
Real world 3D object pose estimation and the Sim2Real gap

Umesh Thillaivasan*

Department of Computer Science
Stanford University
uthillai@stanford.edu

Abstract

Training deep neural networks for robotic manipulation such as pick-and-place tasks requires accurate detection and pose estimation of one or multiple objects. Using synthetic data for training deep neural networks for these tasks can efficiently generate substantial custom labeled and annotated datasets; however, synthetic data present what is known as the Sim2Real gap or reality gap where networks trained on only synthetic data do not generalize well to real-world data. We show the programmatic process of generating custom synthetic datasets using Unreal Engine and the NVIDIA Deep learning Dataset Synthesizer (NDDS) to explore the Sim2Real gap in the context of 6-DoF pose estimation of a custom object from a single RGB image. We present qualitative issues when testing real-world integration with an RGB camera in the form of bounding box heat maps, and we then perform a study on synthetic dataset size versus performance, as well as a sensitivity study to explore the Sim2Sim robustness of our model, testing camera perspectives, lighting, object rotation, and environment randomization. We then present next steps and strategies for bridging the reality gap.

1 Introduction

As artificial intelligence and computer vision advance [1, 2], so does the possibility of training robots to perform repetitive, dangerous, and collaborative tasks [3, 4, 5, 6]. In order for robots to successfully perform actions in an environment, it first needs to be able to understand what objects are in its environment in the form of their 3D position and orientation. The 3D position and orientation of an object is also known as the 6 degrees of freedom (6-DoF) pose [3]. The 6-DoF pose of an object is important for robot-object interaction such as pick-and-place, or path planning for interaction or avoidance. Being able to determine these poses from a single RGB image is also beneficial in simplifying application, hardware required, and costs, as usually multiple cameras (i.e., stereo) or depth cameras (i.e., RGB-D) are required to discern the object distance.

Collecting and annotating real-world 3D object data for pose estimation is both complicated and expensive. Since we cannot manually label 3D data, but need labeled data for training deep learning models for robotic manipulation, this work aims to substitute artificially generated data that can be programmatically annotated and labeled. Once we generate the synthetic data, we explore how well this method transfers from simulation to reality [8].

We test a state-of-the-art object pose estimator, Deep Object Pose Estimation (DOPE) from NVIDIA, which claims to generalize better to novel environments including extreme lighting conditions. Our

*Stanford Center for Professional Development

exploration delves into how well this implementation can generalize beyond benchmark datasets when introducing custom objects for real-world applications, as well as how performance endures when testing our network on not just sim2sim data, but real-world images.

The input to train our neural network is an RGB image of our object class, the pose annotation data file, and the camera intrinsics. Using VGG-19 network pre-trained on ImageNet, we then learn the image features using the first 10 layers, then the network forks to generate the outputs of the network which are belief maps and vector fields named affinity maps.

2 Related work

While it is common to annotate data for 2D object detection [1, 2], it is both complicated and expensive to manually collect and label data for 3D object detection and pose estimation due to the need of camera intrinsic data and needing to map physical 3D positions to a 2D image capture [3, 6, 7]. Researchers have been making progress in various aspects of robotic grasping and manipulation, focused on estimating the 6-DoF pose of objects [4, 5, 6]. Some of this research focuses solely on how to estimate object poses, others on detecting multiple objects in a cluttered scene [8].

DOPE [3] and PoseCNN [5] are two leading methods in estimating object poses. One key difference between DOPE and PoseCNN is that PoseCNN calculates the pose loss and the shape loss, which is computationally expensive and also makes training harder and more unstable. PoseCNN also requires some real-world training images for implementation. DOPE claims a one-shot, deep neural network-based system that can infer in near real-time, as well as is able to be trained on only synthetic data and generalize to real, novel environments. From a network perspective, DOPE uses a multistage approach to first output heat maps of where vertices and the centroid are located, and then process them using a perspective-n-point algorithm to determine the object pose.

Another important area of 3D object detection involves training data. Due to the difficulty of generating real-world labelled data, networks trained on synthetic data has been explored for benchmarking and training purposes. The YCB dataset is a common benchmarking dataset of 21 objects [9]. With training using synthetic data also brings research into bridging the reality gap or Sim2Real gap where networks that are trained on synthetic data do not perform well when encountering real-world data [10, 11, 12, 13, 14, 15].

Our research focuses on one of the two leading methods in estimating object poses, DOPE, and also on the challenge of bridging the Sim2Real gap where we use custom made synthetic training data to train our network, then test its performance against real-world objects.

3 Dataset and Features

A major motivator for this project was to explore how state-of-the-art networks trained on large, benchmark datasets like YCB household objects [9], generalizes to new and real-world object classes. To test this, we created our own custom object, *Tetris Block*, to explore generating synthetic data for training and testing, as well as 3D printed the object to test the real world Sim2Real transfer.

To train with custom objects, DOPE needs data in a particular format defined during the creation of the Falling Things (FAT) dataset [10]. FAT is a collection of synthetic images with ground truth annotations for research in object detection and 3D pose estimation. As described in the FAT dataset documentation, each synthetic data capture consists of “a stereo pair of RGB-D images (i.e., RGB stereo images with ground truth depth for both cameras) and 3D poses, per-pixel semantic segmentation, and 2D/3D bounding box coordinates for all object instances” [3].

To generate our own data, we use three different scenes (arena, table, and free space) and capture between 100 to 2000 images per scenario. We then repeat this six times, each time randomizing the object location within the scene. During each capture, we randomize the domains of lighting (pitch and yaw, shadow diffuseness, intensity, and colour), camera perspective (pitch and yaw, distance from object), environment (scene, rotation), and finally the object (rotation and position). We also generated data with distractor objects that would add variation and occlusions to the scene. We set the distance between the left and right camera as 60 mm to keep it consistent with DOPE’s stereo implementation [3]. With each of the left and right camera perspective captures of the object, a json file is stored which contains the camera data (global coordinate location, and quaternion), and pose

information about the object (class, location and quaternion), which provides the 3D and projected 2D cuboid and projected cuboid, respectively. There are also two general settings files per data generation: camera settings and object settings. The camera settings file includes all of the camera intrinsic values of both left and right virtual cameras. The object setting file includes the object class names, 4x4 transformation matrix, and the class cuboid dimensions. The synthetic data and annotations are generated using a custom plug-in for Unreal Engine 4. Figure 1, shows a sampling of the synthetic data generated of our custom object, Tetris Block.

3.1 NVIDIA Deep Learning Dataset Synthesizer

NVIDIA’s Deep Learning Dataset Synthesize (NDDS) is a custom plug-in for Unreal Engine 4 (UE4) [16]. NDDS allows researchers to export synthetic images with metadata, and supports the randomization of variables such as lighting, objects, camera position, poses, textures, and distractor objects (e.g., cubes, cones).

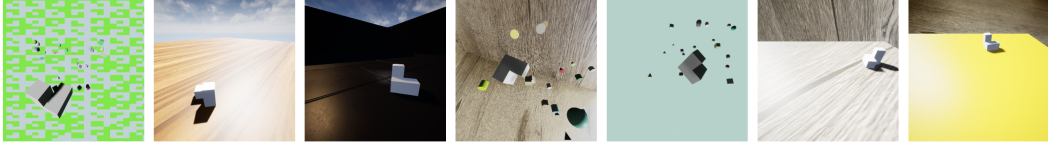


Figure 1: Examples of the 10K generated synthetic training data of custom class, which includes random backgrounds, lighting, environments, camera perspective, and object pose. More in Appendix.

4 Methods

DOPE’s network uses a multistage architecture. The feedforward network inputs a 512x512x3 RGB image of our object class, the pose and orientation annotation data information, and the camera intrinsics. Using VGG-19 network pretrained on ImageNet, we then learn the image features using the first 10 layers, then go into two belief map convolutional layers to reduce the feature dimension to 256 then 128 with a kernel size of 3, stride of 1, padding of 1, and ReLU activation functions after each. The network then forks to output either the belief maps or the vector fields of the object, known as affinity maps. To avoid the vanishing gradients problem, the L2 loss (equation 1) for the belief maps and affinity maps were computed after each stage.

$$L2loss = (y_{true} - y_{pred})^2 \quad (1)$$

4.1 Belief and Affinity Maps

Belief maps are heatmaps of where the network predicts a keypoint such as a vertex or centroid is located. The affinity map is a vector field that each keypoint is associated. Once the network outputs the belief and affinity maps, each vertex-to-centroid vector is compared to the vector field of that vertex and then that vertex is assigned to the closet centroid. With each vertex of the cuboid assigned, they are passed through a perspective-n-point (PnP) algorithm [17] to estimate the 6-DoF of the object by recovering the final translation and rotation of the object with respect to the camera.

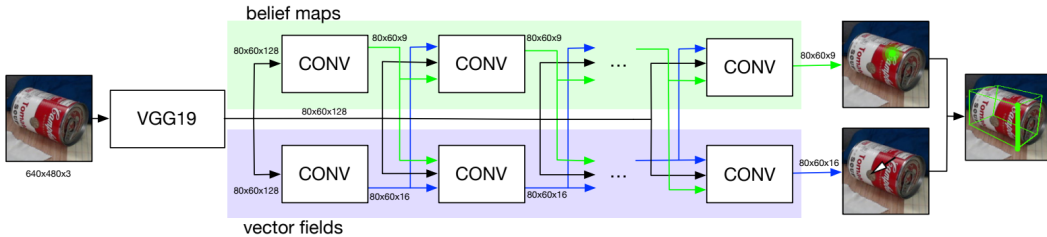


Figure 2: The NVIDIA DOPE network architecture.

5 Experiments/Results/Discussion

5.1 Training

The initial synthetic training dataset size used 3K domain-randomized images that were generated using NDDS as previously described. In addition to the NDDS domain randomization of lighting, camera perspective, object distance and pose, and scene, the network also augments the data by applying Gaussian noise, random contrast, and random brightness, similar to the original DOPE network [3]. Our networks were trained for 40 and 60 epochs, with a batchsize of 6 or 12. Adam was used as the optimizer with learning rate set at 0.0001 to keep hyperparameters consistent with the original paper for initial baselining [18]. After training on only synthetic data, we immediately tested how the network estimated poses on similar simulated data and real-world images. Table 1 shows that the larger the synthetic dataset, the lower the testing loss.

Table 1: Synthetic training dataset size impacts on training and testing performance.

Synthetic dataset size	Training L2 Loss	Testing L2 Loss
2K	2.58 E-8	2.40 E-8
4K	3.00 E-8	1.99 E-8
6K	1.38 E-8	1.32 E-8

5.2 Sim2Real and Sim2Sim

Our initial experiments aimed to confirm if our model was estimating the 6-DoF accurately. We captured a real-world image of our one-to-one scale 3D printed custom object tetris block in the same white colour used for generating the synthetic data as seen in Appendix Figure 4. Our model was unable to produce an accurate pose estimation for the real-world images (Appendix Figure 10). To address this, we compared the belief maps of a test synthetic image and the real world image. The belief map for a synthetic test image showed clear, distinct peaks depicting the estimated vertices, while the belief maps for the real world image showed a noisy heatmap (Appendix Figure 9).

To explore what was causing the extreme heat map noise, we adjusted the threshold parameters for processing the belief maps and that reduced the noise enough for a bounding box to be produced, however the estimated pose was not correct. Digging deeper, we captured another real-world image much closer in appearance to a synthetic test image to see how much or little variation from simulated to real-world data causes our model to fail. The four main variables that we hypothesized could impact a prediction were scene appearance, object orientation, camera perspective, and lighting; however, this closer real-world image also failed to produce a clean belief map. We then cropped out the synthetic object with shadow from the synthetic image and transposed it onto the real-world image background as seen in the Appendix Figure 7, then ran this image through our model. Figure 3 shows the output belief maps and that the model was able to successfully estimate a pose for this hybrid image. This led us to believe that the lighting was impacting the prediction. When comparing the lighting from the synthetic image to the real-world image, the shadows can be seen as very distinct in the synthetic image, and very diffuse or soft and dispersed in the real-world image. To explore Sim2Sim further, we then modified a test synthetic image with Gaussian blur to see if our model would still be able to predict an accurate pose for a novel image. Qualitatively, our model was able to handle Sim2Sim novel images where Gaussian blurs were applied to synthetic data. Additional tests of real-world performance included testing a blue one-to-one 3D printed object, as well as a miniature scaled down version of the object class, both of which failed to produce any meaningful prediction by the model seen in the Appendix Figure 9.

To take a step back, we began to explore how well this network and model handled simulated-to-simulated data generalization but testing the model with novel synthetic images. The model was able to produce meaningful bounding boxes and orientations for these simulated test images.

5.3 Sensitivity Analysis

With positive results from our Sim2Sim exploration, we wanted to understand if and by how much any of the four key variables (scene, camera, lighting, object orientation) impacted model performance

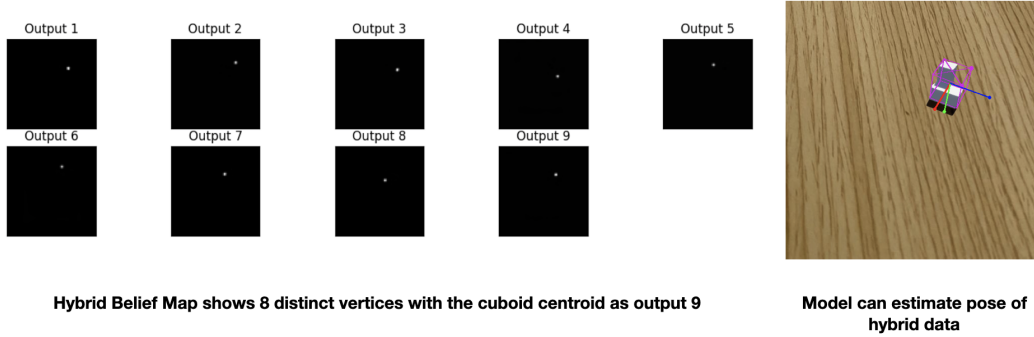


Figure 3: Hybrid belief map and predicted cuboid and orientation.

the most in the form of a sensitivity analysis. We generated 10K synthetic images for each variable where only that variable was randomized in the dataset generation. We then trained on each of these datasets and used a separate test set to evaluate performance. These results (Table 2) indicate that lighting impacts model performance the most of the four variables, while the room and object pose were magnitudes less detrimental. This matches our qualitative experience and intuition as our model suffered the most due to the Sim2Real versus real-world lighting deviation.

Table 2: Sensitivity analysis of four primary variables randomized for synthetic data generation.

Variable	Training Loss	Testing Loss
Lighting	5.62 E-4	6.71 E-3
Camera	4.74 E-4	6.46 E-3
Room	1.37 E-8	1.81 E-8
Object	4.85 E-7	8.62 E-7

6 Conclusion/Future Work

Detecting and estimating the 6-DoF pose of a known object is an exciting research area in the field of computer vision and robotics. We present results on how a state-of-the-art network handles new custom object classes that are trained using labelled data generated using a domain randomization and photorealistic data pipeline. The DOPE network estimates the vertices of each object’s cuboid bounding box and projects that into 2D, called a belief map. These estimated vertices and centroid are passed through a PnP algorithm for generating the prediction cuboid pose estimation.

We concluded that while this network was able to produce a substantial amount of domain randomized labelled data for the purpose of 6-DoF detection, the network still had difficulty spanning the Sim2Real gap. Our real-world object failed to be detected even when conditions such as lighting, object orientation, camera perspective, and scene textures were nearly replicated. After identifying that the real-world lighting was too extreme for the belief maps to identify the pose cuboid, we performed a sensitivity analysis to see which of the four primary variables might impact performance most. The results were that the lighting then camera perspective generated the greatest loss by several magnitudes. This should be considered for future work where more attention can be taken to producing photo realistic data that accounts for the sensitivity to lighting.

Future work includes generating more photo realistic data and obtaining more computational resources to be able to train on significantly larger synthetic datasets. Additionally, we would like to begin introducing multiple custom object classes, stress testing on how to handle symmetric objects, and testing how the model performs with mixed scenes where multiple object poses need to be detected and estimated. We would also like to evaluate PoseCNN, another leading method for 6-DoF pose estimation, in a similar way we analyzed DOPE. Finally, we would like to explore a 3D printed object with markers on the object vertices and perform a real-world image pixel-to-pixel distance calculation to the project bounding boxes to establish a methodology for quantifying how real-world images without annotated ground truths can be used to evaluate model testing.

References

- [1] Dai, Jifeng, et al. "R-fcn: Object detection via region-based fully convolutional networks." *Advances in neural information processing systems*. 2016.
- [2] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
- [3] Tremblay, Jonathan, et al. "Deep object pose estimation for semantic robotic grasping of household objects." *arXiv preprint arXiv:1809.10790* (2018).
- [4] Rad, Mahdi, and Vincent Lepetit. "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.
- [5] Xiang, Yu, et al. "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes." *arXiv preprint arXiv:1711.00199* (2017).
- [6] Tekin, Bugra, Sudipta N. Sinha, and Pascal Fua. "Real-time seamless single shot 6d object pose prediction." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
- [7] Zhao, Wenshuai, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey." *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020.
- [8] Hinterstoisser, Stefan, et al. "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes." *Asian conference on computer vision*. Springer, Berlin, Heidelberg, 2012.
- [9] Calli, Berk, et al. "Yale-CMU-Berkeley dataset for robotic manipulation research." *The International Journal of Robotics Research* 36.3 (2017): 261-268.
- [10] Tremblay, Jonathan, Thang To, and Stan Birchfield. "Falling things: A synthetic dataset for 3d object detection and pose estimation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018.
- [11] Eslami, SM Ali, et al. "Neural scene representation and rendering." *Science* 360.6394 (2018): 1204-1210.
- [12] Behl, Harkirat Singh, et al. "Autosimulate:(quickly) learning synthetic data generation." *European Conference on Computer Vision*. Springer, Cham, 2020.
- [13] Doersch, Carl, and Andrew Zisserman. "Sim2real transfer learning for 3d human pose estimation: motion to the rescue." *Advances in Neural Information Processing Systems* 32 (2019): 12949-12961.
- [14] Tremblay, Jonathan, et al. "Training deep networks with synthetic data: Bridging the reality gap by domain randomization." *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018.
- [15] Tobin, Josh, et al. "Domain randomization for transferring deep neural networks from simulation to the real world." *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017.
- [16] To, Thang, Tremblay et al. "NVIDIA Deep Learning Dataset Synthesizer." Github (2018), https://github.com/NVIDIA/Dataset_Synthesizer
- [17] Lepetit, Vincent, Francesc Moreno-Noguer, and Pascal Fua. "Epnnp: An accurate o(n) solution to the pnp problem." *International journal of computer vision* 81.2 (2009): 155.
- [18] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).
- [19] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019): 8026-8037.

Appendix

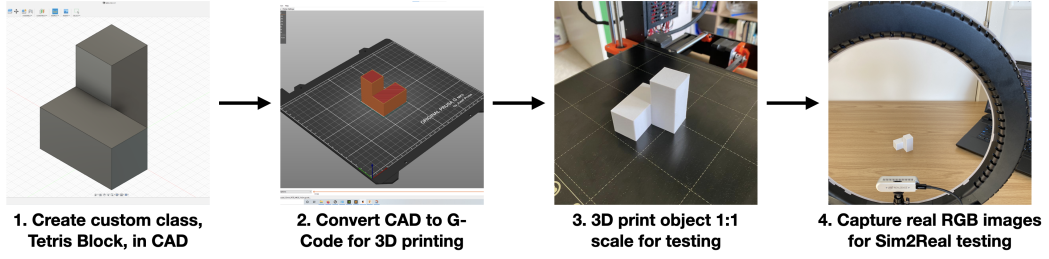


Figure 4: Custom tetris class object generation.

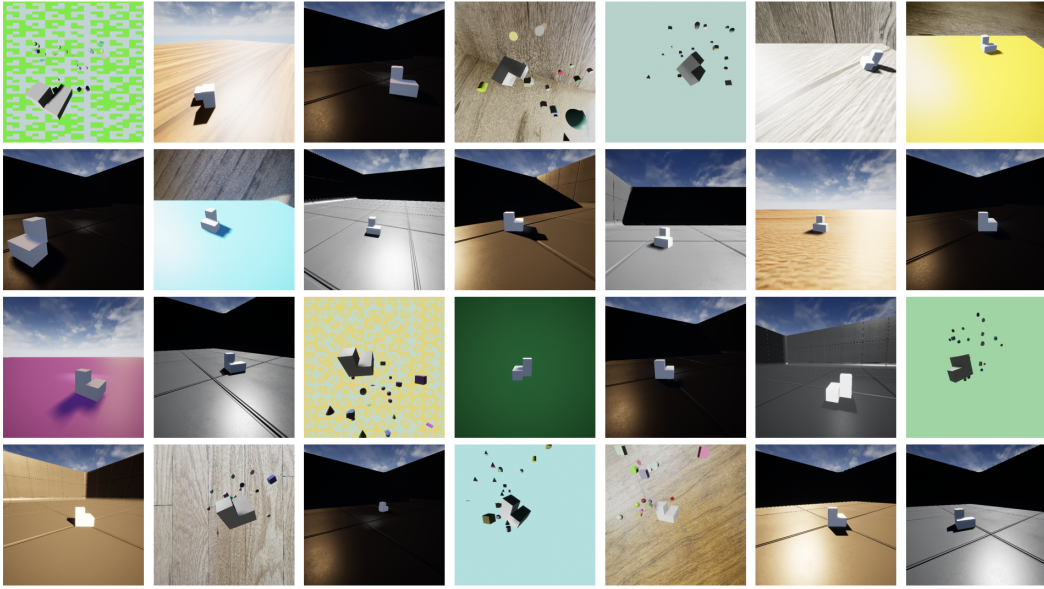
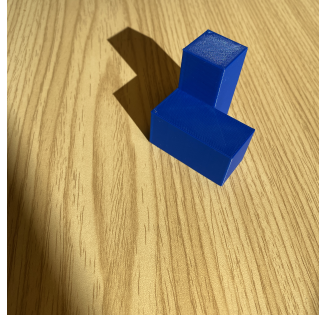


Figure 5: Examples of the 10K generated synthetic training data of custom class, which includes random backgrounds, lighting, environments, camera perspective, and object pose.



(a) 1:1 3D printed tetris object for real-world testing.

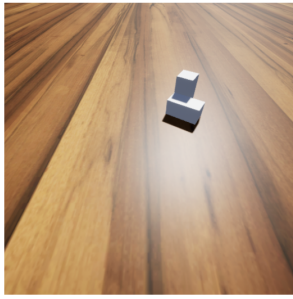


(b) 1:1 blue color 3D printed tetris object for real-world testing.

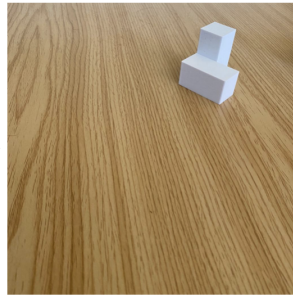


(c) 1:10 scaled down tetris object for testing model generalization.

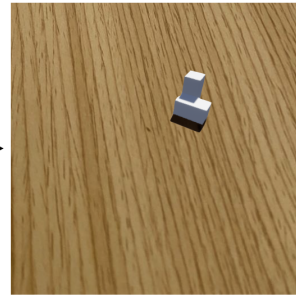
Figure 6: Three successful model predictions on synthetic data.



Synthetic Data

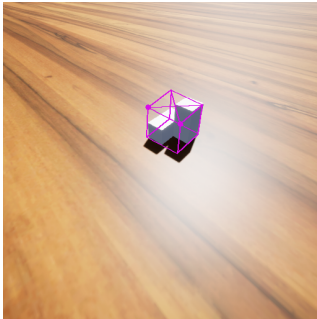


Real-world capture to replicate Synthetic Data

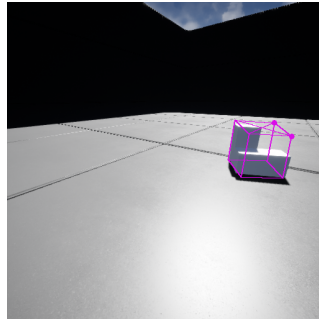


Hybrid Data moving synthetic object onto real-world background.

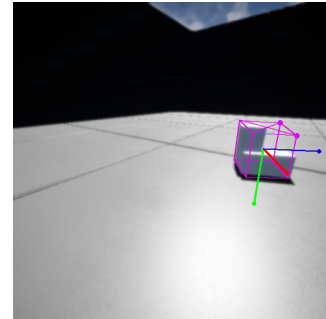
Figure 7: Generation of hybrid data where the synthetic object is overlaid the real-world background.



(a) Successful cuboid prediction on test data.



(b) Successful cuboid prediction on test data.



(c) Sim2Sim prediction after applying Gaussian blur to test data.

Figure 8: Three successful model predictions on synthetic data.

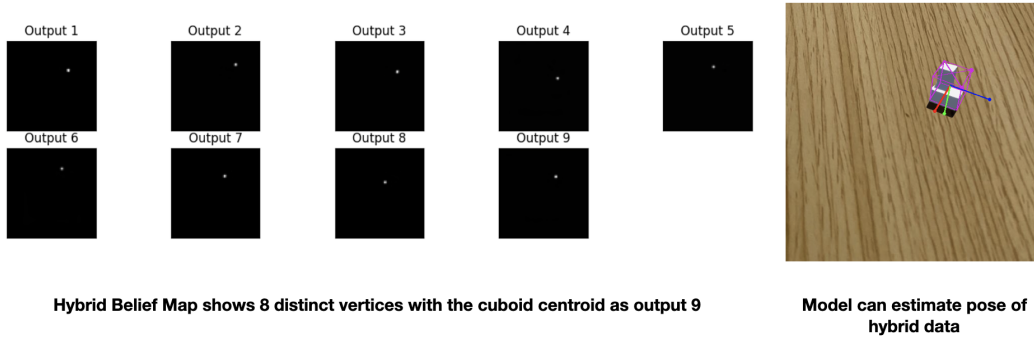


Figure 9: Comparison of a clear belief map for synthetic test data with accurate pose estimation.

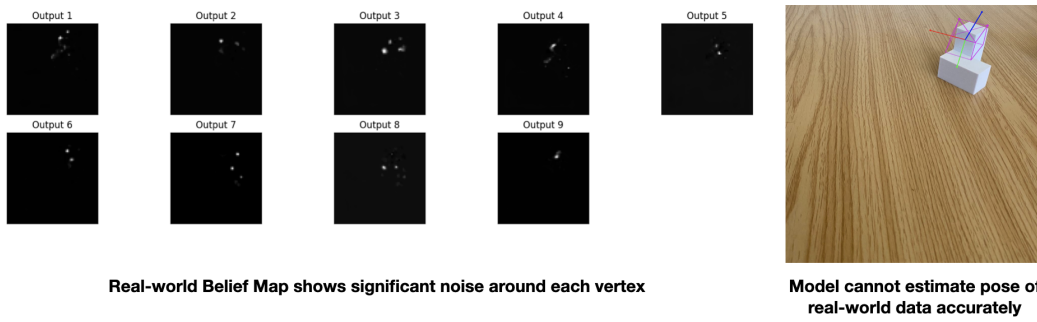


Figure 10: The output belief map of real-world test data showing noisy potential vertices with the erroneous output.

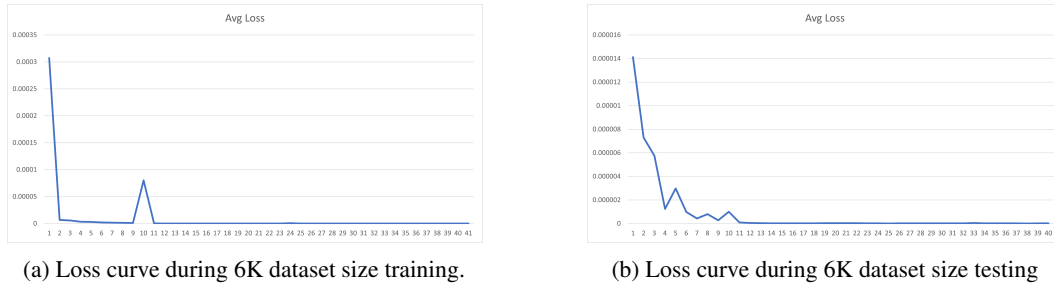


Figure 11: Loss curves for 6K dataset size.

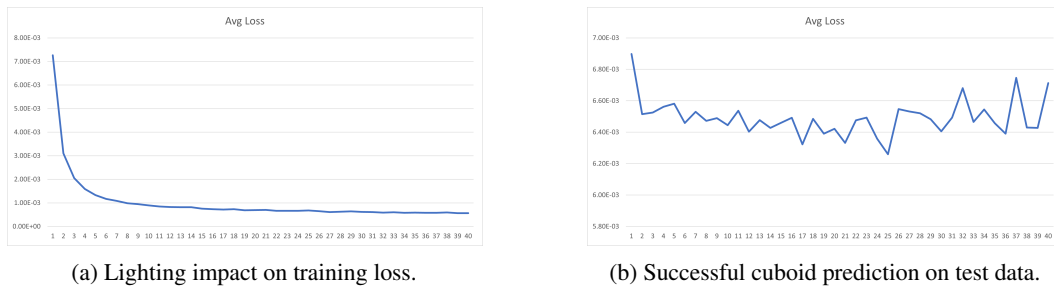


Figure 12: Sensitivity analysis results of most impactful variable, lighting.