Challenges and New Benchmarks for Deep Learning in Robotic Vision*, June 2018.

[12] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal Computer Vision*, 81(2), 2009.

[13] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[14] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *CVPR*, 2017.

[15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[16] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake. LabelFusion: A pipeline for generating ground truth labels for real RGBD data of cluttered scenes. In *ICRA*, 2018.

[17] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *Intl. Conf. on Advanced Robotics (ICAR)*, 2015.

[18] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, and S. Birchfield. NDDS: NVIDIA deep learning dataset synthesizer, 2018. https://github.com/NVIDIA/Dataset_Synthesizer.

[19] M. Mueller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem. Sim4CV: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, pages 1–18, 2018.

[20] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *CVPR Workshop on Autonomous Driving (WAD)*, 2018.

[21] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *CVPR*, 2014.

[22] 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 *ICCV*, pages 1521–1529, 2017.

[23] 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 *ACCV*, 2012.

[24] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D object pose estimation using 3D object coordinates. In *ECCV*, 2014.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[27] L. Rupert, P. Hyatt, and M. D. Killpack. Comparing model predictive control and input shaping for improved response of low-impedance robots. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015.

[28] E. Jang, S. Vijayanarasimhan, P. Pastor, J. Ibarz, and S. Levine. End-to-end learning of semantic grasping. In *CoRL*, 2017.

[29] 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. *PAMI*, 34(5):876–888, 2012.

[30] K. Pauwels and D. Kragic. SimTrack: A simulation-based framework for scalable real-time object pose detection and tracking. In *IROS*, 2015.

[31] K. Pauwels, L. Rubio, and E. Ros. Real-time pose detection and tracking of hundreds of objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(12), 2016.

[32] H. Tjaden, U. Schwanecke, and E. Schmer. Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. In *ICCV*, pages 124–132, 2017.

[33] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[34] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-DoF object pose from semantic keypoints. In *ICRA*, pages 2011–2018, 2017.

[35] 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 *IROS*, pages 545–551, 2017.

[36] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3D orientation learning for 6D object detection from RGB images. In *ECCV*, 2018.

[37] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi. Discovery of latent 3D keypoints via end-to-end geometric reasoning. In *NIPS*, 2018.

[38] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[39] A. Handa, V. Pătrăucean, V. Badrinarayanan, S. Stent, and R. Cipolla. SceneNet: Understanding real world indoor scenes with synthetic data. In *arXiv 1511.07041*, 2015.

[40] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[41] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[42] W. Qiu and A. Yuille. UnrealCV: Connecting computer vision to unreal engine. In *arXiv 1609.01326*, 2016.

[43] Y. Zhang, W. Qiu, Q. Chen, X. Hu, and A. Yuille. UnrealStereo: A synthetic dataset for analyzing stereo vision. In *arXiv:1612.04647*, 2016.

[44] J. McCormac, A. Handa, and S. Leutenegger. SceneNet RGB-D: 5M photorealistic images of synthetic indoor trajectories with ground truth. In *ICCV*, 2017.

[45] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[46] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016.

[47] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

[48] A. Tsirikoglou, J. Kronander, M. Wrenninge, and J. Unger. Procedural modeling and physically based rendering for synthetic data generation in automotive applications. In *arXiv 1710.06270*, 2017.