

Supplementary Material for Markerless Camera-to-Robot Pose Estimation via Self-supervised Sim-to-Real Transfer

Jingpei Lu, Florian Richter, and Michael C. Yip
University of California, San Diego
{jil360, frichter, yip}@ucsd.edu

1. Generate Synthetic Training Data

Setting up a pipeline for generating robot masks and keypoints can be complicated or simple depending on the choice of the simulator. However, an interface for acquiring the robot pose with respect to a camera frame must exist for every robot simulator. Therefore, instead of generating ground-truth labels for the robot mask and keypoints directly from a simulator, we only save the robot pose and configuration pair for each synthetic image, and the labels for the robot mask and keypoint are generated on-the-fly during the training process.

Given the ground-truth robot pose with respect to the camera frame $\tilde{\mathbf{T}}_b^c$ and robot configuration \mathbf{q} , the keypoint labels $\tilde{\mathbf{o}}_i$ can be generated through the projection operation:

$$\tilde{\mathbf{o}}_i = \pi(\mathbf{p}_i | \tilde{\mathbf{T}}_b^c, \mathbf{K}) \quad (1)$$

where \mathbf{K} is the camera intrinsic matrix and \mathbf{p}_i is calculated using robot forward kinematics with known robot joint angles \mathbf{q} as ???. For generating the robot mask, we apply a silhouette renderer \mathcal{R} which is described in Section 3.1. Given the ground-truth robot pose, the robot mask is generated as:

$$\tilde{\mathbb{S}} = \mathcal{R}(\tilde{\mathbf{T}}_b^c | \mathbf{K}) \quad (2)$$

where $\tilde{\mathbb{S}}$ is the ground-truth label for robot mask.

For generating the synthetic images, similar to Domain Randomization [2], we also randomize a few settings of the robot simulator so that the generated samples can have some varieties. Specifically, we randomize the following aspects:

- Robot joint configuration.
- The camera position, which applies the look-at method so that the robot is always in the viewing frustum.
- The number, position, and intensity of the scene lights.
- The position of the virtual objects.
- The background of the images.
- The color of robot mesh.

We also perform image augmentation with additive white Gaussian noise. The synthetic data for the Baxter is generated using CoppeliaSim¹. For the Panda, the synthetic dataset provided from [1] is used for training.

2. Details on Position-based Visual Servoing

The diagram of the position-based visual servoing system is shown in the Fig. 1. In our experimental setup, the camera is not necessarily stationary and the goal pose, \mathbf{T}_g^c , defined in the camera frame, is also changing over time. Therefore, for each control loop, we need to compute the camera-to-robot pose \mathbf{T}_b^c to update the goal pose in the robot base frame:

$$\mathbf{T}_g^b = [\mathbf{T}_b^c]^{-1} \mathbf{T}_g^c. \quad (3)$$

Given the current joint angle reading and image, CtRNet outputs the camera-to-robot pose. The goal pose is a pre-defined trajectory in the camera frame. For each loop, we take the camera-to-robot pose estimation to transform the goal end-effector pose from the camera frame to the robot base frame, and the goal joint configuration is computed via inverse kinematics. Then, a joint controller is employed to minimize the joint configuration error by taking a step toward the goal joint configuration. The camera was running at 30Hz and the joint controller was running at 120Hz. All of the communication between sub-modules was done using the Robot Operating System (ROS), and everything ran on a single computer with an Intel® Core™ i9 Processor and NVIDIA’s GeForce RTX 3090.

For qualitative analysis, we provide 2 experiment setups. At first, we fixed the goal pose at the camera center with a depth of one meter. In the second experiment, we set the goal pose following a circle centered at $[0, 0, 1]$, with a radius of 0.15 meters, in the camera frame. For both experiments, we manually moved the camera, and the results are presented in the supplementary videos.

¹<https://www.coppeliarobotics.com/>

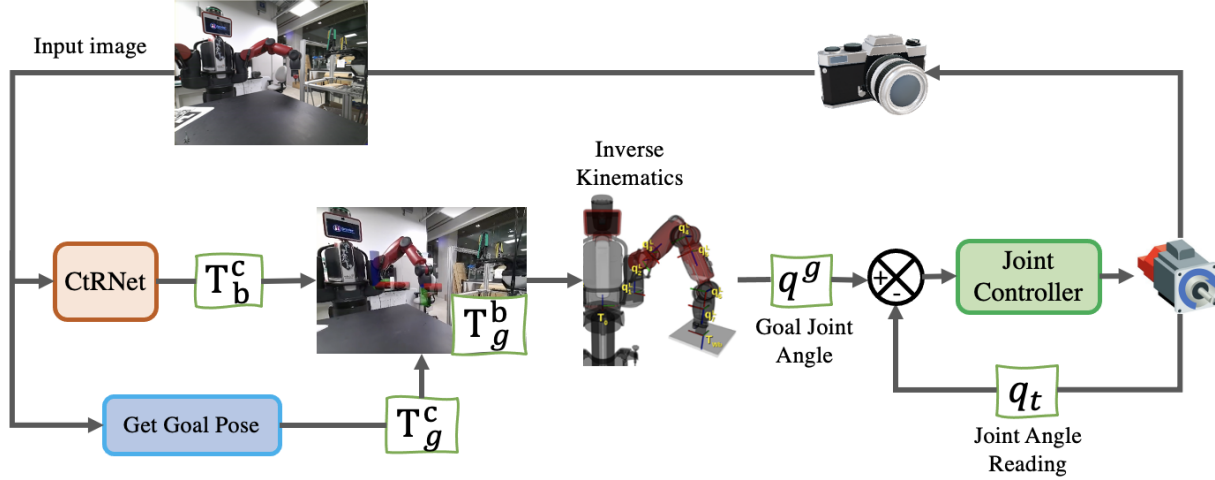


Figure 1. The diagram of the position-based visual servoing system. Given the current joint angle reading and image, CtRNet outputs the camera-to-robot pose. After transforming the goal end-effector pose to the robot base frame, the goal joint configuration is computed via inverse kinematics. Then, a joint controller is employed to minimize the error between the current and goal joint configuration.

$N_{pretrain}$	Mean ADD (mm) ↓		AUC ADD ↑	
	w/ \mathcal{L}_{seg}	w/o \mathcal{L}_{seg}	w/ \mathcal{L}_{seg}	w/o \mathcal{L}_{seg}
500	2167.30	184.45	47.62	65.26
1000	92.91	172.34	76.65	69.71
2000	67.51	104.69	82.98	76.20
4000	63.00	84.67	84.12	78.71
8000	63.81	83.21	83.93	79.07

Table 1. Ablation study on the impact of segmentation loss.

3. Evaluation on Foreground Segmentation

In this section, we evaluate the performance of foreground segmentation before and after the self-supervised training of the CtRNet. As the real-world datasets do not have segmentation labels, we evaluate the segmentation performance qualitatively on the DREAM-real dataset. The qualitative results are shown in Fig. 2, where we show sample segmentation masks before and after the self-supervised training, together with the original RGB images. Notably, the robot mask exhibits enhanced quality after self-supervised training, preserving fine details of corners and boundaries. We believe the high-quality segmentation mask is the key to achieving SOTA performance on robot pose estimation, which provides close-to-ground-truth supervision for the keypoint detector.

4. Segmentation Accuracy Impact on Pose Estimation

The performance of the robot pose estimation relies on the accuracy of the foreground segmentation, as the segmentation mask offers image-level guidance during the self-supervised training phase. In this section, we investigate the influence of segmentation accuracy on the performance of robot pose estimation. To analyze this, we carry out experiments using the Baxter dataset by training the neural network with and without the segmentation loss \mathcal{L}_{seg} . We use different numbers of synthetic training samples to pre-train the neural network as described in Section 4.4 in the main text. Throughout the self-supervised training phase, only the mask loss \mathcal{L}_{mask} is utilized to train the keypoint detector, while refraining from fine-tuning the segmentation layers with segmentation loss \mathcal{L}_{seg} .

We report the mean ADD and AUC ADD for robot pose estimation in Tab. 1, and include the results obtained with segmentation loss \mathcal{L}_{seg} for comparative purposes. We observed that training with segmentation loss consistently yields better performance by a considerable margin, given enough pretrained samples. We also discovered a similar trend as in Section 4.4: having more pretraining samples improves the performance but the improvement is saturated after having more than 4000 training samples. However, when limited to a low number of pretraining samples (≤ 500), CtRNet performs better without using segmentation loss because the segmentation layers are not affected by the large numbers of inaccurately detected keypoints.

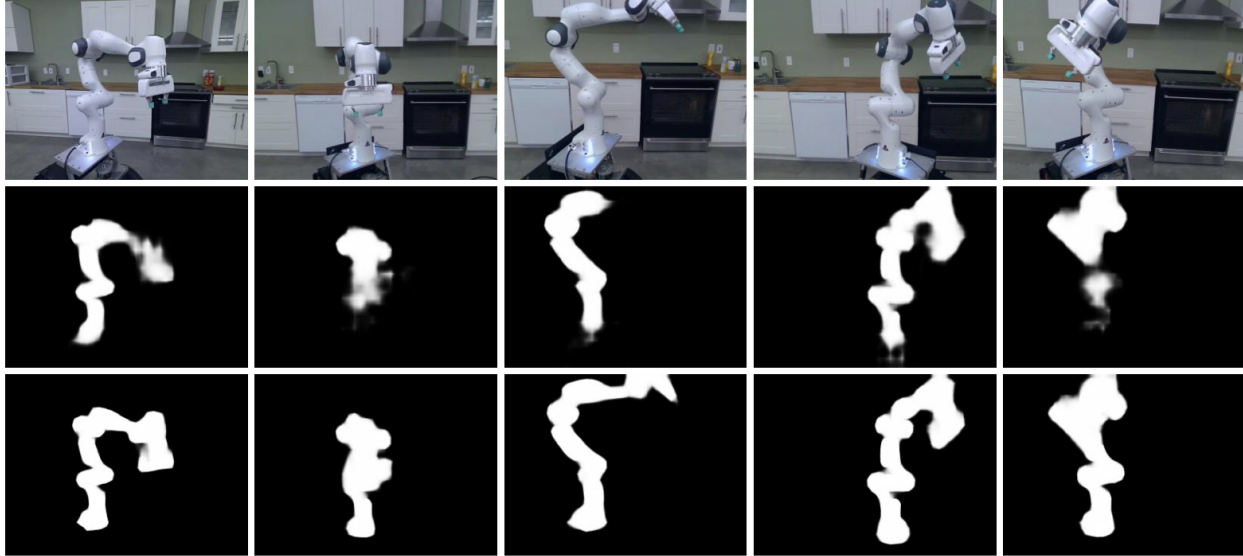


Figure 2. Qualitative results of CtRNet foreground segmentation before and after self-supervised training. From top to bottom row shows the RGB images, the segmentation masks before, and after self-supervised training.

5. Limitation

The keypoint detection can only detect the points in the image frame. In some cases, part of the robot body is out of the camera frustum and hence does not appear in the image frame. This will result in false positives in keypoint detection, which undermines the performance of the robot pose estimation. However, the false positives in keypoint usually result in a pose that has a large reprojection error. Therefore, the foreground segmentation would not be affected by these bad samples as the weights are close to zero, according to Equation 6 in the main text. During the inference, we can also use the reprojection error to indicate the confidence of the pose estimation.

References

- [1] Timothy E Lee, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Oliver Kroemer, Dieter Fox, and Stan Birchfield. Camera-to-robot pose estimation from a single image. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9426–9432. IEEE, 2020. [1](#)
- [2] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017. [1](#)