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

Abstract

Solving the camera-to-robot pose is a fundamental requirement for vision-based robot control, and is a process that takes considerable effort and cares to make accurate. Traditional approaches require modification of the robot via markers, and subsequent deep learning approaches enabled markerless feature extraction. Mainstream deep learning methods only use synthetic data and rely on Domain Randomization to fill the sim-to-real gap, because acquiring the 3D annotation is labor-intensive. In this work, we go beyond the limitation of 3D annotations for real-world data. We propose an end-to-end pose estimation framework that is capable of online camera-to-robot calibration and a self-supervised training method to scale the training to unlabeled real-world data. Our framework combines deep learning and geometric vision for solving the robot pose, and the pipeline is fully differentiable. To train the Camera-to-Robot Pose Estimation Network (CtRNet), we leverage foreground segmentation and differentiable rendering for image-level self-supervision. The pose prediction is visualized through a renderer and the image loss with the input image is back-propagated to train the neural network. Our experimental results on two public real datasets confirm the effectiveness of our approach over existing works. We also integrate our framework into a visual servoing system to demonstrate the promise of real-time precise robot pose estimation for automation tasks.

1. Introduction

The majority of modern robotic automation utilizes cameras for rich sensory information about the environment to infer tasks to be completed and provide feedback for closed-loop control. The leading paradigm for converting the valuable environment information to the robot's frame of reference for manipulation is position-based visual servoing (PBVS) [4]. At a high level, PBVS converts 3D environmental information inferred from the visual data (e.g.
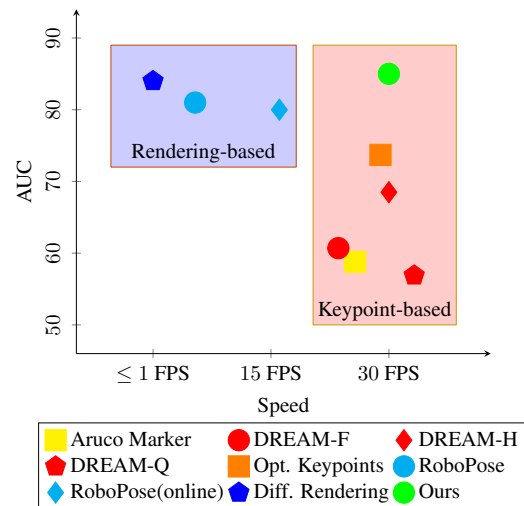

Figure 1. Comparison of speed and accuracy (based on AUC metric) for existing image-based robot pose estimation methods.

the pose of an object to be grasped) and transforms it to the robot coordinate frame where all the robot geometry is known (e.g. kinematics) using the camera-to-robot pose. Examples of robotic automation using the PBVS range from bin sorting [35] to tissue manipulation in surgery [31].

Calibrating camera-to-robot pose typically requires a significant amount of care and effort. Traditionally, the camera-to-robot pose is calibrated with externally attached fiducial markers (*e.g.* Aruco Marker [14], AprilTag [38]). The 2D location of the marker can be extracted from the image and the corresponding 3D location on the robot can be calculated with forward kinematics. Given a set 2D-3D correspondence, the camera-to-robot pose can be solved using Perspective-n-Point (PnP) methods [13, 30]. The procedure usually requires multiple runs with different robot configurations and once calibrated, the robot base and the camera are assumed static. The incapability of online calibration limits the potential applications for vision-based robot control in the real world, where minor bumps or simply shifting due to repetitive use will cause calibrations to be thrown off,

not to mention real-world environmental factors like vibration, humidity, and temperature, are non-constant. Having flexibility on the camera and robot is more desirable so that the robot can interact with an unstructured environment.

Deep learning, known as the current state-of-the-art approach for image feature extraction, brings promising ways for markerless camera-to-robot calibration. Current approaches to robot pose estimation are mainly classified into two categories: keypoint-based methods [27–29,34,43] and rendering-based methods [16,26]. Keypoint-based methods are the most popular approach for pose estimation because of the fast inference speed. However, the performance is limited to the accuracy of the keypoint detector which is often trained in simulation such that the proposed methods can generalize across different robotic designs. Therefore, the performance is ultimately hampered by the sim-to-real gap, which is a long-standing challenge in computer vision and robotics [55].

Rendering-based methods can achieve better performance by using the shape of the entire robot as observation, which provides dense correspondence for pose estimation. The approaches in this category usually employ an iterative refinement process and require a reasonable initialization for the optimization loop to converge [32]. Due to the nature that iteratively render and compare is time- and energy-consuming, rendering-based methods are more suitable for offline estimation where the robot and camera are held stationary. In more dynamic scenarios, such as a mobile robot, the slow computation time make the rendering-based methods impracticable to use.

In this work, we propose CtRNet, an end-to-end framework for robot pose estimation which, at inference, uses keypoints for the fast inference speed and leverages the high performance of rendering-based methods for training to overcome the sim-to-real gap previous keypoint-based methods faced. Our framework contains a segmentation module to generate a binary mask of the robot and keypoint detection module which extracts point features for pose estimation. Since segmenting the robot from the background is a simpler task than estimating the robot pose and localizing point features on robot body parts, we leverage foreground segmentation to provide supervision for the pose estimation. Toward this direction, we first pretrained the network on synthetic data, which should have acquired essential knowledge about segmenting the robot. Then, a self-supervised training pipeline is proposed to transfer our model to the real world without manual labels. We connect the pose estimation to foreground segmentation with a differentiable renderer [24,33]. The renderer generates a robot silhouette image of the estimated pose and directly compares it to the segmentation result. Since the entire framework is differentiable, the parameters of the neural network can be optimized by back-propagating the image loss.

Contributions. Our main contribution is the novel framework for image-based robot pose estimation together with a scalable self-training pipeline that utilizes unlimited real-world data to further improve the performance *without any manual annotations*. Since the keypoint detector is trained with image-level supervision, we effectively encompass the benefits from both keypoint-based and rendering-based methods, where previous methods were divided. As illustrated in the Fig. 1, our method maintains high inference speed while matching the performance of the rendering-based methods. Moreover, we integrate the CtRNet into a robotic system for PBVS and demonstrate the effectiveness on real-time robot pose estimation.

2. Related Works

2.1. Camera-to-Robot Pose Estimation

The classical way to calibrate the camera-to-robot pose is to attach the fiducial markers [14,38] to known locations along the robot kinematic chain. The marker is detected in the image frame and their 3D position in the robot base frame can be calculated with forward kinematics. With the geometrical constraints, the robot pose can be then derived by solving an optimization problem [11,20,22,39].

Early works on markerless articulated pose tracking utilize a depth camera for 3D observation [10,37,41,45]. For a high degree-of-freedom articulated robot, Bohg *et al*. [1] proposed a pose estimation method by first classifying the pixels in depth image to robot parts, and then a voting scheme is applied to estimate the robot pose relative to the camera. This method is further improved in [52] by directly training a Random Forest to regress joint angles instead of part label. However, these methods are not suitable for our scenario where only single RGB image is available.

More recently, as deep learning becomes popular in feature extraction, many works have been employing deep neural networks for robot pose estimation. Instead of using markers, a neural network is utilized for keypoint detection, and the robot pose is estimated through an optimizer (*e.g*. PnP solver) [28,29,34,56]. To further improve the performance, the segmentation mask and edges are utilized to refine the robot pose [16,27]. Labbé *et al*. [26] also introduces the *render&compare* method to estimate the robot pose by matching the robot shape. These methods mainly rely on synthetically generated data for training and hope the network can generalize to the real world by increasing the variance in data generation. Our method explicitly deals with the sim-to-real transfer by directly training on real-world data with self-supervision.

2.2. Domain Adaptation for Sim-to-Real Transfer

In computer vision and robotics, Domain Randomization (DR) [48] is the most widely used method for sim-

to-real transfer due to its simplicity and effectiveness. The idea is to randomize some simulation parameters (e.g. camera position, lighting, background, etc.) and hope that the randomization captures the distribution of the real-world data. This technique has been applied to object detection and grasping [2, 18, 21, 47, 49], and pose estimation [26, 28, 29, 34, 36, 46, 50, 56]. The randomization is usually tuned empirically hence it is not efficient.

Another popular technique for domain transfer is Domain Adaptation (DA), which is to find the feature spaces that share a similar distribution between the source and target domains [51]. This technique has shown recent success in computer vision [7, 19, 44] and robotic applications [3, 15, 23]. In this work, instead of finding the latent space and modeling the distribution between the simulation and the real world, we perform sim-to-real transfer by directly training on the real-world data via a self-training pipeline.

3. Methods

In this paper, we introduce an end-to-end framework for robot pose estimation and a scalable training pipeline to improve pose estimation accuracy on real-world data without the need for any manual annotation. We first explain the self-supervised training pipeline for sim-to-real transfer in Sec. 3.1 given a pretrained CtRNet on synthetic data which both segments the robot and estimates its pose from images. Then, we detail the camera-to-robot pose estimation network in Sec. 3.2 which utilizes a keypoint detector and a PnP solver to estimate the pose of the robot from image data in real-time.

3.1. Self-supervised Training for Sim-to-Real Transfer

The most effective way to adapt the neural network to the real world is directly training the network on real sensor data. We propose a self-supervised training pipeline for sim-to-real transfer to facilitate the training without 3D annotations. To conduct the self-supervised training, we employ foreground segmentation to generate a mask of the robot, f_{seg}, alongside the pose estimation, f_{pose}. Given an input RGB image from the physical world, $\mathbb{I}$, and the robot joint angles, $\mathbf{q}$, f_{pose} estimates the robot pose which is then transformed to a silhouette image through a differentiable renderer. Our self-supervised objective is to optimize neural network parameters by minimizing the difference between the rendered silhouette image and the mask image. We formulate the optimization problem as:

$$\theta_{bb}, \theta_{kp}, \theta_{seg} = \underset{\theta_{bb}, \theta_{kp}, \theta_{seg}}{\arg\min} \mathcal{L}[f_{seg}(\mathbb{I}|\theta_{bb}, \theta_{seg}),$$
$$\mathcal{R}(f_{pose}(\mathbb{I}|\mathbf{q}, \theta_{bb}, \theta_{kp})|\mathbf{K})] \quad (1)$$

where $\theta_{bb}, \theta_{kp}, \theta_{seg}$ denote the parameters of the backbone, keypoint, and segmentation layers of the neural network. $\mathcal{R}$ is the differentiable renderer with camera parameters $\mathbf{K}$, and $\mathcal{L}(.)$ is the objective loss function capturing the image difference.

We pretrained CtRNet's parameters which makes up f_{seg} and f_{pose}, with synthetic data where the keypoint and segmentation labels are obtained freely (details in Supplementary Materials). During the self-training phase, where CtRNet learns with real data, the objective loss in (1) captures the difference between the segmentation result and the rendered image. The loss is iteratively back-propagated to, Θ, where each iteration f_{seg} and f_{pose} take turns learning from each other to overcome the sim-to-real gap.

Overview. The overview of the self-supervised training pipeline is shown in the Fig. 2. The segmentation module, f_{seg}, simply takes in a robot image and outputs its mask. The pose estimation module, f_{pose}, consists of a keypoint detector and a PnP solver to estimate the robot pose using the 2D-3D point correspondence, as shown in Fig. 3. Given the input robot image and joint angles, our camera-to-robot pose estimation network outputs a robot mask and the robot pose with respect to the camera frame. Mathematically, these functions are denoted as

$$\mathbb{M} = f_{seg}(\mathbb{I}|\theta_{bb}, \theta_{seg}) \qquad \mathbf{T}_b^c = f_{pose}(\mathbb{I}|\mathbf{q}, \theta_{bb}, \theta_{kp}) \quad (2)$$

where $\mathbb{M}$ is the robot mask and $\mathbf{T}_b^c \in SE(3)$ is the 6-DOF robot pose. Finally, the self-supervised objective loss in (1) is realized through a differentiable renderer, $\mathcal{R}$, which generates a silhouette image of the robot given its pose, $\mathbf{T}_b^c$.

Differentiable Rendering. To render the robot silhouette image, we utilize the PyTorch3D differentiable render [42]. We initialize a perspective camera with intrinsic parameters $\mathbf{K}$ and a silhouette renderer, which does not apply any lighting nor shading, is constructed with a rasterizer and a shader. The rasterizer applies the fast rasterization method [42] which selects the k nearest mesh triangles that effects each pixel and weights their influence according to the distance along the z-axis. Finally, the *SoftSilhouette-Shader* is applied to compute pixel values of the rendered image using the sigmoid blending method [33].

We construct the ready-to-render robot mesh by connecting the CAD model for each robot body part using its forward kinematics and transforming them to the camera frame with the estimated robot pose $\mathbf{T}_b^c$ from f_{pose}. Let $\mathbf{v}^n \in \mathbb{R}^3$ be a mesh vertex on the n-th robot link. Each vertex is transformed to the camera frame, hence ready-to-render, by

$$\overline{\mathbf{v}}^c = \mathbf{T}_b^c \mathbf{T}_n^b(\mathbf{q}) \overline{\mathbf{v}}^n \quad (3)$$

where $\overline{\cdot}$ represents the homogeneous representation of a point (e.g. $\overline{\mathbf{v}} = [\mathbf{v}, 1]^T$), and $\mathbf{T}_n^b(\mathbf{q})$ is the coordinate frame transformation obtained from the forward kinematics [8].
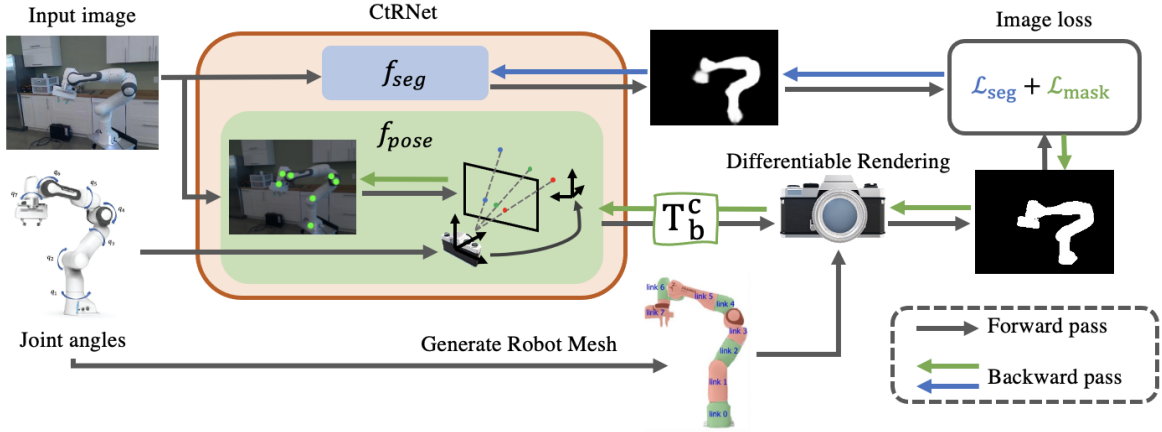
Figure 2. The overview of our proposed self-supervised training framework for sim-to-real transfer. The CtRNet contains a foreground segmentation module and a pose estimation module, which output a robot mask and a camera-to-robot pose respectively. The output pose is transformed into a silhouette image through a differentiable renderer. The image loss is back-propagated to train the keypoint detector and fine-tune the segmentation.

Objective loss function. The objective loss in (1) is iteratively minimized where f_{seg} and f_{pose} take turns supervising each other on real data to overcome the sim-to-real gap faced by keypoint detection networks. To optimize f_{pose}, the L2 image loss is used since the segmentation network's accuracy, within the context of estimating robot poses, has been shown to effectively transfer from simulation to the real world [26]. Mathematically the loss is expressed as

$$\mathcal{L}_{mask} = \sum_{i=1}^{H} \sum_{j=1}^{W} (\mathbb{S}(i,j) - \mathbb{M}(i,j))^2 \quad (4)$$

where H and W is the height and width of the image, and $\mathbb{S}$ is the rendered silhouette image.

Although the pretrained robot segmentation, f_{seg}, already performs well on real-world datasets, it is still desirable to refine it through self-supervised training to better extract fine details of corners and boundaries. To prevent the foreground segmentation layers from receiving noisy training signals, we apply the weighted Binary Cross Entropy Loss so that the high-quality rendering image can be used to further refine the foreground segmentation:

$$\mathcal{L}_{seg} = -\frac{w}{H * W} \sum_{i=1}^{H} \sum_{j=1}^{W} [\mathbb{M}(i,j) \log \mathbb{S}(i,j) \\ + (1 - \mathbb{M}(i,j)) \log(1 - \mathbb{S}(i,j))]. \quad (5)$$

where w is the weight for the given training sample. For PnP solvers, the optimal solution should minimize the point reprojection error. Therefore, we assign the weight for each training sample according to the reprojection error:

$$w = exp\left(-sO(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c)\right) \quad (6)$$
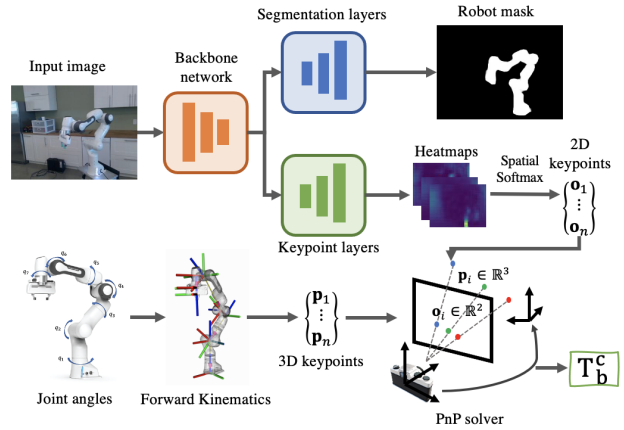

Figure 3. The diagram of the camera-to-robot pose estimation network (CtRNet) which describes the inference process of network. Given an input RGB image, the neural network generates a robot mask and a set of keypoint. Given the associated robot joint angles, a set of corresponding 3D keypoints are computed with forward kinematics. The camera-to-robot pose is estimated by a PnP solver with provided 2D-3D keypoint pairs.

where s is a scaling constant, O is the reprojection loss in the PnP solver (explained in Sec. 3.2), $\{\mathbf{o}_i | \mathbf{o}_i \in \mathbb{R}^2\}_{i=1}^n$ and $\{\mathbf{p}_i | \mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^n$ are the 2D-3D keypoints inputted into the PnP solver. The exponential function is applied to the weight such that training samples with poor PnP convergence are weighted exponentially lower than good PnP convergence thereby stabilizing the training.

3.2. Camera-to-Robot Pose Estimation Network

The overview of the proposed Camera-to-Robot Pose Estimation Network, CtRNet, is shown in Fig. 3. Given an

input RGB image, we employ ResNet50 [17] as the backbone network to extract the latent features. The latent features are then passed through the Atrous Spatial Pyramid Pooling layers [6] to form the segmentation mask of input resolution. The keypoint detector, sharing the backbone network with the foreground segmentation, upsamples the feature maps through transposed convolutional layers and forms the heatmaps with n channels. Then, we apply the spatial softmax operator [12] on the heatmaps, which computes the expected 2D location of the points of maximal activation for each channel and results in a set of keypoints $[\mathbf{o}_1, ..., \mathbf{o}_n]$ for all n channels. For simplicity, we define the set of keypoints at each joint location of the robot. Given the joint angles, the corresponding 3D keypoint location $\mathbf{p}_i$ can be calculated with robot forward kinematics:

$$\overline{\mathbf{p}}_i = \mathbf{T}_i^b(\mathbf{q})\overline{\mathbf{t}}, \; for \; i = 1, ..., n \qquad (7)$$

where $\mathbf{t} = [0, 0, 0]$. With the 2D and 3D corresponding keypoints, we can then apply a PnP solver [30] to estimate the robot pose with respect to the camera frame.

Back-propagation for PnP Solver. A PnP solver is usually self-contained and not differentiable as the gradient with respect to the input cannot be derived explicitly. Inspired by [5], the implicit function theorem [25] is applied to obtain the gradient through implicit differentiation. Let the PnP solver be denoted as followed in the form of a non-linear function g:

$$\mathbf{T}_b^{c*} = g(\mathbf{o}, \mathbf{p}, \mathbf{K}) \qquad (8)$$

where $\mathbf{T}_b^{c*}$ is output pose from the PnP solver. In order to back-propagate through the PnP solver for training the keypoint detector, we are interested in finding the gradient of the output pose $\mathbf{T}_b^{c*}$ with respect to the input 2D points $\mathbf{o}$. Note that, the objective of the PnP solver is to minimize the reprojection error, such that:

$$\mathbf{T}_b^{c*} = \underset{\mathbf{T}_b^c}{\arg\min} \, O(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c) \qquad (9)$$

with

$$O(\mathbf{T}_b^c, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c) = \sum_{i=1}^n ||\mathbf{o}_i - \pi(\mathbf{p}_i | \mathbf{T}_b^c, \mathbf{K})||_2^2 \qquad (10)$$

$$= \sum_{i=1}^n ||\mathbf{r}_i||_2^2 \qquad (11)$$

where $\pi(.)$ is the projection operator. Since the optimal solution $\mathbf{T}_b^{c*}$ is a local minimum for the objective function $O(\mathbf{o}, \mathbf{p}, \mathbf{T}_b^c, \mathbf{K})$, a stationary constraint of the optimization process can be constructed by taking the first order derivative of the objective function with respect to $\mathbf{T}_b^c$:

$$\frac{\partial O}{\partial \mathbf{T}_b^c}(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c)|_{\mathbf{T}_b^c = \mathbf{T}_b^{c*}} = \mathbf{0}. \qquad (12)$$

Following [5], we construct a constrain function F to employ the implicit function theorem:

$$F(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c) = \frac{\partial O}{\partial \mathbf{T}_b^c}(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^{c*}) = \mathbf{0}. \qquad (13)$$

Substituting the Eq. (10) and Eq. (11) to Eq. (13), we can derive the constraint function as:

$$F(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c) = \sum_{i=1}^n \frac{\partial ||\mathbf{r}_i||_2^2}{\partial \mathbf{T}_b^c} \qquad (14)$$

$$= -2 \sum_{i=1}^n \mathbf{r}_i^T \frac{\partial \pi}{\partial \mathbf{T}_b^c}(\mathbf{p}_i | \mathbf{T}_b^{c*}, \mathbf{K}). \qquad (15)$$

Finally, we back-propagate through the PnP solver with the implicit differentiation. The gradient of the output pose with respect to the input 2D points is the Jacobian matrix:

$$\frac{\partial g}{\partial \mathbf{o}}(\mathbf{o}, \mathbf{p}, \mathbf{K})$$
$$= -\left(\frac{\partial F}{\partial \mathbf{T}_b^c}(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c)\right)^{-1}\left(\frac{\partial F}{\partial \mathbf{o}}(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c)\right). \qquad (16)$$

4. Experiments

We first evaluate our method on two public real-world datasets for robot pose estimation and compare it against several state-of-the-art image-based robot pose estimation algorithms. We then conduct an ablation study on the pre-training procedure and explore how the number of pre-training samples could affect the performance of the self-supervised training. Finally, we integrate the camera-to-robot pose estimation framework into a visual servoing system to demonstrate the effectiveness of our method on real robot applications.

4.1. Datasets and Evaluation Metrics

DREAM-real Dataset. The DREAM-real dataset [29] is a real-world robot dataset collected with 3 different cameras: Azure Kinect (AK), XBOX 360 Kinect (XK), and RealSense (RS). This dataset contains around 50K RGB images of Franka Emika Panda arm and is recorded at (640×480) resolution. The ground-truth camera-to-robot pose is provided for every image frame. The accuracy is evaluated with average distance (ADD) metric [54],

$$ADD = \frac{1}{n}\sum_{i=1}^n ||\widetilde{\mathbf{T}}_b^c \overline{\mathbf{p}}_i - \mathbf{T}_b^c \overline{\mathbf{p}}_i||_2 \qquad (17)$$

where $\widetilde{\mathbf{T}}_b^c$ indicates the ground-truth camera-to-robot pose. We also report the area-under-the-curve (AUC) value,

Method	Category	Backbone	Panda 3CAM-AK		Panda 3CAM-XK		Panda 3CAM-RS		Panda ORB		All	
			AUC ↑	Mean (m) ↓	AUC ↑	Mean (m) ↓	AUC ↑	Mean (m) ↓	AUC ↑	Mean (m) ↓	AUC ↑	Mean (m) ↓
DREAM-F [29]	Keypoint	VGG19	68.912	11.413	24.359	491.911	76.130	2.077	61.930	95.319	60.740	113.029
DREAM-Q [29]	Keypoint	VGG19	52.382	78.089	37.471	54.178	77.984	0.027	57.087	67.248	56.988	59.284
DREAM-H [29]	Keypoint	ResNet101	60.520	0.056	64.005	7.382	78.825	0.024	69.054	25.685	68.584	17.477
RoboPose [26]	Rendering	ResNet34	76.497	0.024	**85.926**	**0.014**	76.863	0.023	80.504	**0.019**	80.094	**0.020**
CtRNet	Keypoint	ResNet50	**89.928**	**0.013**	79.465	0.032	**90.789**	**0.010**	**85.289**	0.021	**85.962**	**0.020**

Table 1. Comparison of our methods with the state-of-the-art methods on DREAM-real datasets using ADD metric. We report the mean and AUC of the ADD on each dataset and the overall accuracy.

Method	Category	PCK (2D)		Mean 2D Err.	ADD (3D)		Mean 3D Err.
		@50 pixel ↑	AUC ↑	(pixel) ↓	@100 mm ↑	AUC ↑	(mm) ↓
Aruco Marker [14]	Keypoint	0.49	57.15	286.98	0.30	43.45	2447.34
DREAM-Q [29]	Keypoint	0.33	44.01	1277.33	0.32	40.63	386.17
Opt. Keypoints [34]	Keypoint	0.69	75.46	49.51	0.47	65.66	141.05
Diff. Rendering	Rendering	0.74	78.60	42.30	0.78	81.15	74.95
CtRNet	Keypoint	**0.99**	**93.94**	**11.62**	**0.88**	**83.93**	**63.81**

Table 2. Comparison of our methods with the state-of-the-art methods on Baxter dataset.

which integrates the percentage of ADD over different thresholds. A higher AUC value indicates more predictions with less error.

Baxter Dataset. The Baxter dataset [34] contains 100 RGB images of the left arm of Rethink Baxter collected with Azure Kinect camera at (2048×1526) resolution. The 2D and 3D ground-truth end-effector position with respect to the camera frame is provided. We evaluate the performance with the ADD metric for the end-effector. We also evaluate the end-effector reprojection error using the percentage of correct keypoints (PCK) metric [34].

4.2. Implementation details

The entire pipeline is implemented in PyTorch [40]. We initialize the backbone network with ImageNet [9] pretrained weights, and we train separate networks for different robots. The number of keypoints n is set to the number of robot links and the keypoints are defined at the robot joint locations. The neural network is pretrained on synthetic data for foreground segmentation and keypoint detection for 1000 epochs with 1e-5 learning rate. We reduce the learning rate by a factor of 10 once learning stagnates for 5 epochs. The Adam optimizer is applied to optimize the network parameters with the momentum set to 0.9. For self-supervised training on real-world data, we run the training for 500 epochs with 1e-6 learning rate. The same learning rate decay strategy and Adam optimizer is applied here similar to the pretraining. To make the training more stable, we clip the gradient of the network parameters at 10. The scaling factor in Eq. (6) is set to 0.1 for DREAM-real dataset and 0.01 for Baxter dataset, mainly accounting for

the difference in resolution.

4.3. Robot Pose Estimation on Real-world Datasets

Evaluation on DREAM-real Dataset. The proposed CtRNet is trained at (320×240) resolution and evaluated at the original resolution by scaling up the keypoints by a factor of 2. Some qualitative results for foreground segmentation and pose estimation are shown in the Fig. 4a. We compared our method with the state-of-the-art keypoint-based method DREAM [29] and the rendering-based method RoboPose [26]. The results for DREAM and RoboPose are compiled from the implementation provided by [26]. In Tab. 1, we report the AUC and mean ADD results on DREAM-real dataset with 3 different camera settings and the overall results combining all the test samples. Our method has a significantly better performance compared to the method in the same category and achieves comparable performance with the rendering-based method. We outperform DREAM on all settings and outperform RoboPose on the majority of the dataset. Overall on DREAM-real dataset, we achieve higher AUC (+17.378 compared to DREAM, +5.868 compared to RoboPose), and lower error compared to DREAM (-17.457).

Evaluation on Baxter Dataset. For the Baxter dataset, we trained the CtRNet at (640×480) resolution and evaluate at the original resolution, and Fig. 4b shows some of the qualitative results. We compared our method with several keypoint-based methods (Aruco Marker [14], DREAM [29], Optimized Keypoints [34]). We also implemented Differentiable Rendering for robot pose estimation, where the robot masks are generated with the pre-
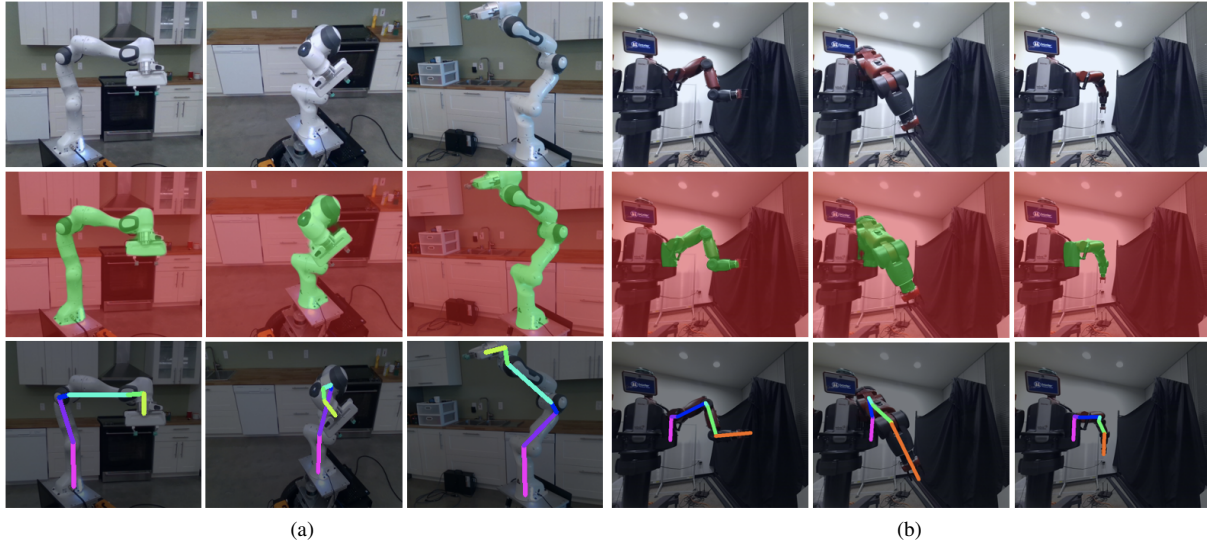
Figure 4. Qualitative results of CtRNet foreground segmentation and pose estimation on (a) DREAM-real dataset and (b) Baxter dataset. The first row shows the input RGB image, the second row shows the foreground segmentation, and the third row shows the projected robot skeleton based on the estimated robot pose.

trained foreground segmentation. The 2D PCK results and 3D ADD results are reported in Tab. 2. Our method outperforms all other methods on both 2D and 3D evaluations. For 2D evaluation, we achieve 93.94 AUC for PCK with an average reprojection error of 11.62 pixels. For 3D evaluation, we achieve 83.93 AUC for ADD with an average ADD of 63.81mm. Notably, 99 percent of our estimation has less than 50 pixel reprojection error, which is less than 2 percent of the image resolution, and 88 percent of our estimation has less than 100mm distance error when localizing the end-effector.

4.4. Ablation Study

We study how the number of pretraining samples affects the convergence and performance of the self-supervised training empirically on the Baxter dataset. We pretrain the neural network with different numbers of synthetic data samples $N_{pretrain} = \{500, 1000, 2000, 4000, 8000\}$, and examine the convergence of the self-supervised training process. Fig. 5 shows the plot of self-training loss ($\mathcal{L}_{mask} + \mathcal{L}_{seg}$) vs. the number of epochs for networks pretrianed with different number of synthetic data. We observe that doubling the size of pretraining dataset significantly improves the convergence of the self-training process at the beginning. However, the improvement gets smaller as the pretrainig size increase. For the Baxter dataset, the improvement saturates after having more than 2000 pretraining samples. Continuing double the training size results in very marginal improvement. Noted that the Baxter dataset captures 20 different robot poses from a fixed camera position. The required number of pretraining samples might
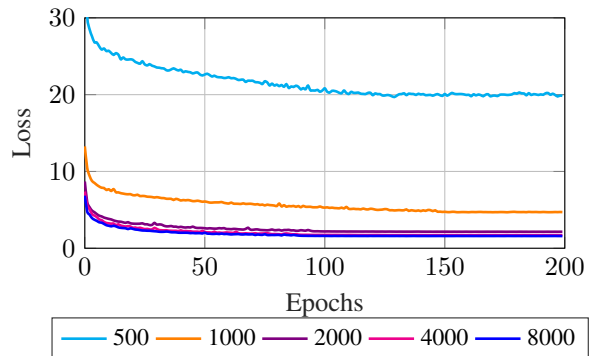

Figure 5. The training loss vs. number of epochs for the self-supervised training with different numbers of pretraining samples. More pretraining samples results in better convergence. The improvement saturates after having more than 2000 pretraining samples as only marginal improvement by adding more samples.

vary according to the complexity of the environment.

We further evaluate the resulting neural networks with the ground-truth labels on the Baxter dataset. We report the mean ADD and AUC ADD for the pose estimation in Tab. 3. The result verifies our observation on the convergence analysis. Having more pretraining samples improves the performance of pose estimation at the beginning, but the improvement stagnates after having more than 2000 pretraining samples.

4.5. Visual Servoing Experiment

We integrate the proposed CtRNet into a robotic system for position-based visual servoing (PBVS) with eye-to-hand

$N_{pretrain}$	Mean ADD (mm) ↓	AUC ADD ↑
500	2167.30	47.62
1000	92.91	76.65
2000	67.51	82.98
4000	63.00	84.12
8000	63.81	83.93

Table 3. Ablation study for the number of pretraining samples.

Method	Loop Rate	Trans. Err. (m)	Rot. Err. (rad)
DREAM [29]	30Hz	0.235 ± 0.313	0.300 ± 0.544
Diff. Rendering	1Hz	0.046 ± 0.062	0.036 ± 0.066
CtRNet	30Hz	$\mathbf{0.002 \pm 0.001}$	$\mathbf{0.002 \pm 0.001}$

Table 4. Mean and standard deviation of the translational error and rotational error for the visual servoing experiment.

configuration. We conduct the experiment on a Baxter robot and the details of the PBVS are described in the Supplementary Materials. The PBVS is purely based on RGB images from a single camera and the goal is to control the robot end-effector reaching a target pose defined in the camera frame. Specifically, we first set a target pose with respect to the camera frame. The target pose is then transformed into the robot base frame through the estimated camera-to-robot transformation. The robot controller calculates the desired robot configuration with inverse kinematics and a control law is applied to move the robot end-effector toward the target pose.

For comparison, we also implemented DREAM [29] and a Differentiable Renderer for PBVS. For DREAM, the pre-trained model for Baxter is applied. For Differentiable Renderer, we use the foreground segmentation of CtRNet to generate a robot mask. The optimizing loop for the renderer takes the last estimation as initialization and performs 10 updates at each callback to ensure convergence and maintain 1Hz loop rate. In the experiment, we randomly set the target pose and the position of the camera, and the robotic system applies PBVS to reach the target pose from an arbitrary initialization, as shown in the Fig. 6. We ran the experiment for 10 trails with different robot pose estimation methods, and the translational (Euclidean distance) and rotational errors (Euler angles) of the end-effector are reported in Tab. 4. The experimental results show that our proposed method significantly improves the stability and accuracy of the PBVS, achieving 0.002m averaged translational error and 0.002rad rotational error on the end-effector.

We also plot the end-effector distance-to-goal over time for a selected trail in Fig. 7. In this selected trial, the system could not converge with DREAM because the poor robot pose estimation confuses the controller by giving the wrong target pose in the robot base frame, which is unreachable.

Figure 6. Snapshots of PBVS. The goal is to move the end-effector to the target pose (green). The figure on the right shows the robot configuration upon the convergence of PBVS.
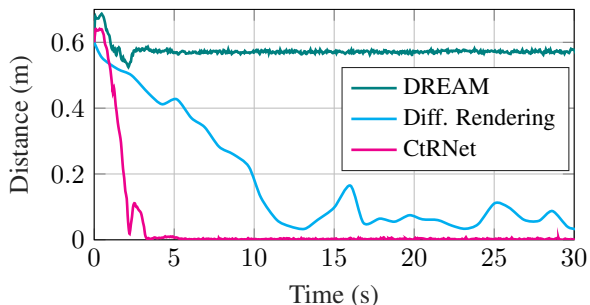

Figure 7. The plot of end-effector distance-to-goal over time on a selected PBVS trail.

With the differentiable renderer, the servoing system takes more than 10 seconds to converge and oscillate due to the low loop rate. With our proposed CtRNet, the servoing system converges much faster (≤ 5 seconds), thanks to the fast and robust robot pose estimation. We show more qualitative results in the Supplementary Materials.

5. Conclusion

We present the CtRNet, an end-to-end image-based robot pose estimation framework, and a self-supervised training pipeline that utilizes unlabelled real-world data for sim-to-real transfer. The CtRNet, using a keypoint detector for pose estimation while employing a rendering method for training, achieves state-of-the-art performance on robot pose estimation while maintaining high-speed inference. The Fig. 1 illustrates the advantages of CtRNet over existing methods, where the AUC values are normalized across two evaluation datasets by taking DREAM and CtRNet as references. We further experiment with different robot pose estimation methods by applying them to PBVS, which demonstrates CtRNet's fast and accurate robot pose estimation enabling stability when using single-frame robot pose estimation for feedback. Therefore, CtRNet supports real-time markerless camera-to-robot pose estimation which has been utilized for surgical robotic manipulation [43] and mobile robot manipulators [53]. For future work, we would like to extend our method to more robots and explore vision-based control in an unstructured environment.

References

[1] Jeannette Bohg, Javier Romero, Alexander Herzog, and Stefan Schaal. Robot arm pose estimation through pixel-wise part classification. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3143–3150. IEEE, 2014. 2

[2] Joao Borrego, Rui Figueiredo, Atabak Dehban, Plinio Moreno, Alexandre Bernardino, and José Santos-Victor. A generic visual perception domain randomisation framework for gazebo. In *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 237–242. IEEE, 2018. 3

[3] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018. 3

[4] François Chaumette, Seth Hutchinson, and Peter Corke. Visual servoing. In *Springer Handbook of Robotics*, pages 841–866. Springer, 2016. 1

[5] Bo Chen, Alvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8100–8109, 2020. 5

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 5

[7] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3339–3348, 2018. 3

[8] Jacques Denavit and Richard S Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. 1955. 3

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[10] Karthik Desingh, Shiyang Lu, Anthony Opipari, and Odest Chadwicke Jenkins. Factored pose estimation of articulated objects using efficient nonparametric belief propagation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7221–7227. IEEE, 2019. 2

[11] Irene Fassi and Giovanni Legnani. Hand to sensor calibration: A geometrical interpretation of the matrix equation ax= xb. *Journal of Robotic Systems*, 22(9):497–506, 2005. 2

[12] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016. 5

[13] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003. 1

[14] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. 1, 2, 6

[15] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017. 3

[16] Ran Hao, Orhan Özgüner, and M Cenk Çavuşoğlu. Vision-based surgical tool pose estimation for the da vinci® robotic surgical system. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1298–1305. IEEE, 2018. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[18] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 3

[19] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018. 3

[20] Radu Horaud and Fadi Dornaika. Hand-eye calibration. *The international journal of robotics research*, 14(3):195–210, 1995. 2

[21] Dániel Horváth, Gábor Erdős, Zoltán Istenes, Tomáš Horváth, and Sándor Földi. Object detection using sim2real domain randomization for robotic applications. *IEEE Transactions on Robotics*, 2022. 3

[22] Jarmo Ilonen and Ville Kyrki. Robust robot-camera calibration. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 67–74. IEEE, 2011. 2

[23] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019. 3

[24] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018. 2

[25] Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002. 5

[26] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Single-view robot pose and joint angle estimation via render & compare. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1654–1663, 2021. 2, 3, 4, 6

[27] Jens Lambrecht, Philipp Grosenick, and Marvin Meusel. Optimizing keypoint-based single-shot camera-to-robot pose estimation through shape segmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13843–13849. IEEE, 2021. 2

[28] Jens Lambrecht and Linh Kästner. Towards the usage of synthetic data for marker-less pose estimation of articulated robots in rgb images. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 240–247. IEEE, 2019. 2, 3

[29] 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. 2, 3, 5, 6, 8

[30] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009. 1, 5

[31] Yang Li, Florian Richter, Jingpei Lu, Emily K Funk, Ryan K Orosco, Jianke Zhu, and Michael C Yip. Super: A surgical perception framework for endoscopic tissue manipulation with surgical robotics. *IEEE Robotics and Automation Letters*, 5(2):2294–2301, 2020. 1

[32] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018. 2

[33] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019. 2, 3

[34] Jingpei Lu, Florian Richter, and Michael C Yip. Pose estimation for robot manipulators via keypoint optimization and sim-to-real transfer. *IEEE Robotics and Automation Letters*, 7(2):4622–4629, 2022. 2, 3, 6

[35] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019. 1

[36] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 800–815, 2018. 3

[37] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Pose estimation of kinematic chain instances via object coordinate regression. In *BMVC*, pages 181–1, 2015. 2

[38] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011. 1, 2

[39] Frank C Park and Bryan J Martin. Robot sensor calibration: solving AX= XB on the Euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, 1994. 2

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[41] Karl Pauwels, Leonardo Rubio, and Eduardo Ros. Real-time model-based articulated object pose detection and tracking with variable rigidity constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4001, 2014. 2

[42] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 3

[43] Florian Richter, Jingpei Lu, Ryan K Orosco, and Michael C Yip. Robotic tool tracking under partially visible kinematic chain: A unified approach. *IEEE Transactions on Robotics*, 2021. 2, 8

[44] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3752–3761, 2018. 3

[45] Tanner Schmidt, Richard A Newcombe, and Dieter Fox. Dart: Dense articulated real-time tracking. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA, 2014. 2

[46] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the european conference on computer vision (ECCV)*, pages 699–715, 2018. 3

[47] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018. 3

[48] 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. 2

[49] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977, 2018. 3

[50] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018. 3

[51] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018. 3

[52] Felix Widmaier, Daniel Kappler, Stefan Schaal, and Jeannette Bohg. Robot arm pose estimation by pixel-wise regression of joint angles. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 616–623. IEEE, 2016. 2

[53] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots*, 2016. 8

[54] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 5

[55] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020. 2

[56] Yiming Zuo, Weichao Qiu, Lingxi Xie, Fangwei Zhong, Yizhou Wang, and Alan L Yuille. Craves: Controlling robotic arm with a vision-based economic system. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4214–4223, 2019. 2, 3