VI. APPENDIX

A. Implementation Details

In this section, we will provide more details about the implementation of the algorithms described in Sec. III.

1) Online Self-supervised Learning: The online self-supervised learning pipeline is described in Sec. III-C and Algorithm I. Here we will provide more details for the algorithm. The Finetune() function is called after every 32 finetuning examples of \((I, h)\) are added to the finetuning set \(F\). Within each Finetune(), the detection network is finetuned on the all examples in \(F\) for a single epoch. Similar to [1], we use AMS-Grad [2] for optimization with a learning rate of \(10^{-4}\), a weight decay of \(10^{-6}\) and a batch size of 8.

2) Transductive Learning: The transductive learning pipeline simulates the scenarios where all the testing images are known before the network makes any inference and thus the network can be self-supervised trained on the testing images with more time budgets. Specifically, we run the zero-shot pose estimator on the uncropped images in the testing dataset first and use the pose estimates to train the detector in a regular epoch training fashion. In this way the pose estimation stage is the same as ZePhyR [3] and we can simply take the results from [3] as pseudo ground truth in the implementation. For training, we initialize the weights of the detector as described in Sec. IV-A and finetune it for 50 epochs on the pseudo ground truth. We use the same optimizer as in the online learning pipeline, and shrink the learning rate to its tenth at the epoch 20 and 40.

Here we further report the training time using the transductive learning protocol. Since the network is only trained on the testing dataset, which contains 1445 images for LM-O and 4123 images for YCB-V. Therefore the network training time for transductive learning only takes 50 epochs is roughly 25 minutes for LM-O and 72 minutes for YCB-V. This demonstrates that the proposed pipeline is capable to adapt to new environments and novel objects in a short time.

B. Synthetic Data Generation and Training

Since YCB-V objects were already used for training in [1], we need to re-train the DTOID network using another dataset in order to show the generalization ability to novel objects. Therefore we generated a synthetic dataset using BlenderProc [4]. We adopted the objects from BOP datasets [5] except for those from LM, LM-O and YCB-V and additionally used 200 ShapeNet objects [6] with randomized CC0 textures [7]. The scenes were generated by randomly dropping objects onto a table and images were captured at randomly sampled camera poses. In this way, we produced a dataset of 40,000 images and trained the detection network from random weights for 100 epochs. We used the same optimizer and scheduler as described in Sec. VI-A.2.

Note that our DTOID weights did not reproduce the zero-shot detection performance as reported in [1]. However, the performance of our DTOID model quickly adapts to YCB-V objects as shown in Sec. IV-D and Table I.

C. Comparison to Non Zero-shot Detector

To analyze the benefit of zero-shot networks in our pipeline, we tested our self-supervised learning framework where the DTOID detection network is replaced by a non zero-shot detector, specifically Mask R-CNN [8] with the ResNet-50 [9] backbone pretrained on MS COCO dataset [10]. The results on the LM-O dataset are shown in Table III. We found that in the transductive learning setting, Mask R-CNN can yield similar pose estimation performance as DTOID, but worse detection results. In the online learning setting, DTOID shows much better performance than Mask R-CNN. The reason might be that we need a much larger dataset to train a non zero-shot object detector, while zero-shot detectors like DTOID are designed to quickly adapt to new objects.

D. Qualitative Results

In Fig. 6, we show some qualitative results of the detector during the progress of the online self-supervised learning pipeline. Here we recorded the model weights after it is trained on different portion of the test dataset and compare their performance. We can see that the performance gradually improves and the previously missed or false detection are corrected as the online self-supervised learning continues.
Fig. 6. Qualitative results of the online self-supervised learned detector, after seeing different number of images. Green, blue and red boxes mean correct, false and missed detection results respectively. The leftmost column shows the results of the baseline detection model and the others show the performance after the detector has been trained on different portion of the test set. The first three rows are the results from the LM-O dataset and the last three rows are from the YCB-V dataset.
REFERENCES


