Robust Pose Estimation from 3D Keypoints

Maggie Wang

***Abstract*—We present a pipeline to produce robust category-level keypoint detection and pose estimations from RGBD images. These robust detections of keypoints are useful for downstream robot manipulation tasks in domestic and industrial applications. We create an instance segmentation and semantic 3D keypoint detection pipeline using synthetic data of box scenes to generate heatmap and depth predictions of the box corners. We use the partial Procrustes superimposition over all possible correspondences to find an affine transformation of a unit box to fit feasible box shapes and poses to the perception model predictions. We find counterexamples using Monte Carlo sampling in pose space of the entire perception system pipeline. In future work, we will use counterexample-guided data augmentation, adversarial training, and sampling techniques to improve the keypoint model performance. We hope to quantify the distributional robustness over scene graphs of the perception algorithm, which will allow us to detect out-of-distribution scenes in real-world environments.**

I. Introduction

KEYPOINT detection is used in many robotics applications such as robot manipulation and warehouse automation because of its semantic and category-level generality. However, robust keypoint detection can be difficult to achieve due to occlusion, noise, and reflections [1]. Using target object shape information, we can make keypoint detections more robust by detecting outliers and extracting object pose estimations from the keypoints.

Previous work in keypoint detection such as kPAM [2] detect semantic 3D keypoints of mugs to demonstrate pick and place tasks. The pipeline is limited by the amount of real-world data and human annotation [3] necessary to train and test the network model, and the pipeline is created for non-symmetric objects. Because of the difficulty in obtaining massive amounts of labeled data, the perception system may not be robust to rare failure cases. This work seeks to leverage simulation data, which can allow us to put the system under rigorous test before deploying in the real world.

kPAM also does not use prior shape information of the object to make keypoint detections more robust. By focusing on detecting the corner keypoints of boxes, which is highly symmetric, we can use box shape information to estimate pose by finding the smallest error over all correspondences in the Iterative Closest Point (ICP) algorithm [4]. This allows us to disregard outliers if the error is too high. This method can be generalized for other known shapes if the number of points is relatively small for brute-force correspondence matching. If we use scaling ICP, which is an extension of ICP that integrates a scaling matrix, we can also estimate anisotropic target object scale.

While recent development in reinforcement learning (RL) has shown advances in pixel-to-torque learning [5] for robot manipulation, they are limited in their modular transferability. Furthermore, the ability to test the perception subsystem alone can help provide guarantees to the overall system [6], which is important in safety-critical applications.

Our goal is to eliminate failure cases for rigorous and scalable deployment in real-world cases. To achieve this, we can use counterexamples (failure cases) to improve the perception pipeline. We can leverage simulation to develop end-to-end attack algorithms to determine how semantic (i.e., real-world) perturbations, e.g. changes in object pose, lighting condition, and object deformation, affect perception outputs. This work focuses on perturbing box poses to find counterexamples using Monte Carlo search.

Contributions and Outline

The main contribution of this work is the keypoint detection pipeline that uses a fully simulated environment to create ground truth keypoints and poses to compare our predictions against and find semantically meaningful counterexamples. Given an RGBD image, we use a CNN to predict a 3D heatmap. We then extract eight 3D keypoints using non-maximum suppression. We calculate an optimal box pose from the detected keypoints. We run our system under test using a Monte Carlo search to find counterexamples that inform potential failure points in the system.

In Section 2, we discuss related work. In Section 3, we describe the data, network, and pose estimation pipeline. Finally, in Section 4, we show counterexamples found through Monte Carlo search of 10,000 scenes.

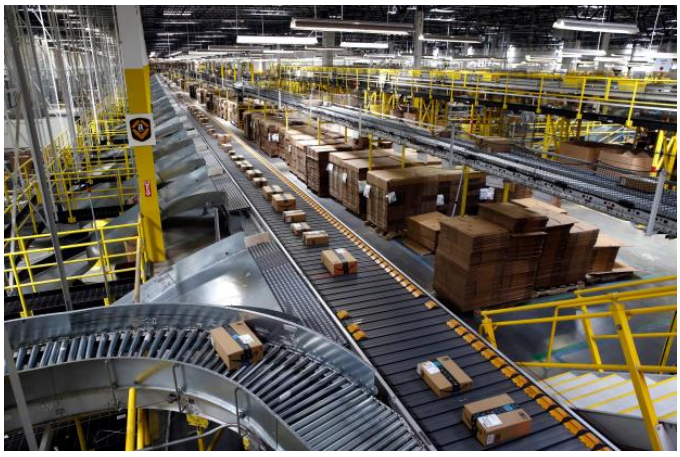


Fig. 1: We would like to use vision to detect corner keypoints and estimate box poses in a scalable, fast, and reliable way for downstream robot manipulation, such as for an Amazon warehouse robot. By using extensive simulation data, we can eliminate many potential failure cases prior to testing in real-world environments.

II. Related Work

We survey two broad categories of pose estimation for robotics: classical geometric pose estimation and deep learning pose estimation. We discuss the shortcomings of each approach and motivate the pose estimation method used in this work, where we use a neural network to create a 3D probability heatmap, non-maximum suppression to extract keypoints, and use partial Procrustes superimposition over all correspondences for pose estimation.

We put the entire perception pipeline under rigorous test through counterexample search. Instead of adversarial attacks in pixel space that lack transferable meaning in physical situations, we make semantically meaningful perturbations in pose space to make an end-to-end attack of the perception pipeline.

A. Pose Estimation in Robotics and Computer Vision

1) Geometric Pose Estimation: Registration algorithms are commonly used for 6D pose estimation in robotics. These techniques use 3D point clouds and prior object shape information to generate pose estimation. RANSAC [7] could work, but the number of permutations is computationally feasible to use an exhaustive approach instead when the number of points is small. TEASER [8] assumes a noisy correspondence generator and is based on many correspondences. Due to its non-convexity and iterative procedure, ICP with poor initialization is susceptible to suboptimal local minima. To bypass the iterative correspondence steps in ICP with poor initialization, we use a small number of keypoints and find the minimum error of all possible correspondences using the partial Procrustes superimposition algorithm.

2) Deep Neural Network-Based Pose Estimation: Learning-based approaches such as PoseCNN [9] and MCN [10] use a Convolutional Neural Network (CNN) to estimate 6D pose estimation. These approaches takes color images as input to give pose estimations. Although these approaches can handle occlusions and symmetric objects, they also require extensive post-processing steps that are slow for real-time applications.

B. Adversarial Attacks

There are three broad classes of adversarial attacks on ML algorithms. Training-time attacks use data poisoning (e.g., model skewing and feedback weaponization) [11] to make the model more robust during training. Model extraction attacks aim to duplicate models or to recover training data via blackbox probing [12]. This work focuses on test-time attacks, or adversarial inputs to a ML algorithm.

Most work on test-time attacks for CNNs focus on pixel-based adversarial attacks. The fast gradient sign method (FGSM) attack [13] aims to make a perturbation that maximizes the loss function to cause the CNN to misclassify the image. The Carlini-Wagner (CW) targeted misclassification attack [14] uses optimization to create adversarial perturbations. Rather than pixel-based attacks, this work focuses on semantic attacks to find failure cases that can occur in real-world environments.

C. Counterexample Search

The FGSM and CW adversarial attacks require white-box knowledge of the target model. However, in nearly all realistic robotic simulations, the renderer is non-differentiable and gradient-free; therefore, we must treat our perception system as a black box.

To find counterexamples, we search for samples where the error between our target and predicted keypoints (which is the output of the 3D heatmap, NMS, and partial Procrustes superimposition steps) is large. In this work, we use a Monte Carlo search for counterexamples. The search for counterexamples could be accelerated using adaptive importance-sampling methods, which [15]. These counterexamples can be used to augment the dataset [16] and expose failure cases in the perception system.

D. Semantic Perturbations

More recently, adversarial attacks have transferred into the semantic space, which modifies the parameter space in simulation to create real-world perturbations (e.g., lighting and pose). In the semantic space, it has been found that varying the pose of objects for CNNs can be extremely effective in attacking the CNN [17], [18], [19]. However, their work in attacking poses in the semantic space leads to mostly physically infeasible configurations. This work thus aims to use a simulator to generate realistic, physically feasible images to generate end-to-end attacks on a CNN classifier.

III. Methods

A. Data Generation

We generate statically stable box scenes using the Drake [20] dynamics simulator and the Blender [21] renderer. The SNOPT solver [22] is used to create a minimum distance constraint between the boxes, and the boxes are dropped from a fixed height and forward-simulated until their velocities fell below a threshold. Figure 2 shows an example cluttered box scene generated using Blender and Drake.

The initial keypoint detection code is based on the open-source code from [2] and is modified to remove ROS dependencies and include a pipeline to train the model on simulated data.

B. Instance Segmentation

We use Mask R-CNN [23] for instance segmentation. Given an image, it returns the object bounding boxes, classes, and masks. For each box object in the scene, we use the bounding box to crop the original image generated by Blender. Figure 3 shows an example of an image cropped by a bounding box.

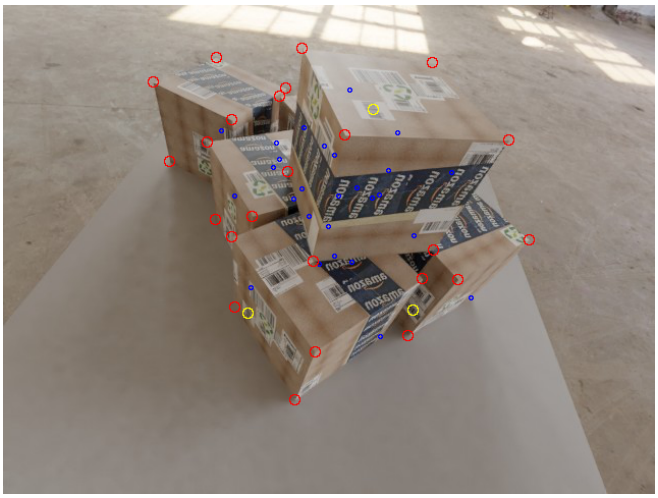
C. Keypoint Model

The keypoint model is a ResNet 34-backbone and integral regression network as described in [2] and [24]. This integral network gives us a probability map that represents the likelihood of a keypoint being at a certain pixel location.

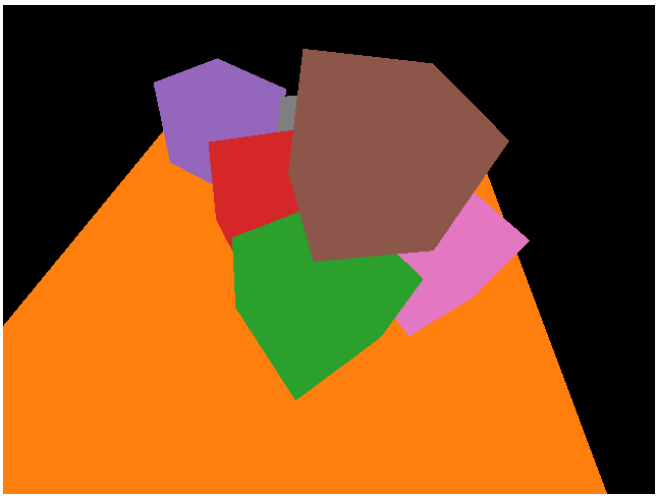
The input to the network is the RGBD image of the cropped region containing a complete box. For training, we create a 3D

Gaussian kernel heatmap from the ground truth keypoints. We use a mean-squared error loss between the target Gaussian kernel heatmap and the network prediction.

The output of the network is a 3D probability heatmap, where each voxel is the probability of a keypoint being at that point, as illustrated in Figure 4.

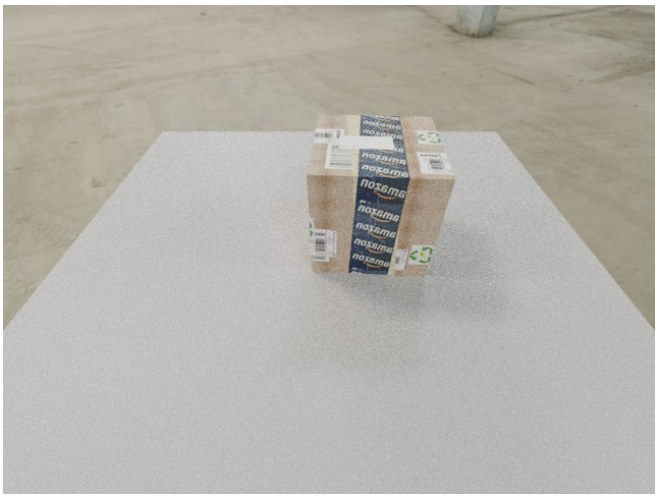


(a) Generated scene using the photorealistic Blender renderer, where the corners are labeled as ground truth keypoints. The red points are visible corners, blue points are occluded corners, and yellow points are additional keypoints indicating visible box label locations that are ignored in this pipeline (but could be used for other tasks).



(b) Generated label image of the same scene using the Drake dynamics simulator, where each object corresponds to a different color.

Fig. 2: Statically stable cluttered box scenes generated in simulation. The boxes were dropped from a fixed height and forward-simulated until stability was achieved. SNOPT is used as a distance constraint solver to ensure physically feasibility.



(a) Original box scene with no cropping. This RGBD image is fed into the Mask R-CNN network to create a bounding box of each object.

(b) Object after bounding box crop (with some tolerance around the edges). This RGBD image is fed into the keypoint NN model.

Fig. 3: Mask R-CNN is used for instance segmentation to find the bounding box of the object. The original image is cropped to contain the object, which allows us to find keypoints only in relevant locations in the scene where there contains an object of interest.

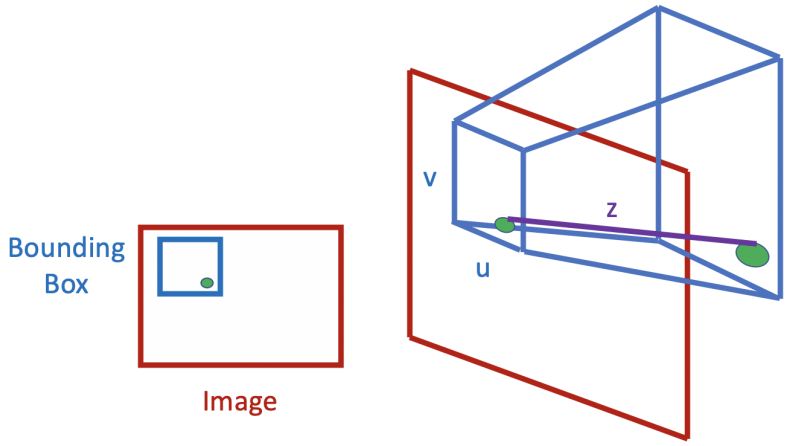


Fig. 4: Original image is cropped with a bounding box from the Mask R-CNN instance. The 3D space is voxelized into a (u, v, z) coordinate system. This image is fed into the keypoint detection model, along with a target 3D Gaussian heatmap of the keypoints. The keypoint detection model uses a mean-squared error loss to learn the 3D Gaussian heatmap.

D. Extracting Box Poses

1) Nonmaximum Suppression (NMS): We first ran non-maximum suppression to turn the 3D heatmap into eight corner keypoints. We find the location with the maximum value in the heatmap and create a Gaussian kernel surrounding that location. This kernel is then subtracted from the entire 3D heatmap. We repeat this process n times to extract the n kernels that represent the keypoints.

2) Search Over All Possible Correspondences Using Least-Squares Fitting: We solve the Procrustes problem by doing least-squares fitting of two 3D point sets [4], which finds the optimal translation and rotation. We calculate the error for all of the possible correspondence sets, and take the correspondence set with the smallest overall error. We brute-force correspondences of all $n!$ permutations (where $n = 8$ for boxes), using only m points (we chose $m = 5$). For small n, this step can be done in a single pass using PyTorch.

For now, we assume that the boxes have unit and constant length.

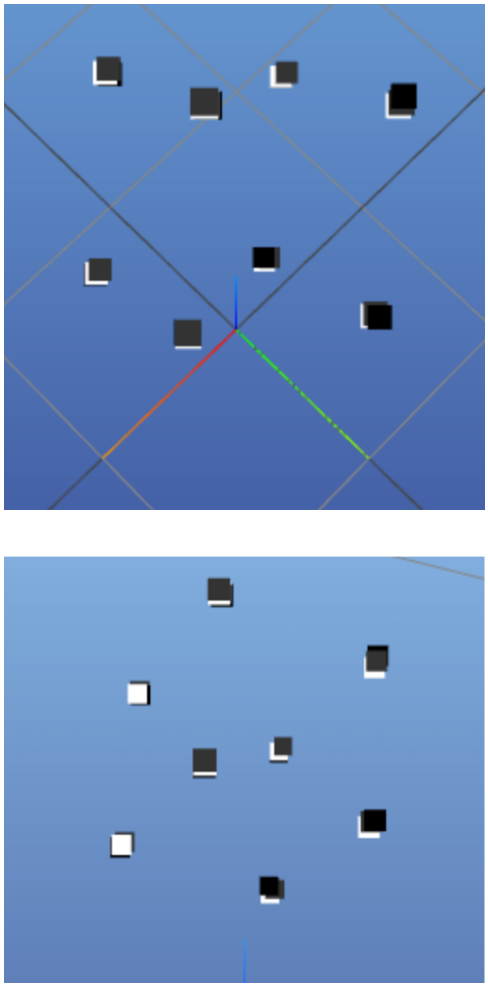


Fig. 5: Example ground truth corner points (white), non-maximum suppression (NMS) output (black), partial Procrustes superimposition output with lowest error (gray). Since the final predicted keypoints match the ground truth keypoints, the error is low.

E. Error Metric for Counterexample Search

We define a counterexample to be a sample that produces a large (e.g., greater than 0.005) error between the nearest-neighbor point-to-point distance between the transformed unit box corner points and ground truth keypoints in camera frame.

The error is given by

$$\text{error} = \frac{\sum_{i=0}^{8} \left\| Rm_{c_i} + t - s_i \right\|_2^2}{8},$$

where m_{c_i} is the unit box corner (where c_i is the integer index correspondence), R is the rotation matrix, t is the transformation vector, and s_i is the corresponding nearest-neighbor ground truth keypoint.

IV. Results

A. Monte Carlo Search for Counterexamples

We use the parallelized Monte Carlo search from [15] to find counterexamples. Running the search with 20 CPU processes for a few hours, we found three counterexamples in 20,000 test images. A histogram of the errors is shown in Figure 6, and the counterexamples are shown in Figures 7, 8, and 9.

B. Discussion

This counterexample search is easily customized for other systems, and the keypoint detection and pose estimation pipeline can be adapted for any object with keypoints. This pipeline is capable of finding interesting failure cases. The pose estimation technique is fairly robust, and the use of simulation to find failure cases allows us to find tail-end failure cases that have a low probability of occurring, yet are still important to be robust against. From Figure 6, we see that the failure cases are extremely rare, yet they are important to find to improve the system. Since it takes around 10,000 runs for a single failure case, we can accelerate this process using adaptive search methods in future work.

In the counterexample found in Figure 7 with 0.1410 mean error, we see that four of the points found from NMS are outliers. Since we chose to use five points for least-squares fitting, the error is large because it is trying to fit all of the points, and one of the points is an outlier. In the counterexample found in Figure 8 with 0.1079 mean error, there are also four outliers found from NMS. Lastly, in the counterexample found in Figure 9 with 0.0754 mean error, the final predictions are slightly off the actual keypoints due to noise in the extracted NMS points. These failure cases are surprising given that they look similar to the normal training data, and the system should be able to correctly locate the keypoints.

This failure case can improve our perception system in a few ways. Firstly, it can show that the network may have a “hole” in this region in the sense that it is unable to give a good 3D heatmap at this box pose. It also shows us limitations in using NMS, as we may lose information while finding the highest probability points. We can try using fewer points when solving for the optimal box rotation and translation–if we chose to use four instead of five points, we would be able to correctly find the correct location of keypoints.

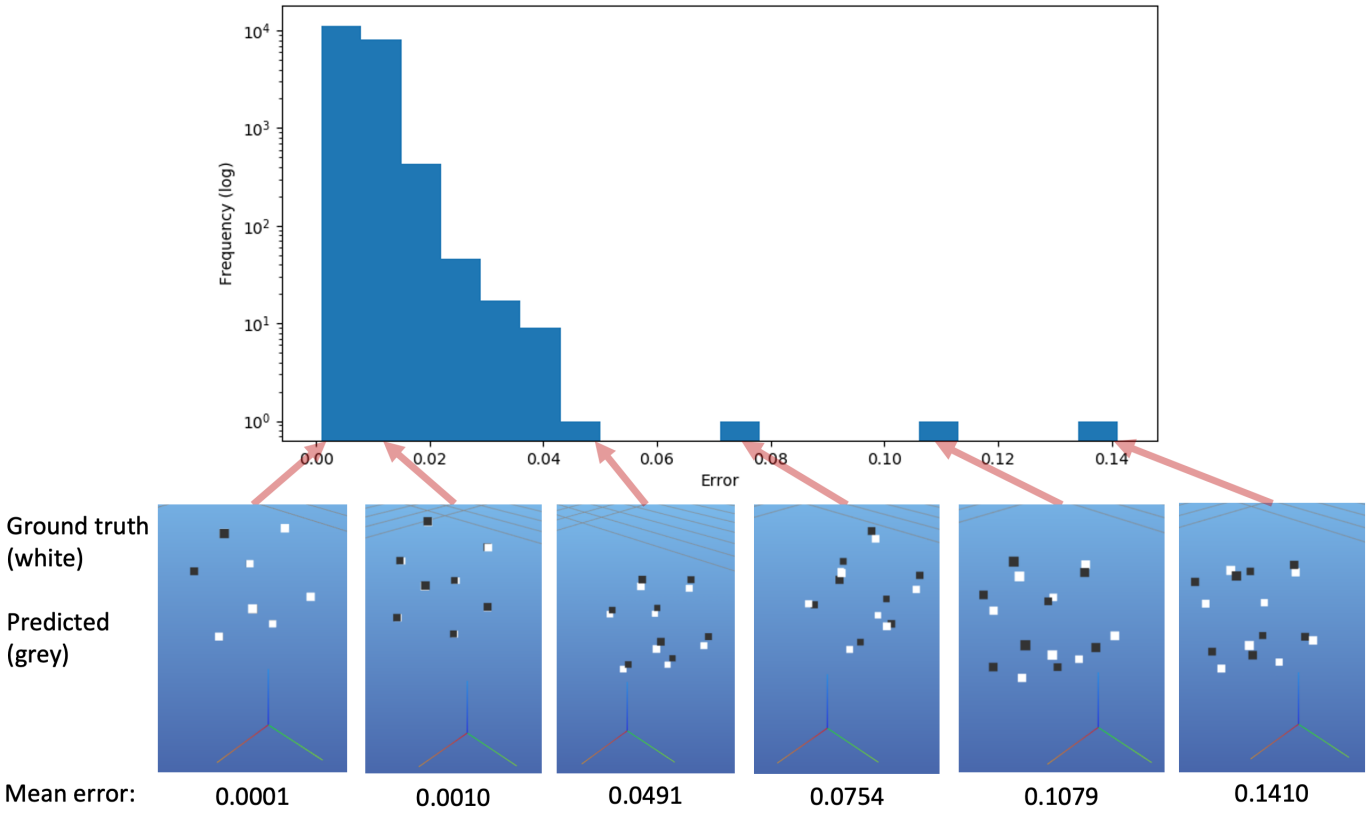
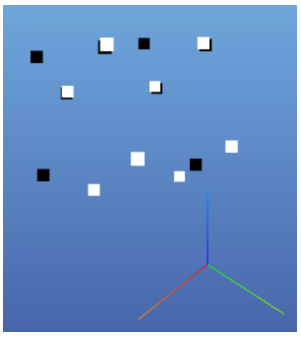


Fig. 6: Log-scaled histogram of errors with 20,000 images. These errors are the mean squared error of the nearest-neighbor point-to-point distance between the transformed unit box corner points and the ground truth keypoints in camera frame. Most of the images are clustered between 0 and 0.02, but there are three counterexamples. From highest to lowest, the counterexamples have mean error of 0.1410, 0.1079, and 0.0754.

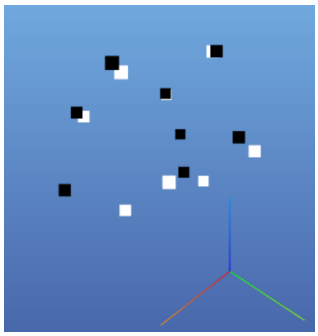
(a) Box that produced output with high error.



(b) Ground truth corner keypoints (white) and points from NMS (black). We see that four of the points found from NMS are outliers.

Fig. 7: Counterexample with mean error of 0.1410. This counterexample exists because four outliers are found by NMS.

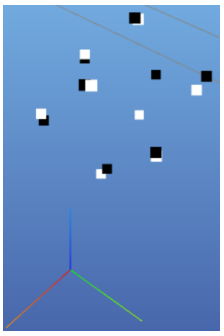
(a) Box that produced output with high error.



(b) Ground truth corner keypoints (white) and points from NMS (black). In this case, there are also four of points found from NMS are outliers.

Fig. 8: Counterexample with mean error of 0.1079. This counterexample exists because four outliers are found by NMS.

(a) Box that produced output with high error.



(b) In this case, there are "only" two outliers, but the other six points still have noise. This causes the final predicted points to not completely match the ground truth points, as shown in Figure 6.

Fig. 9: Counterexample with mean error of 0.0754. While there are fewer outliers in this case, many of the NMS points deviate from the ground truth points a bit.

V. Future Work

We would like to see how well the pipeline performs in cluttered environments such as the boxes in Figure 2. We would also like to incorporate scaling components in the least-squares problem to make a shape estimation of the box (in addition to the current pose estimation).

When naive Monte Carlo search serves to be too inefficient, we will use adaptive Monte Carlo search for failure cases [15]. This approach prioritizes coverage over simply finding a point that minimizes/maximizes a function, thereby speeding up rare-event simulation and find counterexamples with less time and compute.

If the failure in the counterexample comes from a network prediction with a large error, we will fine-tune the neural network with counterexamples.

In the long term, we would like to quantify distributional robustness over scene graphs [25] to detect out-of-distribution scenes [26] in real-world tasks. This would allow us to find failure cases in our perception pipeline prior to running the entire pipeline and using incorrect predictions in downstream manipulation tasks.

VI. Conclusion

We create a box keypoint detection and pose estimation pipeline using a dynamics simulator and photo-realistic graphics render. We use a network that inputs an RGBD image and returns a 3D probability heatmap of box corner locations. We then use non-maximum suppression and least-squares fitting over all correspondences to find the minimum error. This gives us a box pose estimate while ignoring outliers that make the error large. We run this pipeline using Monte Carlo sampling to find counterexamples in over 20,000 scenes. These counterexamples can be used to improve the model and overall perception pipeline.

Acknowledgments

This work was done in the MIT Robot Locomotion Laboratory in collaboration with Gregory Izatt, Tobia Marcucci, and Russ Tedrake.

The code is based on kPAM [27], box scene generation [28], and Trustworthy AI API for counterexample search [29].

References

[1] L. L. S. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Data association for semantic world modeling from partial views," *Int. J. Robotics Res.*, vol. 34, no. 7, pp. 1064–1082, 2015. [Online]. Available: https://doi.org/10.1177/0278364914559754

[2] L. Manuelli, W. Gao, P. R. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," *CoRR*, vol. abs/1903.06684, 2019. [Online]. Available: http://arxiv.org/abs/1903.06684

[3] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3235–3242.

[4] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *CoRR*, vol. abs/1504.00702, 2015. [Online]. Available: http://arxiv.org/abs/1504.00702

[6] P. Kouvaros and A. Lomuscio, "Formal verification of cnn-based perception systems," *CoRR*, vol. abs/1811.11373, 2018. [Online]. Available: http://arxiv.org/abs/1811.11373

[7] C. Papazov and D. Burschka, "An efficient ransac for 3d object recognition in noisy and occluded scenes," *ACCV 2010: Computer Vision – ACCV 2010*, pp. 135–148, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-19315-6_11

[8] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," 2020.

[9] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *CoRR*, vol. abs/1711.00199, 2017. [Online]. Available: http://arxiv.org/abs/1711.00199

[10] C. Li, J. Bai, and G. D. Hager, "A unified framework for multi-view multi-class object pose estimation," *CoRR*, vol. abs/1803.08103, 2018. [Online]. Available: http://arxiv.org/abs/1803.08103

[11] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," *CoRR*, vol. abs/1706.03691, 2017. [Online]. Available: http://arxiv.org/abs/1706.03691

[12] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," *CoRR*, vol. abs/1609.02943, 2016. [Online]. Available: http://arxiv.org/abs/1609.02943

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.

[14] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016. [Online]. Available: http://arxiv.org/abs/1608.04644

[15] M. O'Kelly, A. Sinha, H. Namkoong, J. C. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," *CoRR*, vol. abs/1811.00145, 2018. [Online]. Available: http://arxiv.org/abs/1811.00145

[16] T. Dreossi, S. Ghosh, X. Yue, K. Keutzer, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Counterexample-guided data augmentation," *CoRR*, vol. abs/1805.06962, 2018. [Online]. Available: http://arxiv.org/abs/1805.06962

[17] M. A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W. Ku, and A. Nguyen, "Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects," *CoRR*, vol. abs/1811.11553, 2018. [Online]. Available: http://arxiv.org/abs/1811.11553

[18] J. Mohapatra, T. Weng, P. Chen, S. Liu, and L. Daniel, "Towards verifying robustness of neural networks against semantic perturbations," *CoRR*, vol. abs/1912.09533, 2019. [Online]. Available: http://arxiv.org/abs/1912.09533

[19] H. Hosseini and R. Poovendran, "Semantic adversarial examples," *CoRR*, vol. abs/1804.00499, 2018. [Online]. Available: http://arxiv.org/abs/1804.00499

[20] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[21] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[22] P. E. Gill, W. Murray, M. A. Saunders, and E. Wong, "User's guide for SNOPT 7.7: Software for large-scale nonlinear programming," Department of Mathematics, University of California, San Diego, La Jolla, CA, Center for Computational Mathematics Report CCoM 18-1, 2018.

[23] F. Massa and R. Girshick, "maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch," https://github.com/facebookresearch/maskrcnn-benchmark, 2018, accessed: [Insert date here].

[24] X. Sun, B. Xiao, S. Liang, and Y. Wei, "Integral human pose regression," *CoRR*, vol. abs/1711.08229, 2017. [Online]. Available: http://arxiv.org/abs/1711.08229

[25] G. Izatt and R. Tedrake, "Generative modeling of environments with scene grammars and variational inference," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6891–6897.

[26] J. Chen, Y. Li, X. Wu, Y. Liang, and S. Jha, "Robust out-of-distribution detection for neural networks," 2020.

[27] "kPAM," https://github.com/weigao95/kPAM, accessed: 2021-04-03.

[28] "Scene generation," https://github.com/gizatt/scene_generation, accessed: 2021-04-03.

[29] "Trustworthy AI demo," https://github.com/Trustworthy-AI/trustworthysearch-demo, accessed: 2021-04-03.