UNIVERSITY OF CALIFORNIA SAN DIEGO

Image-based Robot Pose Estimation

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Electrical Engineering (Intelligent Systems, Robotics and Control)

by

Jingpei Lu

2024

DEDICATION

This dissertation is dedicated to my family: my father, Kekang Lu, who taught me to always be humble, diligent, and resilient, which are the keys to navigating my academic journey; my mother, Xiurong Wang, for her unconditional love and support in everything I do; my dog, Maymay, who always cheers me up when I come home; and my wife, Xin Lin, who has walked every step of this journey with me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Prof. Michael Yip, for his invaluable support throughout my PhD journey. His insightful ideas and guidance shaped the direction of my research, and his encouragement for collaboration enriched my work and opened new avenues of exploration. I am also deeply thankful for the many connections he facilitated with brilliant people, which had a profound impact on my research and future career. His mentorship has been instrumental to my growth, both academically and professionally.

I would also like to thank my committee members Prof. Nikolay Atanasov, Prof. Henrik Christensen, and Prof. Nuno Vasconcelos for their support, feedback, and the time they have dedicated to my dissertation.

Additionally, I would like to thank Dr. Yang Li and Dr. Florian Richter for introducing me to the ARClab and for mentoring me on my very first research project. Their passion inspired me to pursue research, and their guidance has been crucial in my academic journey.

Finally, I want to express my gratitude to my collaborators and the members of ARClab: Dr. Shan Lin, Dr. Fei Liu, Dr. Jacob Johnson, Dr. Dimitri Schreiber, Dr. Ahmed Qureshi, Dr. Ryan Orosco, Dr. Jonathan Katz, Dr. Sainan Liu, Dr. Cédric Girerd, Ambareesh Jayakumari, Zihan Li, Yunhai Han, Entong Su, Mingen Li, Chong He, Albert Miao, Albert Liao, Shunkai Yu, Shreya Saha, Zekai Liang, Kaiyuan Wang, Tristin Xie, Aman Gupta, Jason Lim, Zih-Yun Chiu, Nikhil Shinde, Yuheng Zhi, Elizabeth Peiros, Xiao Liang, Soofiyan Atar, Neelay Joglekar, Sharath Matada, Zhaowei Liu, Linjun Li, Hanpeng Jiang, Soumyaraj Bose, Yutong Zhang. Their friendship, collaboration, and insightful discussions have enriched my time in the lab. I am thankful for the opportunity to work alongside such a talented group of individuals.

Chapter 1, in part, is a reprint of the material from J. Lu, F. Richter, M. C. Yip, "Markerless Camera-to-Robot Pose Estimation via Self-supervised Sim-to-Real Transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*,

2023. The dissertation author was the primary author of this paper.

Chapter 2, in part, is a reprint of the material from J. Lu, F. Richter and M. C. Yip, “Pose Estimation for Robot Manipulators via Keypoint Optimization and Sim-to-Real Transfer,” in *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4622-4629, April 2022. The dissertation author was the primary author of this paper.

Chapter 3, in part, is a reprint of the material from J. Lu, F. Liu, C. Girerd and M. C. Yip, “Image-based Pose Estimation and Shape Reconstruction for Robot Manipulators and Soft, Continuum Robots via Differentiable Rendering,” in *IEEE Conference on Robotics and Automation*, 2023. The dissertation author was one of the primary authors of this paper. Chapter 3, in part, is also a reprint of the material from J. Lu, F. Richter, S. Lin and M. C. Yip, “Tracking Snake-like Robots in the Wild using only a Single Camera,” in *IEEE Conference on Robotics and Automation*, 2024. The dissertation author was the primary author of this paper.

Chapter 4, in part, is a reprint of the material from J. Lu, F. Richter, M. C. Yip, “Markerless Camera-to-Robot Pose Estimation via Self-supervised Sim-to-Real Transfer,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. The dissertation author was the primary author of this paper.

Chapter 5, in part, has been submitted for publication of the material as it may appear in J. Lu, Z. Liang, T. Xie, F. Ritcher, S. Lin, S. Liu, M. C. Yip, “CtRNet-X: Camera-to-Robot Pose Estimation in Real-world Conditions Using a Single Camera” in *IEEE Conference on Robotics and Automation*, 2025. The dissertation author was one of the primary authors of this paper.

VITA

2018	Bachelor of Science in Electrical Engineering, University of California San Diego
2020	Master of Science in Electrical Engineering, University of California San Diego
2024	Doctor of Philosophy in Electrical Engineering, University of California San Diego

PUBLICATIONS

J. Lu, F. Richter, S. Lin and M. C. Yip, "Tracking Snake-like Robots in the Wild using only a Single Camera" in *IEEE Conference on Robotics and Automation* (ICRA), 2024.

S. Liu, S. Lin, **J. Lu**, A. Supikov, M. C. Yip, "BAA-NGP: Bundle-Adjusting Accelerated Neural Graphics Primitives" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR) Workshops, 2024.

A. J. Miao, S. Lin, **J. Lu**, F. Richter, B. Ostrander, E. K. Funk, R. K. Orosco, M. C. Yip, "HemoSet: The First Blood Segmentation Dataset for Automation of Hemostasis Management" in *IEEE International Symposium on Medical Robotics* (ISMR), 2024.

S. Lin, A. J. Miao, A. Alabiad, F. Liu, K. Wang, **J. Lu**, F. Richter, M. C. Yip"SuPerPM: A Large Deformation-Robust Surgical Perception Framework Based on Deep Point Matching Learned from Physical Constrained Simulation Data" in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), 2024.

J. E. Katz, J. Finnegan, **J. Lu**, S. Lin, M. C. Yip, R. Sur "3D Rendering of Cystoscopy Video Footage: A Novel Method Utilizing Neural Radiance Field Processing" in *Journal of Urology*, 211(5S), p.e552., 2024.

J. Lu, F. Richter, M. C. Yip, "Markerless Camera-to-Robot Pose Estimation via Self-supervised Sim-to-Real Transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 2023.

J. Lu, F. Liu, C. Girerd and M. C. Yip, "Image-based Pose Estimation and Shape Reconstruction for Robot Manipulators and Soft, Continuum Robots via Differentiable Rendering," in *IEEE Conference on Robotics and Automation* (ICRA), 2023.

S. Lin, A.J. Miao, **J. Lu**, S. Yu, Z.Y. Chiu, F. Richter, and M.C. Yip, "Semantic-SuPer: A Semantic-aware Surgical Perception Framework for Endoscopic Tissue Classification, Reconstruction, and Tracking," in *IEEE Conference on Robotics and Automation* (ICRA), 2023.

F. Liu, E. Su, **J. Lu**, M. Li, M. C. Yip "Differentiable Robotic Manipulation of Deformable Rope-like Object using Compliant Position-based Dynamics," in *IEEE Robotics and Automation Letters* (RA-L), vol. 8, no. 7, pp. 3964-3971, July 2023.

J. Lu, F. Richter and M. C. Yip, "Pose Estimation for Robot Manipulators via Keypoint Optimization and Sim-to-Real Transfer." in *IEEE Robotics and Automation Letters* (RA-L), vol. 7, no. 2, pp. 4622-4629, April 2022.

F. Richter, **J. Lu**, R. K. Orosco, M.C. Yip, "Robotic Tool Tracking under Partially Visible Kinematic Chain: A Unified Approach," *IEEE Transactions on Robotics* (T-RO), vol. 38, no. 3, pp. 1653-1670, June 2022.

F. Liu, Z. Li, Y. Han, **J. Lu**, F. Richter, M. C. Yip, "Real-to-Sim Registration of Deformable Soft Tissue with Position-Based Dynamics for Surgical Robot Autonomy," in *IEEE Conference on Robotics and Automation* (ICRA), 2021.

J. Lu, A. Jayakumari, F. Richter, Y. Li and M. C. Yip, "SuPer Deep: A Surgical Perception Framework for Robotic Tissue Manipulation using Deep Learning for Feature Extraction," in *IEEE Conference on Robotics and Automation* (ICRA), 2021.

Y. Li, F. Richter, **J. Lu**, E. K. Funk, R. K. Orosco, J. Zhu and M. C. Yip, "SuPer: A Surgical Perception Framework for Endoscopic Tissue Manipulation with Surgical Robotics," in *IEEE Robotics and Automation Letters* (RA-L), vol. 5, no. 2, pp. 2294-2301, April 2020.

ABSTRACT OF THE DISSERTATION

Image-based Robot Pose Estimation

by

Jingpei Lu

Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics and Control)

University of California San Diego, 2024

Professor Michael C. Yip, Chair

Modern robotic automation often relies on cameras for rich sensory input to infer tasks and provide feedback for closed-loop control. Accurate robot pose estimation is critical for linking visual feedback to the robot's operational space. Traditional camera-to-robot calibration methods are labor-intensive, typically requiring externally attached fiducial markers, collecting images of several robot configurations, and solving for transformations—limitations that hinder their use in dynamic or unstructured environments. This dissertation presents deep learning-based approaches for markerless robot pose estimation, aimed at eliminating cumbersome physical setups while enhancing calibration flexibility and accuracy. First, two methods are introduced: a keypoint-based approach and a rendering-based approach. The keypoint-based method employs a deep neural network for detecting robot keypoints, followed by a Perspective-n-Points solver to estimate the robot pose. In contrast, the rendering-based method uses binary robot masks as input, iteratively updating pose estimates through a differentiable rendering process that minimizes differences between rendered and observed data. Both methods achieve state-of-the-art performance in image-based robot pose estimation, and their capability for online pose tracking is demonstrated on surgical robot and snake robot when integrated with probabilistic filtering techniques. The dissertation further examines the strengths and limitations of both approaches and proposes a self-supervised training framework to leverage their complementary advantages. Finally, this work extends the pose estimation framework to scenarios where the robot is only partially visible by integrating a vision-language foundation model to evaluate the visibility of the robot's components. Consequently, this method enhances robot pose estimation across a broader range of real-world manipulation scenarios.

Chapter 1

Introduction

Robot manipulators are widely used in manufacturing, packaging, and processing industries. Traditionally, these manipulators were programmed once and then operated continuously, primarily in tasks like automobile assembly and object pick-and-place applications. However, modern factories now incorporate robotics for tasks that require human collaboration, as well as for autonomously executing intelligent tasks in unstructured environments. The use of manipulators in autonomous operations for tasks such as bin picking, welding, painting, and inspection relies heavily on concepts from robotics, including visual servoing, perception, and motion planning [4]. Visual serving [44], which uses visual feedback to control the movement of robots, is a crucial technique that enables manipulators to perform complex tasks in dynamic and unpredictable environments.

There are two fundamental configurations for the robot end-effector and the camera [16]: 1) Eye-in-hand, where the camera is attached to the moving hand, allowing it to observe the relative position of the target. 2) Eye-to-hand, where the camera is fixed in the environment, monitoring the target and the motion of the hand. Similarly, visual servoing can be categorized into two main schemes based on the camera's mounting position: Image-based visual servoing (IBVS) and Position-based visual servoing (PBVS) [15, 51]. These two schemes are distinguished by how they define the error formulation. IBVS, proposed by Weiss and Sanderson [113], is a control law based on the error between current and desired features on the image plane, without requiring any estimation of the target's pose. The features used can include the coordinates of visual elements, lines, or moments of specific regions. However, IBVS encounters challenges when dealing with large rotations, a scenario often referred to as camera retreat [28, 14]. On the other hand, PBVS is a model-based technique that utilizes a single camera. In this approach, the position and orientation of the object of interest are estimated with respect to the camera, with the control objective defined in 3D. While image features are also extracted in this method, they are used to estimate 3D information, specifically the object and robot pose in Cartesian space, for enabling 3D servoing.

This dissertation focuses on camera-to-robot pose estimation (also known as hand-eye calibration), which is a fundamental component of position-based visual servoing in an eye-to-hand configuration.

1.1 Position-based Visual Servoing

Position-based visual serving (PBVS) typically uses images captured by a camera to determine the target pose, position and orientation, of the robot end-effector. For example, based on the location of the object to be grasped seen in the image, the PBVS generates the ideal grasp pose as the target pose and tries to move the robot toward it. Examples of robotic automation using the PBVS range from bin sorting [82] to tissue manipulation in surgery [73].

As illustrated in Figure 1.1, when a target pose is identified in the camera frame, the PBVS system converts it to the robot base frame, where the robot's geometry (such as kinematics) is well-defined, using the camera-to-robot transform. Next, the system calculates the desired joint configuration using the inverse kinematics of the robot manipulator. A controller is then utilized to guide the robot to this target joint configuration. The PBVS system continually uses vision feedback from the camera to update the robot's current pose. The error term is defined as the Cartesian pose difference between the target pose and the current pose. The PBVS system aims to minimize this error by adjusting the robot's configuration accordingly.

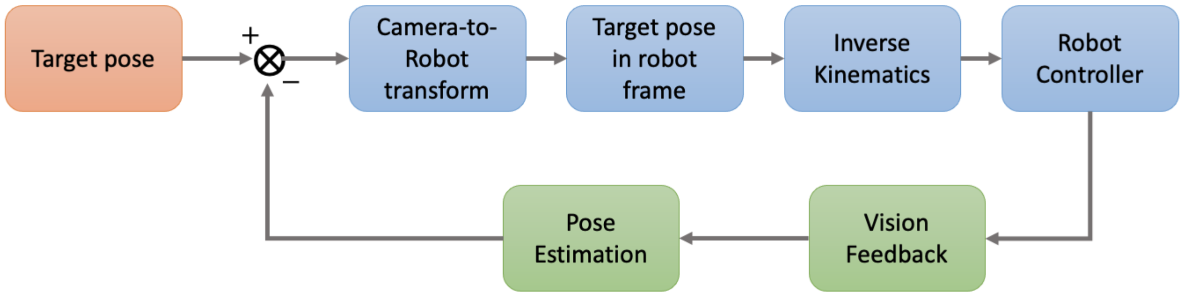

Figure 1.1. The overview of the PBVS system. The PBVS aims to minimize the error between the target robot pose and the current robot pose estimated from vision feedback.

The PBVS uses robot inverse kinematics to convert Cartesian control instructions into joint angle values for the robot, where the pose information is defined in the robot base frame. Hence, the accurate conversion of information from the camera frame to the robot frame, camera-to-robot transform, plays a crucial role in the process.

1.2 Camera-to-Robot Calibration

Camera-to-robot calibration, also known as hand-eye calibration, is a crucial process in robotic systems that determines the spatial relationship between a camera and a robot base or end-effector [124]. The core of camera-to-robot calibration is typically formulated as solving the $AX = XB$ matrix equation, where A and B are two systems, usually a robot base and a camera, and X is the unknown camera-to-robot transform [119]. This formulation has been the basis for numerous algorithms and variations developed over the years.

In practice, the camera-to-robot transform is calibrated with a calibration object, such as a checkerboard or fiducial markers [39, 90]. The calibration object has known geometry, allowing them to be detected from camera images, and their pose relative to the camera, denoted as T^{cam}_{obj}, can be estimated. Then, the calibration object is rigidly mounted onto the robot end-effector, where the end-effector pose with respect to the robot base frame, T^{rob}_{ee}, can be computed using robot forward kinematics. To derive the camera-to-robot transform, we close the pose loop using the following equation:

$$T^{cam}_{rob} = T^{cam}_{obj}(T^{rob}_{ee}T^{ee}_{obj})^{-1}. \tag{1.1}$$

In this context, the relative pose between the calibration object and robot end-effector, T^{ee}_{obj}, needs to be solved. Since T^{ee}_{obj} remains constant during the robot's motion, we can move a robot to a set of different configurations. For each one, T^{ee}_{obj} can be expressed as a function of the other two variables, T^{cam}_{obj} and T^{rob}_{ee}. With this set of equations, the T^{ee}_{obj} can be solved using the optimization techniques, such as the Least Squares method, to calculate the desired pose T^{cam}_{rob}.

1.3 Image-based Robot Pose Estimation

The camera-to-robot calibration procedure usually requires multiple runs with different robot configurations. 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 image-based markerless camera-to-robot calibration. Current approaches to robot pose estimation are mainly classified into two categories:

keypoint-based methods and rendering-based methods. The keypoint-based method, presented in Chapter 2, employs a deep neural network for detecting robot keypoints, followed by a Perspective-n-Points solver to estimate the robot pose. In contrast, the rendering-based method, presented in Chapter 3, uses binary robot masks as input, iteratively updating pose estimates through a differentiable rendering process that minimizes differences between rendered and observed data. Both methods achieve state-of-the-art performance in image-based robot pose estimation, and their capability for online pose tracking is demonstrated on surgical robot and snake robot when integrated with probabilistic filtering techniques. The dissertation further examines the strengths and limitations of both approaches in Chapter 4 and proposes a self-supervised training framework to leverage their complementary advantages. Finally, Chapter 5 extends the pose estimation framework to scenarios where the robot is only partially visible by integrating a vision-language foundation model to evaluate the visibility of the robot's components. Consequently, this method enhances robot pose estimation across a broader range of real-world manipulation scenarios.

1.4 Acknowledgements

Chapter 1, in part, is a reprint of the material from J. Lu, F. Richter, M. C. Yip, "Markerless Camera-to-Robot Pose Estimation via Self-supervised Sim-to-Real Transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. The dissertation author was the primary author of this paper.

Chapter 2

Keypoint-based Robot Pose Estimation

Visual feedback plays an integral role in robotics because of the rich information images provide. Historically, there have been two popular approaches to incorporating visual feedback. The first is through calibration and finding the transform between the base of the robot and the camera [34]. This transform describes the geometric relationship between objects in the camera frame and the robot, hence allowing image processing and detection algorithms to provide the context necessary for a robot to perform in an environment. The second approach is directly estimating the relationship between the control, typically joint angles, and the end-effector position in the camera frame [98]. This type of visual feedback also allows for end-effector control in the camera frame.

An important step to properly integrating visual feedback techniques is detecting features and finding their correspondence on the robot. A common approach is placing visual markers on the robot that are easy to detect and hence provide keypoints in the image frame [39][90]. But where should one place the markers? There does not seem to be any established approach that works the best. One must consider how these marker-based methods require modification of the robot, and how the visibility of the markers will frequently suffer from self-occlusions. Moreover, it is challenging to find the 3D location of the marker relative to the robotic kinematic chain which can cause inaccuracies in visual feedback. Deep learning approaches for detecting keypoints have been proposed to remove the need for modification of the robot [77]. The training of Deep Neural Networks (DNNs) for keypoint detection has even been extended to use synthetically generated data for accurate training with the correct corresponding 3D location relative to the kinematic chain [69].

In spite of the fact that these proposed methods have presented promising ways of keypoint detection, they have failed to address an important consideration which is where to place the keypoints relative to the kinematic chain. Previous work relies on hand-picked locations which may be sub-optimal and even limiting in performance for the detection algorithm. An example of this challenge can be found on symmetric robotic tools where the keypoint detection algorithm cannot solve the correspondence problem correctly [77].

Taking the advantages of the DNN as the learnable keypoint detector and robotic simulation tools, we present the following contributions:

1. a general keypoint optimization algorithm which solves for the locations of the set of keypoints to maximize their performance on localization tasks,

2. demonstrations showing that optimized keypoints can improve the performance on real robot pose estimation via sim-to-real transfer, and

3. methodology for incorporating optimized keypoints with a particle filter to achieving the state-of-the-art performance on surgical tool tracking.

We conducted live experiments on both a calibration and a tracking scenario to show the effectiveness of the proposed methods: (i) a Rethink Robotics Baxter robot [108] for calibrating the robot-to-camera transform and (ii) the da Vinci Research Kit (dVRK) [60] for real-time surgical tool tracking. The datasets for our calibration and tracking experiments are available online. The significant kinematic differences between these two robots show the generality of our approach. We show that the DNN based keypoint detector is capable of consistently and accurately detecting the 2D image projections of optimized keypoints even in cases of self-occlusion. The performance using optimized keypoints in both tasks outperforms previous methods with keypoints were selected manually by experienced roboticists, thereby highlighting the usefulness of keypoint optimization.

2.1 Related Works

Keypoint Optimization

Early works in robotics have explored how to select the optimal keypoints extracted from SIFT [76] and SURF [3], for visual odometry [89] and localization [125]. In computer vision, [10] and [88] select the salient keypoints by considering their detectability, distinctiveness, and repeatability. Recently, there has been a shift to using DNNs for detecting keypoints instead because of the improved performance. Works in the computer vision community have learned and optimized keypoints for better face recognition and human pose estimation [126][53][148]. However, those algorithms usually consume a large number of real images for training. Recently, [123] proposed an end-to-end framework to optimized the keypoints for object pose estimation in an unsupervised manner by utilizing the synthetically generated data. Our work differs from those in the particular goal that optimizing the 3D keypoints for 2D and 3D localization of robot manipulators. Instead of optimizing the keypoints for specific downstream tasks, our algorithm is more general and can be applied to various robotic tasks that use visual feedback, where the kinematic and the 3D geometric information of keypoints are required.

Robot Pose Estimation

A common approach for robot pose estimation is rigidly attaching markers to the robot (e.g. ArUco [39]), and directly estimate pose from visual data. The marker-based approach may fail in the case of motion blur and self-occlusion, hence, marker-less approaches are preferable in this scenario. [116] and [26] utilize depth images for tracking the pose of articulated objects. [77] and [106] employs DNN to extract point features from RGB images for surgical tool tracking. Recent works also address the data labeling issue and investigate using synthetic data to train DNNs for marker-less pose estimation [68, 69]. None of these previous approaches considered the specific problem of selecting the keypoint locations on robotic manipulators. We directly compare against previous manually selected keypoint locations for the Baxter robot [69] and the daVinci Surgical Robot Research Kit (dVRK) [77] in our experiments and achieve state-of-the-art performance with our optimized keypoints.

2.2 Methodology

We consider the problem of finding a set of keypoints $\mathcal{P}$ on the robot links that maximize the performance of localizing the robot in both 2D and 3D. The formulation of this problem is:

$$\mathcal{P}^* = argmin_{\mathcal{P}} \mathcal{L}_{2D}(\mathcal{P}, \Phi) + \lambda \mathcal{L}_{3D}(\mathcal{P}) \tag{2.1}$$

where $\mathcal{L}_{2D}$ is the error of 2D detection with the DNN parameterized by Φ, $\mathcal{L}_{3D}$ is the error of keypoints in the 3D space, and λ is a weighting factor. A set of keypoint is formally defined as $\mathcal{P} := \{\mathbf{p}_i | \mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{K}$, where $\mathbf{p}$ is the 3D position of the keypoint with respect to the robot link it belongs to, and K is the number of keypoints. To save the efforts of collecting and manually annotating the data, the optimization process is done on the synthetic dataset generated from a robot simulator. Finally, the keypoint detection in real images is achieved by an effective sim-to-real transfer technique.

2.2.1 Keypoint Optimization for Robotic Manipulators

Finding the keypoints that maximizing the localization performance will improve many robotic tasks that rely on keypoints as visual feedback. The DNN is trained end-to-end to detect a set of keypoints, and the detection performance of one certain keypoint can be varied while trained together with other keypoints. This has a significant impact on point-based pose estimation process (e.g. the family of Perspective-n-Point algorithms). Therefore, the keypoint optimization algorithm's main objective is to find the set of keypoints that optimize the both detection performance of the DNN and the accuracy of 3D pose estimation as shown in (2.1).

The proposed algorithm assumes N candidate keypoints have existed with known 3D positions along the kinematic chain. The location of the keypoints can be flexible, as they can randomly distributed to allow non-intuitive keypoint locations. The size of the keypoint set, K, can be varied but should be less than the number of candidate keypoints ($N > K$). Going through all possible sets of keypoints to solve (2.1) is intractable. Instead, we solve for the optimal set by sampling K keypoints, denoted as $\mathcal{P}$, and evaluate their performance on 2D and 3D localization like the loss in (2.1). This process is done iteratively where the evaluated performance of each sample set, $\mathcal{P}$, guides future sampling iterations such that the optimal set, $\mathcal{P}^*$, is found without needing to go through all possible sets of keypoints. The whole optimization process is shown in the Algorithm 1, and each iteration can be broke into four steps: sample keypoints, train model, evaluate performance, and update weights.

Sample Keypoints

Each candidate keypoint is associated with weight variable $w \in \mathbb{R}$, which can be interpreted as the confidence of being in the optimal set. The K keypoints are sampled based on their weights in each iteration. The function $\textbf{sampleKeypoints}(\mathbf{W}, K)$ takes in the weights of all the candidates $\mathbf{W} \in \mathbb{R}^{N\times 1}$ and randomly selects K keypoints among the candidates according to the probability:

$$P(\mathbf{p}_i) = \frac{w_i}{\sum\limits_{n\in N} w_n}. \tag{2.2}$$

Algorithm 1: Keypoint Optimization for Robot Manipulators

```
   Input: $N$ candidate keypoints
   Output: The optimal set of $K$ keypoints $\mathcal{P}^*$
   // Initialization
1  for $i = 1$ to $N$ do
2      $w_i^{(0)} = \frac{1}{N}$;
3  $\mathcal{E}_{min} = \infty$;
   // Optimize keypoints iteratively
4  for $t = 1$ to $T$ do
5      $\mathcal{P}^{(t)} \leftarrow \text{sampleKeypoints}(\mathbf{W}^{(t-1)}, K)$;
6      $\Phi^{(t)} \leftarrow \text{trainModel}(\mathcal{P}^{(t)})$;
7      $\mathcal{L}_{total}(\mathcal{P}^{(t)}) \leftarrow \text{evaluatePerformance}(\mathcal{P}^{(t)}, \Phi^{(t)})$;
8      $\mathcal{E}^{(t)} = \frac{1}{K} \sum_{\mathbf{p}_i \in \mathcal{P}^{(t)}} \mathcal{L}_{total}(\mathbf{p}_i)$;
9      if $\mathcal{E}^{(t)} < \mathcal{E}_{min}$ then
           // Update optimal keypoint set
10         $\mathcal{E}_{min} = \mathcal{E}^{(t)}$;
11         $\mathcal{P}^* = \mathcal{P}^{(t)}$;
12     $\mathbf{W}^{(t)} \leftarrow \text{updateWeights}(\mathcal{L}_{total}(\mathcal{P}^{(t)}), \mathbf{W}^{(t-1)})$;
13 return $\mathcal{P}^*$
```

The weights, w_i, are iteratively adjusted according to the keypoints 2D and 3D performance hence guiding the search to solve (2.1). More advanced sampling can be applied by introducing the constraints. For example, if we want to constraint the sampling so that one keypoint is selected per link, the candidates can be divided into sub-groups for each link.

Train Model

For training a detection model to detect the sampled keypoint set $\mathcal{P}$. We utilize the backbone neural network from DeepLabCut [84] as our detection model and the dataset is synthetically generated from a robot simulator (see Section 2.2.2 for data generation). We split the dataset into training set and testing set. The function **trainModel**($\mathcal{P}$) trains the

DNN on the training dataset to predict the image coordinates of the selected keypoints:

$$\Phi^* = argmin_{\Phi} \mathcal{L}_{train}(\mathcal{P}, \Phi) \tag{2.3}$$

where Φ denotes the parameters of the detection model and it is optimized using stochastic gradient descent with training loss $\mathcal{L}_{train}$ defined as the cross-entropy loss (see [84] for details). We generate the training and testing images with the ground-truth label for all N candidate keypoints beforehand. By doing so, the labels for selected keypoints in each iteration can be obtained without re-generating the datasets.

Evaluate Performance

After training, we evaluate the localization performance of the keypoints $\mathcal{P}$ with the detection model Φ using the testing data on the loss shown in (2.1). We define the 2D detection error, $\mathcal{L}_{2D}$, as the pixel-wise L_2 distance (Euclidean distance) between the detection and the ground-truth keypoint:

$$\mathcal{L}_{2D}(\mathbf{p}_i, \Phi) = \frac{1}{M} \sum_{m=1}^{M} ||f^i(I_m; \Phi) - \mathbf{h}_{i,m}||_2 \tag{2.4}$$

where $f^i(I_m; \Phi)$ and $\mathbf{h}_{i,m}$ are the DNN detected and ground-truth pixel location respectively of i-th keypoint for the testing image, I_m, and M is the number of images in the testing dataset. The 3D keypoint error, $\mathcal{L}_{3D}$, is defined as the average L_2 distance between the estimated and ground-truth 3D keypoint position in the camera frame $\{C\}$. For a keypoint with known position with respect to the j-th robot link $\mathbf{p}_i^j$, its position in camera frame can be obtained through forward kinematics and robot-to-camera transformation:

$$\overline{\mathbf{p}}_i^C = \mathbf{T}_B^C \prod_{n=1}^{j} \mathbf{T}_n^{n-1}(q_n) \overline{\mathbf{p}}_i^j \tag{2.5}$$

where $\mathbf{T}_n^{n-1}(q_n)$ is the n-th homogeneous joint transform with joint angle q_n. Note that coordinate frame 0 is the base frame of the robot and $\bar{\cdot}$ represents the homogeneous representation of a point (e.g. $\overline{\mathbf{p}} = [\mathbf{p}, 1]^T$). The robot-to-camera transformation, $\mathbf{T}_B^C$, is computed through the Efficient Perspective-n-Point Pose Estimation Algorithm (EPnP)[70] with the detected keypoints, $f^i(I_m; \Phi)$, and their corresponding position in the robot base frame. Then, the 3D keypoint error is calculated as:

$$\mathcal{L}_{3D}(\mathbf{p}_i) = \frac{1}{M} \sum_{m=1}^{M} ||\tilde{\mathbf{p}}_i^C - \mathbf{p}_i^C||_2 \tag{2.6}$$

where $\tilde{\mathbf{p}}_i^C$ is the estimated keypoint position from (2.5) using the estimated $\mathbf{T}_B^C$ from the current keypiont set and $\mathbf{p}_i^C$ is the ground-truth keypoint position in the camera frame.

Using the 2D and 3D losses in (2.4) and (2.6) respectively, a total loss similar to (2.1) per keypoint in $\mathcal{P}$ can be defined. The function **evaluatePerformance**$(\mathcal{P}, \Phi)$ does this by simply summing the two losses:

$$\mathcal{L}_{total}(\mathbf{p}_i) = \mathcal{L}_{2D}(\mathbf{p}_i, \Phi) + \lambda \mathcal{L}_{3D}(\mathbf{p}_i) \tag{2.7}$$

where $\mathcal{L}_{total}(\mathbf{p}_i)$ is the total loss for i-th keypoint. This loss is considered an evaluation of the keypoint $\mathbf{p}_i$ with respect to the optimization problem in (2.1) and used to guide future sampling to find the optimal set $\mathcal{P}^*$.

Update Weights

At the end of each iteration, the function **updateWeights**$(\mathcal{L}_{total}(\mathcal{P}))$ is used to update the weight of the selected keypoints based on (2.7). At iteration t of the optimization, the weight for the i-th keypoint in the sampled set $\mathcal{P}$ is updated to:

$$w_i^{(t)} = \left(\sum_{k \in K} w_k^{(t-1)} \right) \frac{e^{-\gamma \mathcal{L}_{total}(\mathbf{p}_i)}}{\sum_{k \in K} e^{-\gamma \mathcal{L}_{total}(\mathbf{p}_k)}} \tag{2.8}$$

where γ is a tuned parameter to control the effect of the error on changing the weights. The updated weight will increase or decrease according to the keypoints 2D and 3D performance hence guiding the sampling in **sampleKeyupoints**$(\mathbf{W}, K)$ towards the optimal set, $\mathcal{P}^*$ that solves (2.1).

2.2.2 Keypoint Detection via Domain Randomization

A simulated environment is used to generate the synthetic data for the proposed keypoint optimization method. By using simulated data, ground-truth labels can be directly generated. We set up the simulation environment using the robotic simulator CoppeliaSim [109] and interfaced using PyRep [54] to generate the ground truth label and render RGB images. Fig. 2.1 shows the simulator view and the rendered image for two robots from the virtual cameras.

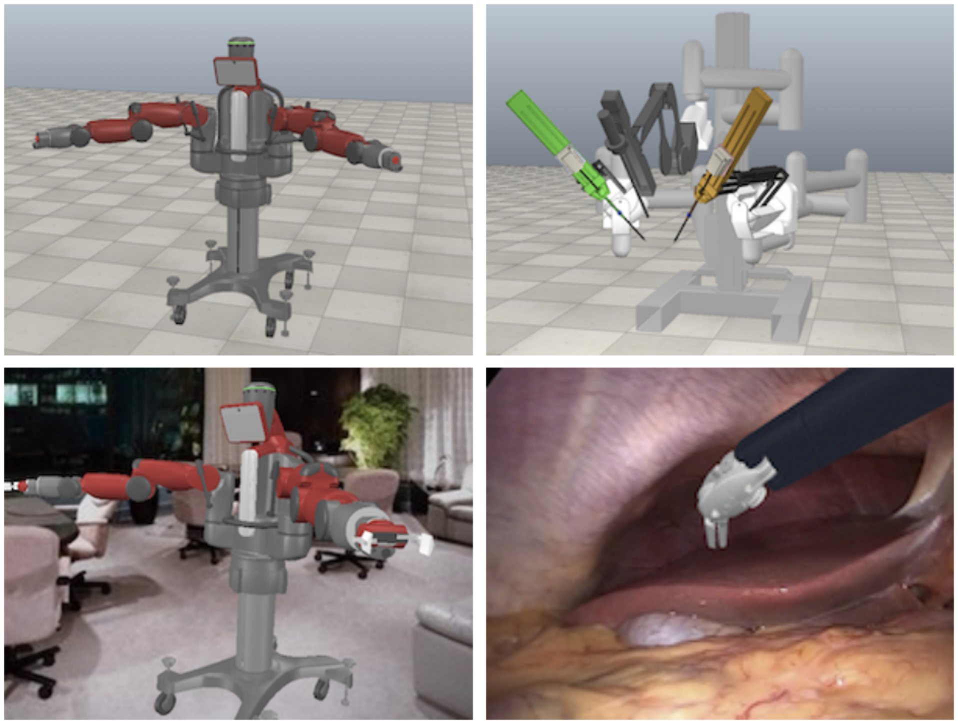

Figure 2.1. Simulation setup and rendered image of the Rethink Baxter (left) and the da Vinci Surgical System (right).

Although the simulator is crucial for ground truth labels and hence ideal for our keypoint optimization method, a naively trained DNN on the labels will be limited to only detecting keypoints on simulated images and may not generalize to the real world. To bridge the reality gap, the simple but effective technique known as *domain randomization* [128] is applied to transfer the keypoint detection DNN from the virtual domain to robots in the physical world. During the data generation, the virtual cameras are placed in the simulated scene that approximately matches the viewpoint of the real camera, and the following randomization settings are applied to generate the training samples:

- The angle for the robot joints is randomized within the joint limits.
- The pose of virtual cameras are randomized by adding a zero-mean Gaussian noise to the initial pose, such that

  $$[\mathbf{q}_{rand}, \mathbf{b}_{rand}]^\top \sim \mathcal{N}([\mathbf{q}_{init}, \mathbf{b}_{init}]^\top, \mathbf{\Sigma}) \tag{2.9}$$

  where $\mathbf{q} \in \mathbb{S}^3$ is the quaternion, $\mathbf{b} \in \mathbb{R}^3$ is the translational vector, and $\mathbf{\Sigma}$ is the covariance matrix.
- The number of the scene lights is randomly chosen between 1 to 3, and are positioned freely in the simulated scene with varying intensities.
- Distractor objects, like chairs and tables, are placed in the simulated environment with random poses.
- The background of the rendered images are randomly selected from Indoor Scene dataset [99] and Hamlyn Centre Endoscopic Video dataset [87].
- The color of the robot mesh is randomized by adding a zero-mean Gaussian noise with a small variance to the default RGB value.
- The rendered images are augmented by adding the additive white Gaussian noise using the image augmentation tool [56].

These randomization techniques were applied when generating synthetic data for both keypoint optimization and domain transfer hence ensuring the DNN optimized in (2.3) will generalize its keypoint detection to the real world.

2.3 Experiments and Results

In this section, we describe efforts towards evaluating the robustness of the keypoints optimization algorithm and the performance of using them in real robotics applications. Specifically, we compare our approach with the state-of-art algorithm and marker-based approach on a robot-to-camera pose estimation task and examine the differences between using the optimized keypoints and hand-picked keypoints on a robot tool tracking experiment. Moreover, we also study the impact of different neural network architectures on the keypoint optimization algorithm.

2.3.1 Datasets and Evaluation Metrics

Baxter dataset

This dataset contains 100 image frames (resolution: 2048×1526) of the Baxter robot with 20 different joint configurations collected using a Microsoft Azure Kinect. The ground-truth end-effector positions in the camera frame are provided, which is obtained by attaching an Aruco marker physically at the end-effector position. The performance of the optimized keypionts is evaluated in both 2D and 3D by estimating the end-effector position.

For 3D evaluation, we first estimate the robot-to-camera transformation using the optimized keypoints (rotation $\widetilde{\mathbf{R}}_B^C$ and translation $\widetilde{\mathbf{b}}_B^C$). Then, we transfer the end-effector from robot base frame to camera frame and calculate the L_2 distance between the ground-truth and estimated end-effector position:

$$\|(\widetilde{\mathbf{R}}_B^C \mathbf{x}_{ee}^B + \widetilde{\mathbf{b}}_B^C) - \mathbf{x}_{ee}^C\|_2 \tag{2.10}$$

where $\mathbf{x}_{ee}^{B}$ is the end-effector position in robot base frame obtained by forward kinematics of the robot, and $\mathbf{x}_{ee}^{C}$ is the ground-truth end-effector position in camera frame measured by Aruco marker.

For 2D evaluation, we propose the reprojection error (RE) for the end-effector. The RE is the L_2 distance between the estimated end-effector position and its ground-truth position in image coordinates,

$$RE = \left\| \frac{1}{z}\mathbf{K}(\widetilde{\mathbf{R}}_{B}^{C}\mathbf{x}_{ee}^{B} + \widetilde{\mathbf{b}}_{B}^{C}) - \overline{\mathbf{h}}_{ee} \right\|_{2} \tag{2.11}$$

where $\mathbf{h}_{ee}$ is the ground-truth end-effector position in image coordinates, $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the intrinsic matrix, and z is the z-value of the projected point. The percentage of correct keypoints (PCK), proposed in [141], is a metric for visualizing the keypoint detection performance across the dataset. The PCK measures the percentage of keypoints that the distance between the predicted and the ground-truth position is within a certain threshold. In this experiment, the PCK for 2D end-effector localization is calculated in pixels and the PCK for 3D end-effector localization is calculated in millimeters.

SuPer tool tracking dataset

The SuPer dataset[1] is a recording of a repeated tissue manipulation experiment using the da Vinci Research Kit (dVRK) surgical robotic system [60], where the stereo endoscopic video stream and the encoder readings of the surgical robot are provided. We extended the original surgical tool tracking dataset, which originally has 50 ground-truth surgical tool masks, to 80 ground-truth masks by hand labeling. The extended dataset covers a better variation of the tool poses, and the performance of the tool tracking is

[1]https://sites.google.com/ucsd.edu/super-framework/home

evaluated using the Intersection-Over-Union (Jaccard Index) for the rendered tool masks,

$$IoU = \frac{|\mathbf{G} \cap \mathbf{P}|}{|\mathbf{G} \cup \mathbf{P}|} \tag{2.12}$$

where **G** is the ground-truth tool mask area and **P** is the predicted tool mask area in the image plane.

2.3.2 Keypoints Optimization

To demonstrate the keypoint optimization algorithm, we randomly placed 32 candidate keypoints ($N = 32$) on the da Vinci surgical tool with a uniform distribution in 3D Euclidean Space and 22 candidate keypoints on the Baxter left arm manually. The Algorithm 1 is applied to optimize the set of 7 ($K = 7$) keypoints for robust detection. The keypoint optimization algorithm was running for 15 iterations ($T = 15$) with $\gamma = 1$ on the synthetic dataset, containing 2K samples for training and 500 samples for evaluations. In the training step, the DNN is trained on the training dataset for 100,000 iterations, with a learning rate of 0.2.

To examine the impact of different neural network architectures on the keypoint optimization algorithm, we trained the DNNs with four different feature extractors: ResNet_50, ResNet_101, MobileNet_v2_1.0, and MobileNet_v2_0.5 [114]. The last digit indicates the number of layers and the width of the network for ResNets and MobilNets respectively, which implies the number of parameters of the network. The average keypoint errors from the **evaluatePerformance** step in each iteration are shown in Fig. 2.2, demonstrating that our algorithm is agnostic to different DNN architectures. With different feature extractors, the algorithm converges at a similar rate to the same set of optimal keypoints. The number of iterations required to converge to an optimal keypoint set is around 10 hence showing that our method is substantially faster than brute-force searching through all possible keypoint sets.

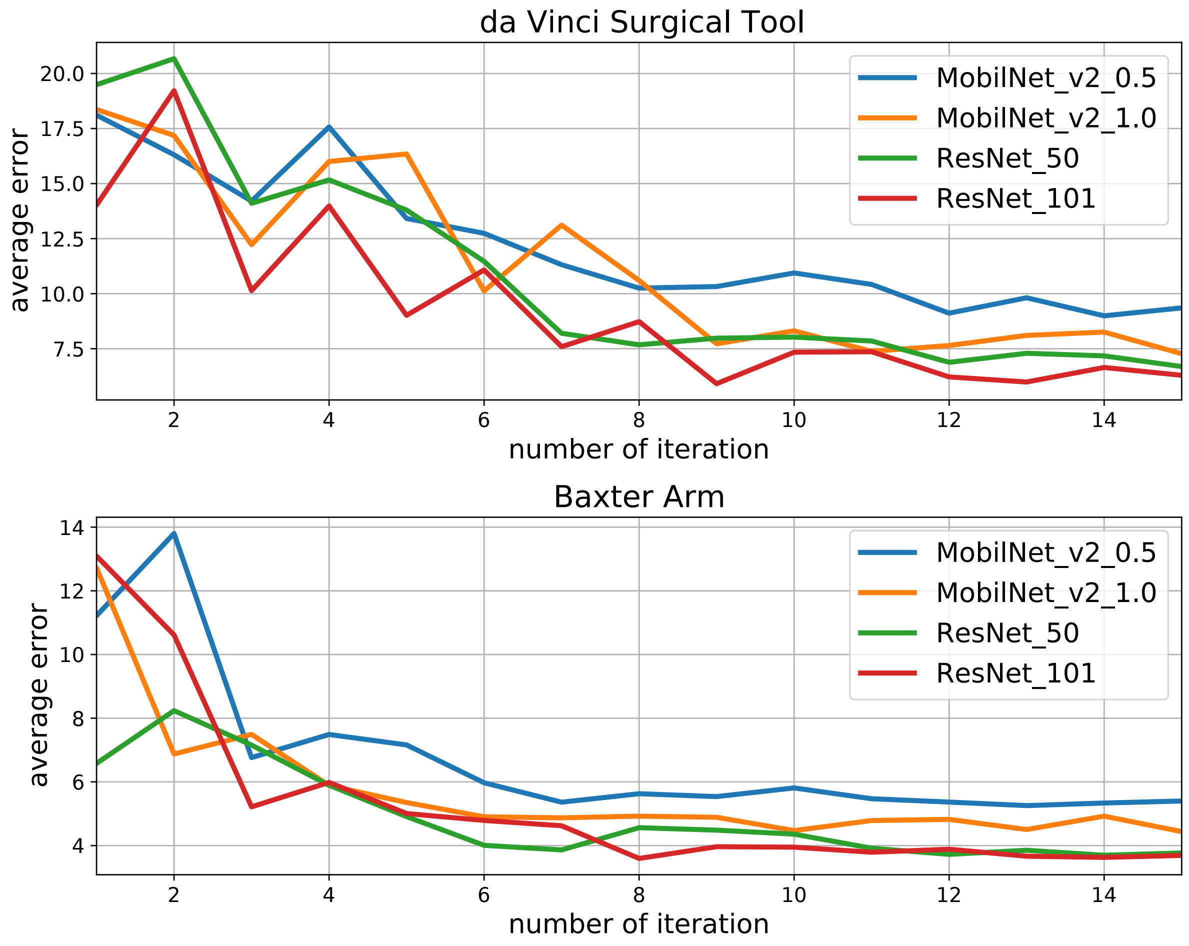

Figure 2.2. The plot of average error from the **evaluatePerformance** step for each iteration of keypoint optimization algorithm, while optimizing the keypoints for da Vinci surgical tool (top) and the Baxter arm (bottom). The algorithm behaves similarly with various DNN architectures.

We also investigated the choice of K (number of optimized keypoints) for pose estimation. We applied Algorithm 1 with various K and evaluated the pose estimation performance with the optimized keypoints on the Baxter dataset. We found that the estimation performance gets better in the beginning as the number of optimized keypoints increases. The resulting optimized keypoints create a better coverage of the robot since there are more keypoints. However, when there are too many (i.e. greater than 16 on the Baxter), the low-quality keypoint detections reduce the overall pose estimation performance.

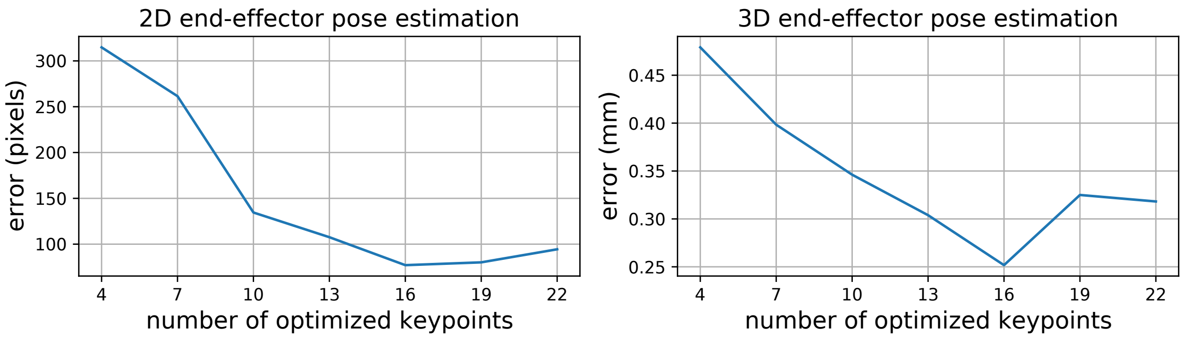

Figure 2.3. The performance of pose estimation using different numbers of optimized keypoints on Baxter dataset.

2.3.3 Robot-to-camera Pose Estimation from a Single Image

We explored robot-to-camera pose estimation performance from a single RGB image by utilizing the optimized keypoints on Baxter robot. The keypoints optimization is constrainted so that one keypoint is sampled for each robot link. The resulting 7 optimized keypoints are detected on the Baxter dataset to estimate the robot-to-camera transform, $\mathbf{T}_B^C$, on each frame in the same manner as the 3D loss, $\mathcal{L}_{3D}$, from the **evaluatePerformance**$(\mathcal{P}, \Phi)$ step of the proposed method. Note that the $(\widetilde{\mathbf{R}}_B^C, \widetilde{\mathbf{b}}_B^C)$ used to compute the errors in (2.11) and (2.10) is simply the rotation matrix and translation from $\mathbf{T}_B^C$.

We randomly placed three candidate keypoints for each link on the left arm, and the Algorithm 1 was applied to find one optimal keypoint per link, with $T = 20, \lambda = 50$, and $\gamma = 2$. The ResNet_50 was utilized as the feature extractor and was initialized with ImageNet-pretrained weights. The DNNs were trained for 500,000 iterations with a decayed learning rate. The optimized keypoints for the Baxter's left arm are shown in the middle-left of Fig. 2.5.

To transfer the keypoint detection to the real robot, we applied the domain randomization to bridge the reality gap, as described in Section 2.2.2. The experimental results show that the DNN generalizes well to real-world images. Fig. 2.4 shows the optimized keypoints' detections on the Baxter dataset, demonstrating that the keypoints

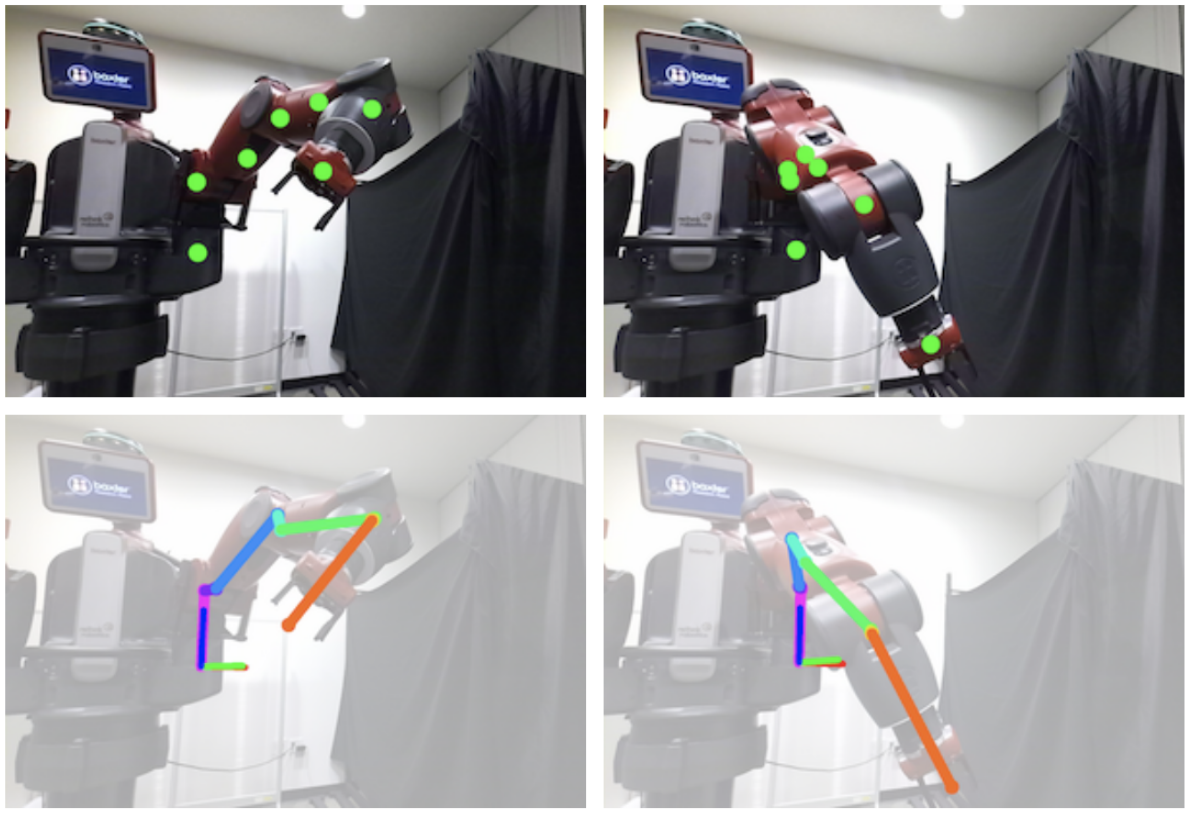
Figure 2.4. The examples of the optimized keypoints detection (top) and the skeletonization (bottom) of the Baxter robot in real-world images. The skeleton of the arms is based on the estimated base frame (RGB reference frame). Keypoints are estimated even with self-occlusions, and the skeletons are perfectly aligned with the robot arm.

can be detected even with self-occlusions. Given the accurate keypoint detections, the estimated skeletons are perfectly aligned with the robot arm.

For comparison, we also experimented the robot-to-camera pose estimation by using all the candidate keypoints or placing the keypoints at the exact joint locations. The state-of-the-art algorithm, DREAM [69], is also implemented for pose estimation which uses robot joints as the keypoints. The 2D keypoints are detected using the DNN from DREAM. We also compared the traditional camera-to-base pose estimation procedure by placing the Aruco markers on the robot arm. The 2D and 3D PCK results are shown in Fig. 2.6. Using the optimized keypoints, around 50 percent of the estimations have fewer than 25 pixels error in the image plane (about 1 percent of the image size) and have an error of less than 100mm in 3D space, which is much better than other methods. The area under the curve (AUC), indicating the mean of PCKs, also highlights the overall better performance of using the optimized keypoints. Due to the self-occlusions and the camera's pose, limiting the visibility of visual markers, some image frames do not have enough detected Aruco markers for EPnP (< 4), which hampers the performance.

2.4 Surgical Tool Tracking

Surgical tool tracking continuously estimates the 3D pose of the tool end-effector with respect to the camera frame. This is necessary for augmented reality displays [103], transferring of learning-based control policies [104], and other applications [144]. We employed the tool tracker previously developed in [77], which combines a keypoint detector and a particle filter for 3D pose estimation. The keypoint detector (e.g. the proposed optimized keypoint detection method) detects keypoints, $\mathbf{h}_{i,t}$, on the input image at time t to provide visual feedback. The corresponding 3D point $\mathbf{p}_i$ is projected to image frame as:

$$\overline{\mathbf{m}}_i(\boldsymbol{\omega}, \mathbf{b}) = \frac{1}{z}\mathbf{K}\mathbf{T}^{C}_{B-}\mathbf{T}^{B-}_{B}(\boldsymbol{\omega}, \mathbf{b}) \prod_{n=1}^{j} \mathbf{T}^{n-1}_{n}(q_n)\overline{\mathbf{p}}^{j}_{i} \tag{2.13}$$

where $\mathbf{m}_i$ is the re-projected keypoint location, $\mathbf{T}^{C}_{B-}$ is the initial hand-eye transform from calibration, and $\mathbf{T}^{B-}_{B}(\boldsymbol{\omega}, \mathbf{b}) \in SE(3)$ is the Lumped Error [106], parameterized by an axis-angle vector $\boldsymbol{\omega} \in \mathbb{R}^3$ and a translational vector $\mathbf{b} \in \mathbb{R}^3$. The Lumped Error compensates for both errors in joint angles and hand-eye in real-time to precisely track the tool in the camera frame. For more details, refer to our previous work [106]. Since the Lumped Error is not constant and the application is real-time tracking (i.e only detections up to time t is known), a tracking formulation with a Hidden Markov Model is proposed where the posterior probability conditioned on all observations is estimated recursively:

$$P(\boldsymbol{\omega}_t, \mathbf{b}_t | \mathbf{h}_{1:K,1:t}) \propto P(\mathbf{h}_{1:K,t} | \boldsymbol{\omega}_t, \mathbf{b}_t) \int P(\boldsymbol{\omega}_t, \mathbf{b}_t | \boldsymbol{\omega}_{t-1}, \mathbf{b}_{t-1}) P(\boldsymbol{\omega}_{t-1}, \mathbf{b}_{t-1} | \mathbf{h}_{1:K,1:t-1}) d[\boldsymbol{\omega}_{t-1}, \mathbf{b}_{t-1}]^T \quad (2.14)$$

which is solved using a particle filter [106]. The motion and observation models are respectively defined as:

$$[\boldsymbol{\omega}_t, \mathbf{b}_t]^T \sim \mathcal{N}([\boldsymbol{\omega}_{t-1}, \mathbf{b}_{t-1}]^T, \boldsymbol{\Sigma}_{\boldsymbol{\omega},\mathbf{b}}) \quad (2.15)$$

where $\boldsymbol{\Sigma}_{\boldsymbol{\omega},\mathbf{b}}$ is the covariance matrix and

$$P(\mathbf{h}_{1:K,t} | \boldsymbol{\omega}_t, \mathbf{b}_t) \propto \sum_{i=1}^{K} \rho_{i,t} e^{-\alpha ||\mathbf{h}_{i,t} - \mathbf{m}_i(\boldsymbol{\omega}_t, \mathbf{b}_t)||^2} \quad (2.16)$$

where $\rho_{i,t}$ is the confidence score from the keypoint detector, and α is a tuning parameter.

Differing from [73] and [77], we are using the optimized keypoints instead of the hand-picked keypoints. Then, domain randomization technique is used to bridge the reality gap between the synthetic and real-world images. The resulting keypoint detections are shown in Fig. 2.7. Note that the optimized keypoints are inside of the tool body, and the DNN can accurately predict their projections onto the image plane in different tool configurations.

For comparison, we evaluated the tool tracking performance with three different setups described as follows.

Hand-labelled SuPer Deep Keypoints: This setup is identical to the tool tracking approach in [77]. The seven hand-picked keypoints were used for tracking, which is on the surface of the tool. The DNN was then trained on the 100 real-world images with the hand-labeled ground-truth position.

Synthetically-labelled SuPer Deep Keypoints: The second setup used the same set of hand-picked keypoints from SuPer Deep, but the DNN was trained on the around 20K synthetic images with domain randomization, where the keypoint labels are provided even with occlusions.

Optimized Keypoints: The third setup was using the 7 optimized keypoints, as shown in Fig. 2.5, and the DNN was also trained on the synthetic images with domain randomization.

The tool tracking performance is computed by rendering a re-projected tool mask on the image frame from the estimation based on the keypoints, and the IoU is computed with the ground-truth mask for evaluation. Quantitative and qualitative results are shown in Fig. 2.8 and Fig. 2.9 respectively. The setup with *Hand-labelled SuPer Deep Keypoints* fails to track the tool when the tool was turning, as those hand-picked keypoints on the tool surface are occluded and humans fail to provide the label. However, using the optimized keypoints, the DNN makes accurate predictions for non-visible keypoints, as shown in Fig. 2.7, since the synthetically generated training data can provide labels for those scenarios. Although both *Synthetically-labelled SuPer Deep Keypoints* and *optimized keypoints* setups are utilizing the synthetic dataset, the *optimized keypoints* achieves higher accuracy because the keypoints are optimizing the detection performance of the DNN. Another advantage of the optimized keypoints is to reduce the ambiguity in detection. As stated in [77], the detection of some keypoints is challenging due to the tool's symmetry, which causes false detections. The optimized keypoints are instead distributed in an asymmetric pattern as shown in Fig. 2.5.

2.5 Discussion and Conclusion

We proposed a general keypoint optimization algorithm to maximize the performance of 2D and 3D localization on robotic manipulators. Our algorithm utilized a DNN for keypoint detections and can handle self-occlusions by optimizing the keypoint locations and training on synthetically generated data. The results show that the optimized keypoints yield higher accuracy compared to manually or randomly selected keypoints, hence resulting in better performance for the wide breadth of robotic applications that rely on keypoints for visual feedback. To show this, we presented both quantitative and qualitative results from two robotic applications: camera-to-base pose estimation and surgical tool tracking. The experimental results of detecting optimized keypoints in cases of self-occlusion further motivate the importance of this work as previously manually selected keypoints were unable to produce this type of result. For future work, we will incorporate the optimized keypoints for visual servoing and explore optimizing keypoints for other robot applications (e.g. motion planing [30]).

2.6 Acknowledgements

Chapter 2, in part, is a reprint of the material from J. Lu, F. Richter and M. C. Yip, "Pose Estimation for Robot Manipulators via Keypoint Optimization and Sim-to-Real Transfer." in *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4622-4629, April 2022. The dissertation author was the primary author of this paper.

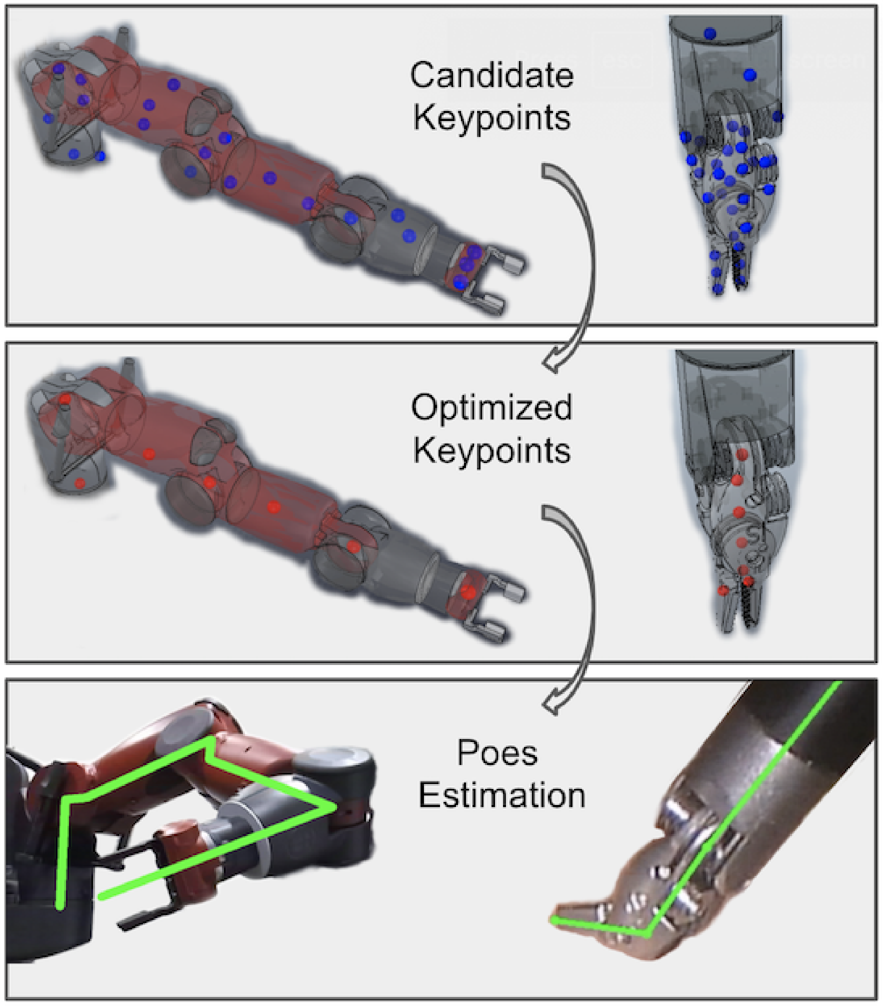

Figure 2.5. A visualization of the keypoints on the Baxter robot arm (left) and the da Vinci surgical robot tool (right). The candidate keypoints are shown in blue, and the optimized keypoints are shown in red. The bottom row shows these detections being used for pose estimation by utilizing the optimized keypoints.

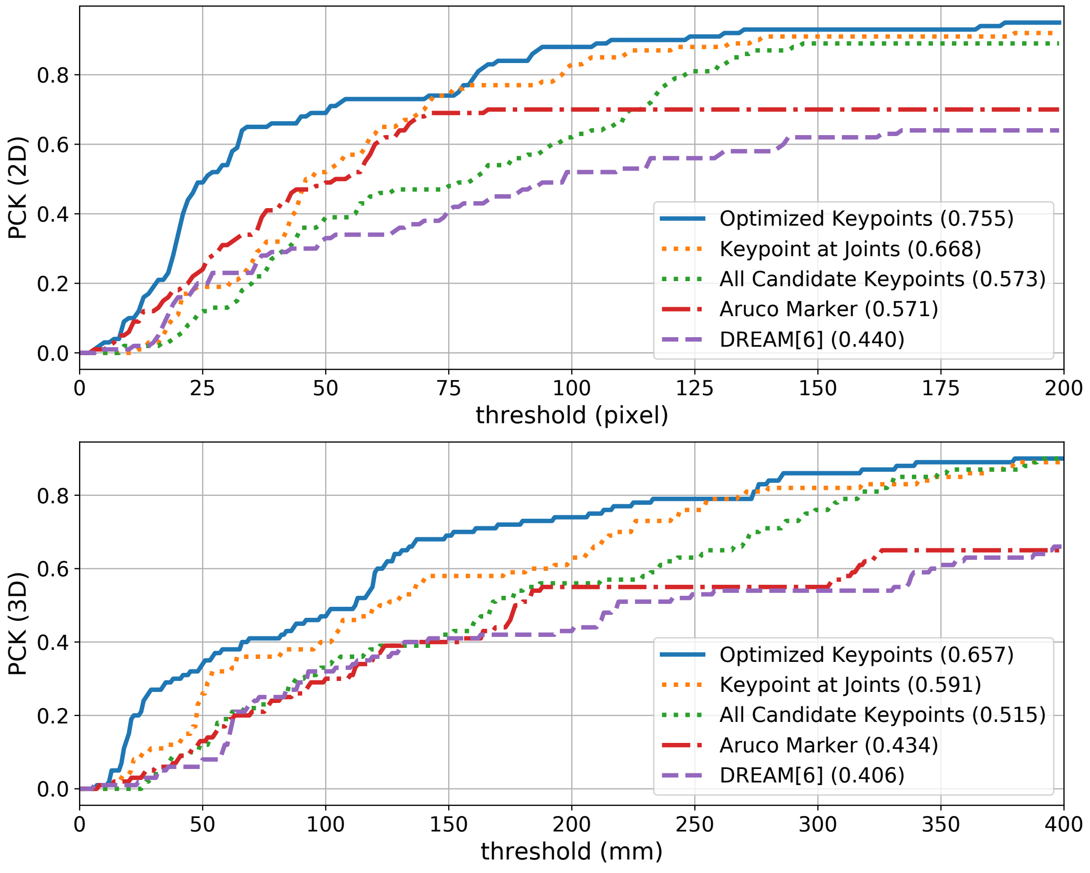

Figure 2.6. The PCK results on the Baxter dataset for end-effector localization in 2D (top) and 3D (bottom). The thresholds are the L_2 distances and the numbers in parentheses indicate the area under the curve (AUC). Around 50 percent of the estimations have an error of less than 1 percent of the image size (2048×1526) using the optimized keypoints.

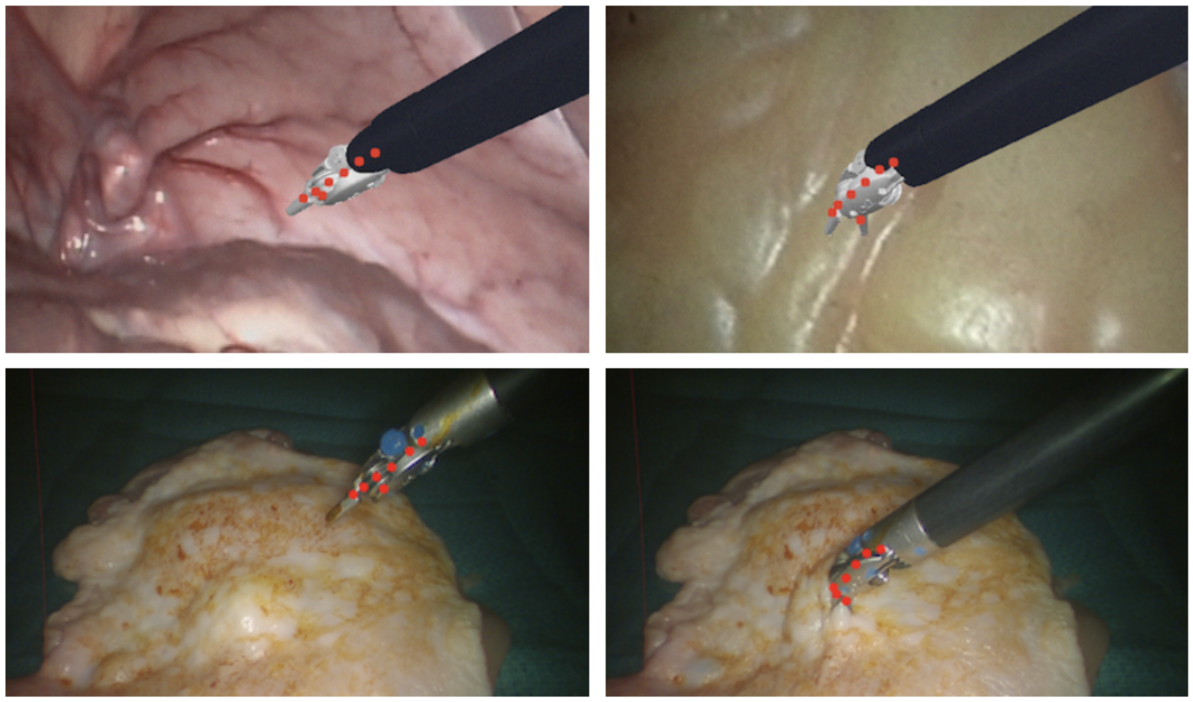

Figure 2.7. Detection of the optimized keypoints for the surgical tool on synthetic (top) and real (bottom) images. The 3D position of the optimized keypoints relative to the tool is shown in the middle-right of Fig. 2.5, and the DNN accurately predicts their projections on the image plane.

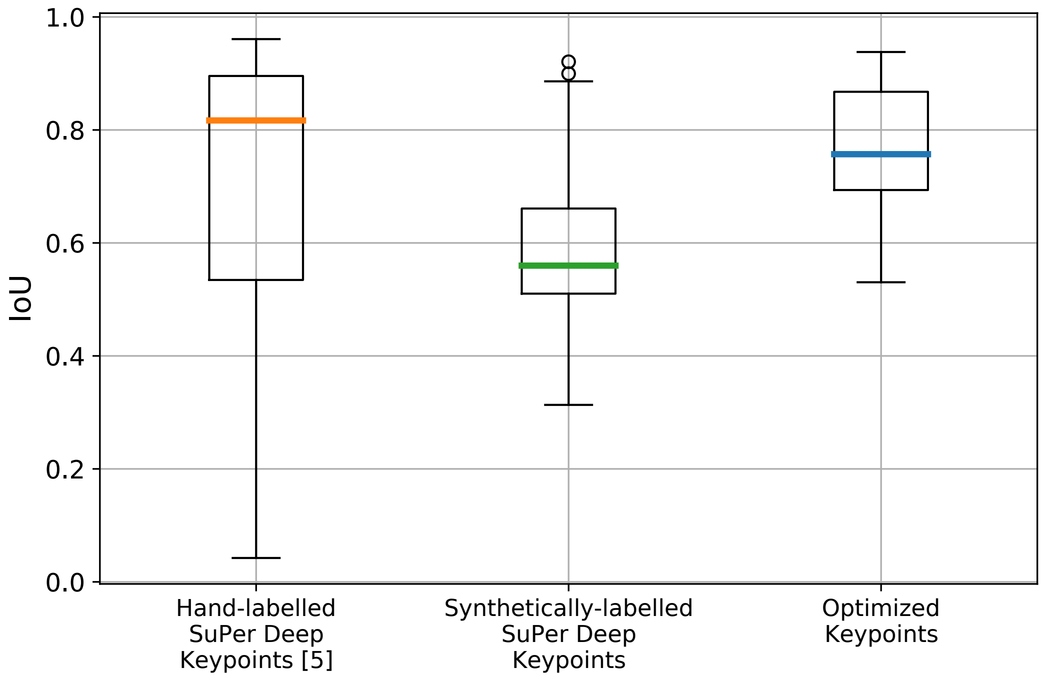

Figure 2.8. The box plot of the IoUs of the rendered tool mask using three different tool tracking setups (circles are outliers). The *Optimized Keypoints* has less variance with high accuracy.

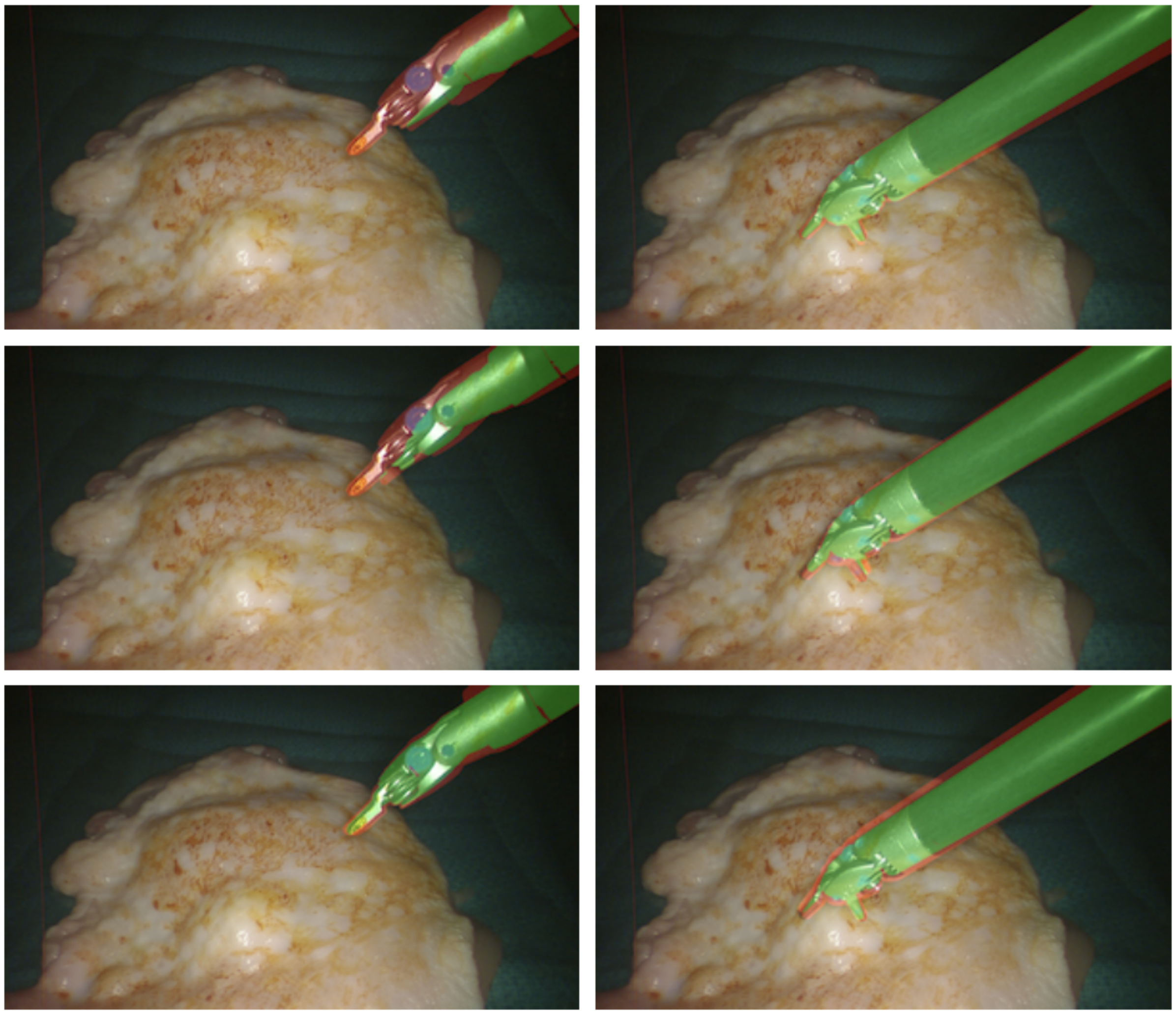

Figure 2.9. Qualitative results of the tool tracking for three different setups. From top to bottom, each row shows the results of *Hand-labelled SuPer Deep Keypoints*, the *Synthetically-labelled SuPer Deep Keypoints*, and the *Optimized Keypoints*. The green area shows the intersection of the rendered mask and ground-truth mask ($\mathbf{G} \cap \mathbf{P}$), and the red area shows the difference between the rendered mask and ground-truth mask ($\mathbf{G} \cup \mathbf{P} - \mathbf{G} \cap \mathbf{P}$).

Chapter 3

Rendering-based Robot Pose Estimation

3.1 Pose Estimation and Shape Reconstruction via Differentiable Rendering

Sensory feedback of state parameters, such as the position and body configuration of a robot in its environment, is a fundamental requirement for operating autonomous systems in real-world, unknown spaces. In-place sensing may exist with motor encoders for robot manipulators, or Fiber Braggs for soft robots [120], all providing an estimate of their relative pose and body configurations in relation to the real world environment; however, the limitation is they all exhibit cumulative position errors due to long kinematic chains. In addition, for both soft and rigid robots, the procedure for mounting internal sensors can be tricky and wiring and communication lines can constrain the mechanics and articulation of the robot. That is why measuring the body configuration of a soft robot is notoriously challenging.

If the goal is to observe and track the motion of robots in the wild, e.g., for behavior cloning or offline reinforcement learning, the robot's state information may not be readily available or even accessible. A good example is in minimally invasive surgery (MIS), where over 1 million procedures are performed yearly on a daVinci Surgical Robot, many researchers are interested in automating aspects of the surgical procedures [144]. Video of surgeons all over the world using the same robot to perform tasks like suturing, where properly grasping suture needles is an expert skill for which data could be useful for learning control policies as shown in [25]. However, these datasets only have video data and lack kinematic data from the robot due to proprietary access.

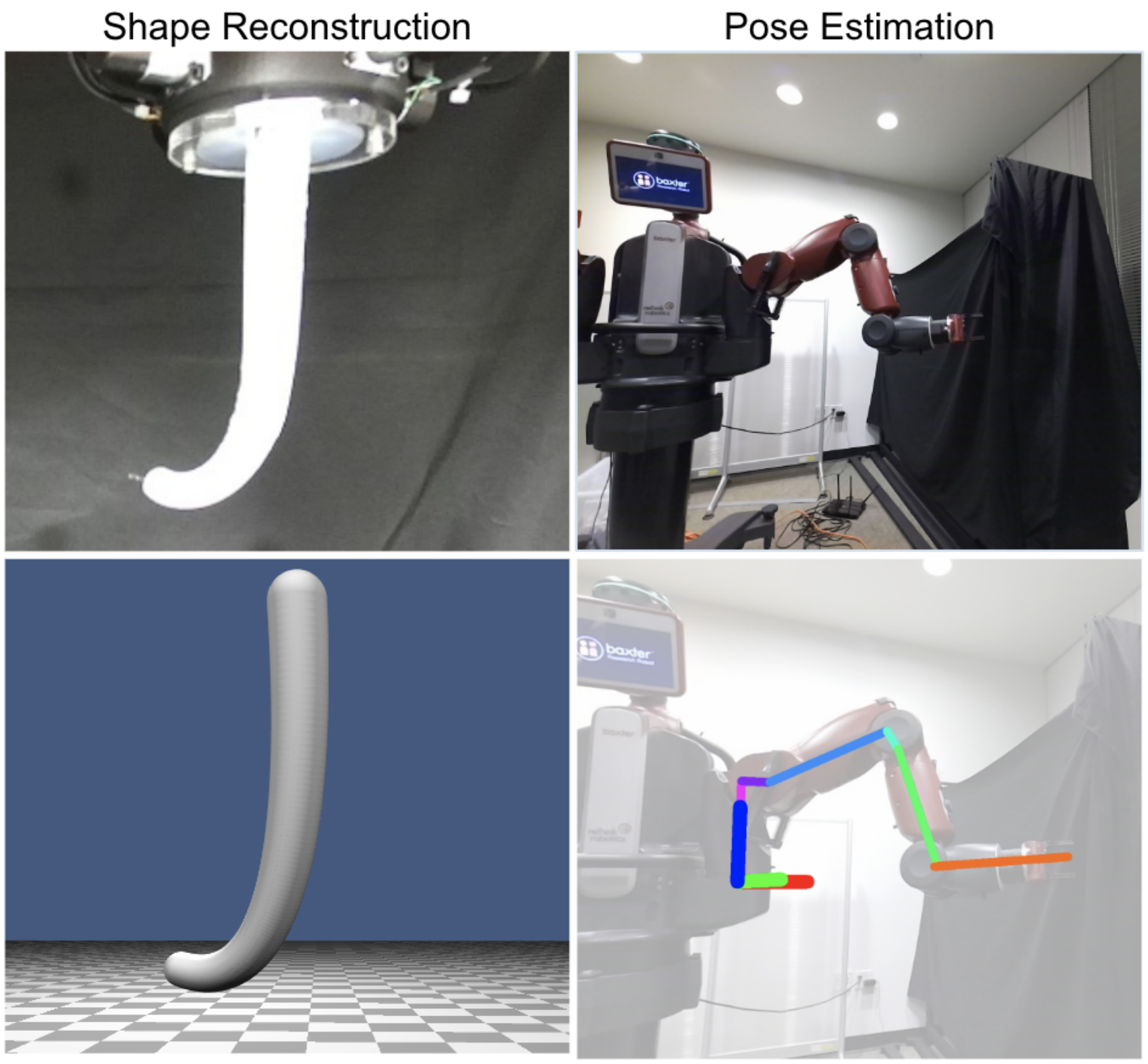

Figure 3.1. Robot shape reconstruction and pose estimation via differentiable rendering. The top row shows the real images. The bottom row shows the estimated robot shape (left) and pose (right).

Ultimately, in the above MIS and many other *in-the-wild* robot scenarios, tracking the robot pose and body configurations *directly from a camera* offers the greatest flexibility. They are easy to set up or are already recording, do not require access to internal robot

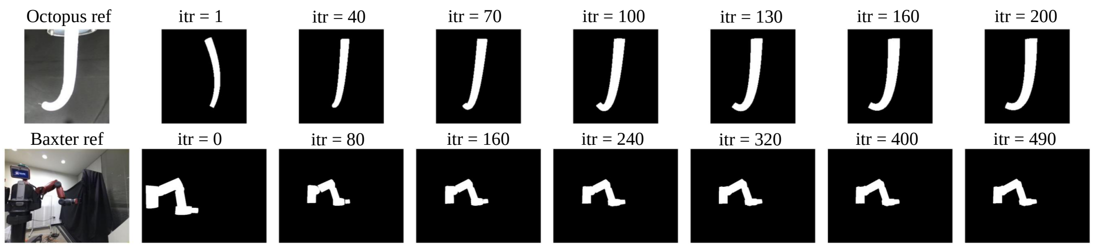

Figure 3.2. Visualization of the optimization process. We show the rendered silhouettes at different iterations to demonstrate the convergence of our algorithm.

sensors, are affordable and widely ubiquitous. Traditionally, fiducial markers, like ArUco marker [39] and AprilTag [90], are widely used for robot pose estimation. These markers are attached to the specific locations of the robot and the robot state parameters can be estimated by knowing the kinematics model. However in most unstructured environments, it is unrealistic to have these markers attached; furthermore, in dirty environments like MIS or constrained environments, these markers can be permanently obscured or occluded. For soft robot applications, their body deformations and their tendency for full-body contact with environments and objects make it frequently impractical for securing fiducials or template markers.

The most flexible way to estimate pose from a camera is to do marker-less tracking. With recent advancements in Computer Vision, Convolutional Neural Networks (CNN) present a promising way for marker-less feature detection [69][79] which no longer requires physical modification on the robot. In spite of the success of the CNN, training a neural network requires significant amounts of labeled datasets which is usually infeasible for soft robots and other robot prototypes, or where labeling of data is costly (e.g., MIS). Recently, in computer graphics, differentiable rendering has proved to be effective in image-based reconstruction by computing the derivative of images with respect to scene parameters such as camera pose and object geometry [59][58][75]. This could be translated to the task of marker-less pose tracking.

In this work, we demonstrate the capability of estimating robot pose and configuration directly from a camera, as shown in Fig. 3.1. The method works via the technique of differentiable rendering, and can be effective both in rigid-link robot manipulators as well as soft continuum robots. This is uniquely challenging, as soft continuum robots have an infinite number of configurations, while some rigid robot manipulators may not come with predefined CAD models for their users (these are typically proprietary). Under these constraints, we are still able to estimate a robot's state without knowledge of a high-resolution CAD by introducing a flexible method involving shape primitives that work across a wide range of robots. Our contributions are:

1. We propose a general framework for parameter estimation by utilizing differentiable rendering with geometrical shape primitives.
2. The framework generalizes to both rigid and soft continuum robots for parameter estimation
3. We investigate the novel loss functions to overcome the local minima when applying differentiable rendering to the objective of robot pose estimation.

To examine the effectiveness of our framework, we collect a synthetic and a real dataset for a soft continuum robot and reconstruct the robot shape by estimating the curve parameters. We also evaluate our method on robot pose estimation where the 6 Degree-of-Freedom (DOF) camera-to-robot pose for a Baxter robot is estimated with provided RGB images and joint encoder readings. The experimental results show that our method outperforms the state-of-the-art pose estimation algorithms.

3.1.1 Related Works

A few techniques in image-based estimation of robot poses and shapes have been previously explored.

Image-based Shape Reconstruction for Soft Continuum Robots

Image-based measurements of soft continuum robots are very task-specific and only work in certain environments. Techniques include using fluoroscopy [92, 47] and ultrasound [133, 24]. These image-based techniques require specific imaging sources which might not always be available. [2, 11, 102] consider using endoscopic images for shape reconstruction while the markers are required for identifying predefined feature points. Moreover, [29, 140] also introduce the shape reconstruction methods of using stereo images and depth data. In contrast, we will be focusing on markerless shape reconstruction from a single RGB camera.

Image-based Robot Pose Estimation

The common approach for image-based robot pose estimation is to attach the fiducial markers [39, 90] to known locations along the robot kinematic chain. Given the joint angles, the position of the marker in the robot base frame can be calculated and the robot pose can be derived by solving an optimization problem [73, 52, 106]. More recently, deep learning brings a promising way of marker-less pose estimation, where a CNN is trained to extract predefined feature points and the robot pose is estimated by solving the Perspective-n-Point problem [68, 69, 77, 79]. Meanwhile, rendering-based methods also demonstrate their advantages in robot pose estimation by using the high-resolution robot CAD model for more precise estimation [42, 66]. Our work utilizes differentiable rendering which requires no robot CAD model or large dataset for model training.

3.1.2 Methodology

We consider the problem of estimating the robot state parameters Θ from a single RGB image. Specifically, we estimate the robot pose and configurations by minimizing differences between the observed RGB image and a rendered reconstruction image. This is

Algorithm 2: Robot State Parameter Estimation via Differentiable Rendering

```
   Input   : Image frame $\mathbb{I}$, initialization $\Theta^{(0)}_{state}, \Theta^{(0)}_{verts}$
   Output  : Estimated robot state parameters $\Theta^{*}_{state}$
   // Generate robot masks
1  $\mathbb{M}^{ref} \leftarrow f_{mask}(\mathbb{I})$
   // Optimization loop
2  $\mathcal{L}_{min} = \infty$
3  for $i = 0$ to $N_o$ do
       // Section III-B
4      $\mathcal{M}^{(i)} \leftarrow reconstructMesh(\Theta^{(i)}_{state}, \Theta^{(i)}_{verts})$
       // Section III-C
5      $\mathbb{S}^{(i)} \leftarrow silhouetteRendering(\mathcal{M}^{(i)})$
6      $\mathcal{L}^{(i)} \leftarrow computeLoss(\mathbb{S}^{(i)}, \mathbb{M}^{ref})$
7      if $\mathcal{L}^{(i)} < \mathcal{L}_{min}$ then
8          $\mathcal{L}_{min} = \mathcal{L}^{(i)}$
9          $\Theta^{*}_{state} = \Theta^{(i)}_{state}$
10     $\Theta^{(i+1)}_{verts} = \Theta^{(i)}_{verts} - \lambda_{verts} \frac{\partial \mathcal{L}^{(i)}}{\partial \Theta^{(i+1)}_{verts}}$
11     $\Theta^{(i+1)}_{state} = \Theta^{(i)}_{state} - \lambda_{state} \frac{\partial \mathcal{L}^{(i)}}{\partial \Theta^{(i+1)}_{state}}$
12 return $\Theta^{*}_{state}$
```

formulated as follows:

$$\Theta^{*} = argmin_{\Theta} \mathcal{L}(f_{mask}(\mathbb{I}), f_{render}(\Theta)) \tag{3.1}$$

where f_{mask} processed the given RGB image $\mathbb{I}$ into a binarized mask image for the robot. The function f_{render} takes in the estimated parameters, reconstructs the robot mesh, and renders the reconstruction. We aim to estimate the state parameters by minimizing the objective loss function $\mathcal{L}$. A visual of the final optimization process are shown in Fig. 3.2. The process to get to this stage is described below.

The State Parameter Estimation Framework

The overall framework for state parameter estimation is described in the Algorithm 2. We first process the observed RGB image $\mathbb{I}$ into a binary mask $\mathbb{M}^{ref}$, which segments the robot pixel from the background. The binary mask contains value 1 for the pixels that belong to the robot and 0 otherwise. In our implementation, the segmentation is achieved by color segmentation for the soft continuum robot and a CNN-based semantic segmentation for the robot manipulator. We also initialize a robot mesh in a renderer as a set of geometrical primitive shapes with predefined vertices, edges, and faces.

During the iterative optimization process, we estimate the deformation of mesh vertices parameterized by Θ_{verts} and reconstruct the robot mesh with state parameters Θ_{state} (Section 12). We render a silhouette image $\mathbb{S}$ from the reconstruction and compare it with the reference masked image $\mathbb{M}^{ref}$. A loss $\mathcal{L}$ is computed based on the curated objective functions (Section 12). Since the full reconstruction and rendering pipeline is differentiable, a gradient on the loss may be taken with respect to the parameters and the objectives can be optimized (lines 11-12 in Algorithm 2). We iterate the optimization process for N_o times and output the state parameters that minimize the objective loss.

Reconstruct Robot Mesh with Geometric Primitives

In this section, we describe the methods of reconstructing the robot mesh using geometric primitives for the soft and rigid robot, respectively. Note that state parameters Θ_{state} and mesh vertex parameters Θ_{verts} are defined differently according to their body types.

Mesh Reconstruction for Soft Continuum Robot. The shape of a soft continuum robot can be described in several ways, most easily using a constant curvature model [107]. However, since this is a limiting approximation, instead a better model chosen is a Bézier curve model, which expresses a smooth and continuous curve with arbitrary curvature in 3D space. A Given a set of N control points $\{\mathbf{c}_i | \mathbf{c}_i \in \mathbb{R}^3\}_{i=0}^{N}$, the shape of

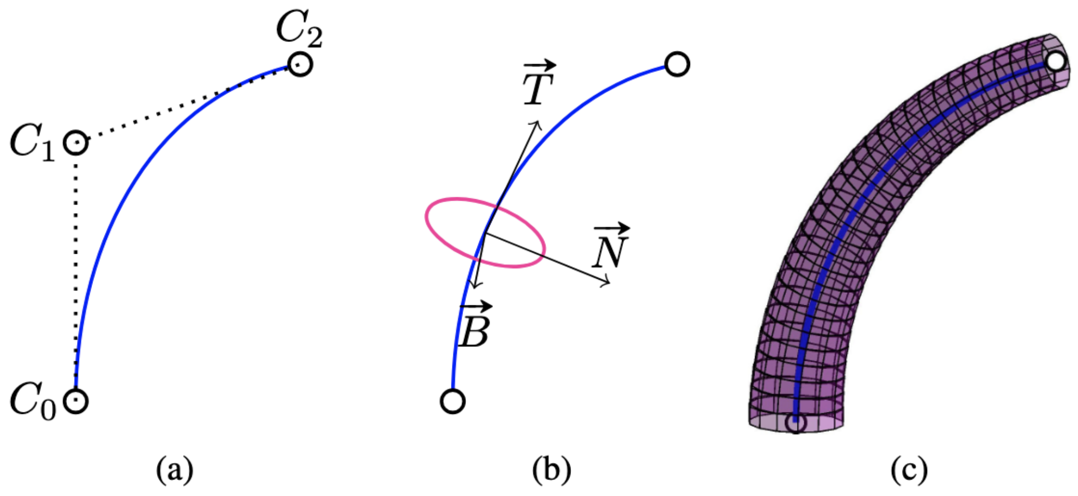

Figure 3.3. Illustration of surface mesh construction for soft continuum robot, with (a) the Bézier curve and control points, (b) the Frenet–Serret frame of cross section, and (c) the constructed surface mesh.

the curve is defined as:

$$\mathbf{p}(s) = \sum_{i=0}^{N} \frac{N!}{i!(N-i)!}(1-s)^{N-i}s^i\mathbf{c}_i \ , \ 0 \leq s \leq 1, \tag{3.2}$$

For simplicity, we use a quadratic Bézier curve ($N = 2$) and estimate the state of the control points Θ_{state} (see Fig. 3.3).

In general, the surface mesh for a continuum robot can be approximated as the tubular structure [72]. A tubular surface is defined as a union of cross sections, and each cross-section is centered at the axis along the 3D curve, as shown in Figure 3.3(a). To describe 3D coordinate frames along a quadratic Bézier curve, we compute the Frenet–Serret frame which is defined by a unit vector $\mathbf{T}$ tangent to the curve, a unit vector $\mathbf{N}$ normal to the curve, and a unit vector $\mathbf{B}$ perpendicular to the tangent and normal vectors

(Figure 3.3(b)). The Frenet–Serret coordinates, parameterized by s, are defined as:

$$\begin{aligned} \mathbf{T}(s) &= \frac{\mathbf{p}'(s)}{\|\mathbf{p}'(s)\|} \\ \mathbf{N}(s) &= \frac{\mathbf{T}'(s)}{\|\mathbf{T}'(s)\|} = \frac{\mathbf{p}'(s) \times (\mathbf{p}''(s) \times \mathbf{p}'(s))}{\|\mathbf{p}'(s)\| \, \|\mathbf{p}''(s) \times \mathbf{p}'(s)\|} \\ \mathbf{B}(s) &= \mathbf{T}(s) \times \mathbf{N}(s) = \frac{\mathbf{p}'(s) \times \mathbf{p}''(s)}{\|\mathbf{p}'(s) \times \mathbf{p}''(s)\|} \end{aligned} \tag{3.3}$$

where $\mathbf{p}'(s), \mathbf{p}''(s)$ are the first and second derivatives of the quadratic Bézier curve model:

$$\begin{aligned} \mathbf{p}(s) &= (1-s)^2\mathbf{c}_0 + 2(1-s)s\mathbf{c}_1 + s^2\mathbf{c}_2 \\ \mathbf{p}'(s) &= 2(1-s)(\mathbf{c}_1 - \mathbf{c}_0) + 2s(\mathbf{c}_2 - \mathbf{c}_1) \\ \mathbf{p}''(s) &= 2(\mathbf{c}_2 - 2\mathbf{c}_1 + \mathbf{c}_0). \end{aligned} \tag{3.4}$$

Each cross-section is approximated as a circle with the radius $r(s)$, and the corresponding tubular surface is defined as:

$$\mathbf{S}(s, \phi) = \mathbf{p}(s) + r(s)\left[-\mathbf{N}(s)\cos\phi + \mathbf{B}(s)\sin\phi\right] \tag{3.5}$$

with $\phi \in [0, 2\pi]$. Since a point on the tubular surface can be specified by s and ϕ, we compute the mesh vertices by discretizing the tubular surface. The mesh vertices are then defined by a set of points on the tubular surface with two additional points at both ends of the curve,

$$\mathcal{V} = \{\mathbf{S}(s_i, \phi_i), \mathbf{p}(0), \mathbf{p}(1) \mid i = 1, ..., N_d\} \tag{3.6}$$

where s_i, ϕ_i are discrete points for surface vertices. The example of reconstructed surface mesh is shown in Figure 3.3(c). During the optimization process, we adjust the mesh vertices by optimizing the radius of the cross sections $\Theta_{verts} := r(s)$, and the robot mesh is formed with adjusted vertices.

Mesh Reconstruction for Robot Manipulator. A robot manipulator can generally be described as a chain of rigid links, and the 3D robot mesh is separated into a set of individual meshes of primitive shape. The number of individual meshes equals the number of rigid links and the meshes are connected by the rigid body transformations which indicate the position and rotation with respect to the robot base frame $\{b\}$. The transformation matrices $\mathbf{T}^b_n(q_1, ..., q_n) \in SE(3)$ are parameterized by joint angles and can be computed from forward kinematics.

We initialize the set of individual meshes as primitive shapes (e.g. boxes or cylinders) with predefined edges, faces, and vertices $\mathcal{V}_{\texttt{primitive}}$. Note that anything can be a primitive shape as long as there are only a few tunable parameters defining it, so a link shape template could be used. Regardless, during the optimization process, we adjust the robot mesh by estimating the offsets of each vertex:

$$\mathbf{v} = \mathbf{v}_{\texttt{primitive}} + \mathbf{v}_{\texttt{offset}} \tag{3.7}$$

where $\mathbf{v}_{\texttt{primitive}} \in \mathcal{V}_{\texttt{primitive}}$ is the vertex initialized with the primitive shape mesh and $\mathbf{v}_{\texttt{offset}}$ is the corresponding offset vector. The set of offset vectors has the same number of elements as $\mathcal{V}_{\texttt{primitive}}$, and is optimized at each iteration through gradient descent $(\Theta_{verts} := \mathcal{V}_{\texttt{offset}})$.

To compose the robot mesh $\mathcal{M}$ for pose estimation, each individual mesh needs to be connected by the forward kinematics and transformed to the camera frame. Let $\mathbf{v}^n \in \mathbb{R}^3$ be a vertex of the n-th link mesh, we transform the vertex to the camera frame as:

$$\overline{\mathbf{v}}^c = \mathbf{T}^c_b \mathbf{T}^b_n(q_1, ..., q_n)\overline{\mathbf{v}}^n \tag{3.8}$$

where $\bar{\cdot}$ represents the homogeneous representation of a point (e.g. $\overline{\mathbf{v}} = [\mathbf{v}, 1]^T$). $\mathbf{T}^b_n$ obtained from forward kinematics transforms mesh vertices to the robot base frame and $\mathbf{T}^c_b$ is the robot-to-camera transformation which is estimated using the Algorithm 2

$(\Theta_{state} := \mathbf{T}_b^c)$.

Differentiable Rendering and Loss Functions

To render the image for robot mesh $\mathcal{M}$, we use the PyTorch3D differentiable render [101] for silhouette rendering. We set up the silhouette renderer with a perspective camera and a *SoftSilhouetteShader* which does not apply any lighting and shading. The differentiable renderer applies the rasterization algorithm which finds the mesh triangles that intersect each pixel and weights the influence according to the distance along the z-axis. Finally, the *SoftSilhouetteShader* computes pixel values of the rendered silhouette image using the sigmoid blending method [75].

Objective Loss Functions for Shape Reconstruction. To minimize the difference between the reconstructed silhouette image and the observed binary mask, the commonly used mask loss is applied. The mask loss computes the sum of the mean square error for every pixel,

$$\mathcal{L}_{mask} = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \left(\mathbb{S}(i,j) - \mathbb{M}^{ref}(i,j)\right)^2 . \tag{3.9}$$

H and W is the image height and width, $\mathbb{S}$ is the rendered silhouette image and $\mathbb{M}^{ref}$ is the reference binary mask.

The mask loss will have non-informative gradients when there is no overlap (e.g. $\mathbb{S}(i,j) = 0$ but $\mathbb{M}^{ref}(i,j) = 1$). Therefore we use an additional keypoint loss to guide the optimization from local minima when the silhouettes do not overlap. The keypoints loss is defined as:

$$\mathcal{L}_{keypoint} = \sum_{i=1}^{K} \|\pi(\mathbf{p}_i) - \hat{\mathbf{x}}_i\|_2 \tag{3.10}$$

where K is the number of keypoints, $\hat{\mathbf{x}}_i$ is the i-th 2D keypoint extracted from center line of the reference mask $\mathbb{M}^{ref}$ as shown in Fig. 3.4, and $\mathbf{p}_i$ is the corresponding 3D keypoint on the Bézier curve. $\pi(\cdot)$ is the camera projection operator. Finally, the shape reconstruction loss is defined as:

$$\mathcal{L}_{shape} = \lambda_{mask}\mathcal{L}_{mask} + \lambda_{keypoint}\mathcal{L}_{keypoint} \tag{3.11}$$

with $\lambda_{mask}, \lambda_{keypoint}$ as loss weights.

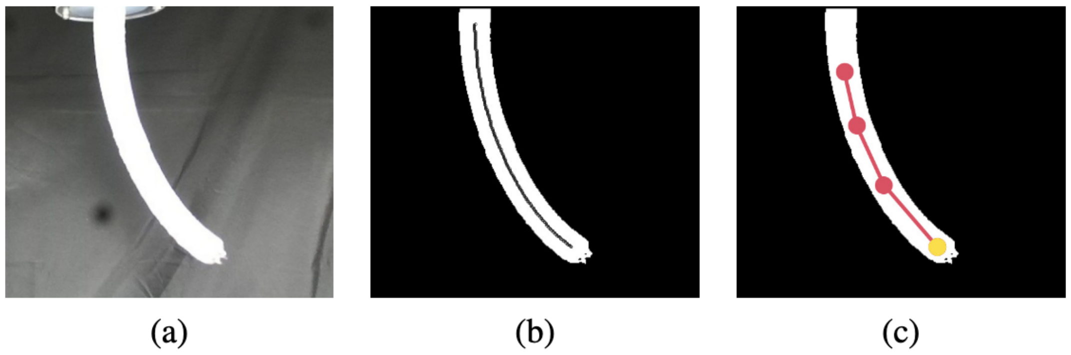

Figure 3.4. The pre-processing of soft octopus arm images, with (a) the observed RGB image, (b) the reference binary mask with center-line and (c) the predefined keypoints and the endpoint (yellow).

Objective Loss Functions for Pose Estimation. For robot pose estimation, poor initialization would hamper the performance of the optimization algorithm by converging to local minima. In addition to the commonly used mask loss (Eq. 3.9), we propose distance loss and appearance loss to aid the optimization. The distance loss utilizes the distance map to propagate the gradient information to the entire image. The distance map $\mathbb{D}^{ref}$ is defined as:

$$\mathbb{D}^{ref}(i,j) = \begin{cases} 0, & \text{if } \mathbb{M}^{ref}(i,j) = 1 \\ \frac{dist(i,j)}{\gamma}, & \text{otherwise} \end{cases} \tag{3.12}$$

where $dist(i,j)$ is the distance from the pixel (i,j) to the closest pixel that has positive value 1, and γ is a discount factor (Fig. 3.5). We use the *scikit-fmm* package[1] which implements the fast marching method [118] for computing the distance map. The distance loss is then calculated as:

$$\mathcal{L}_{dist} = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \mathbb{S}(i,j) * \mathbb{D}^{ref}(i,j) \tag{3.13}$$

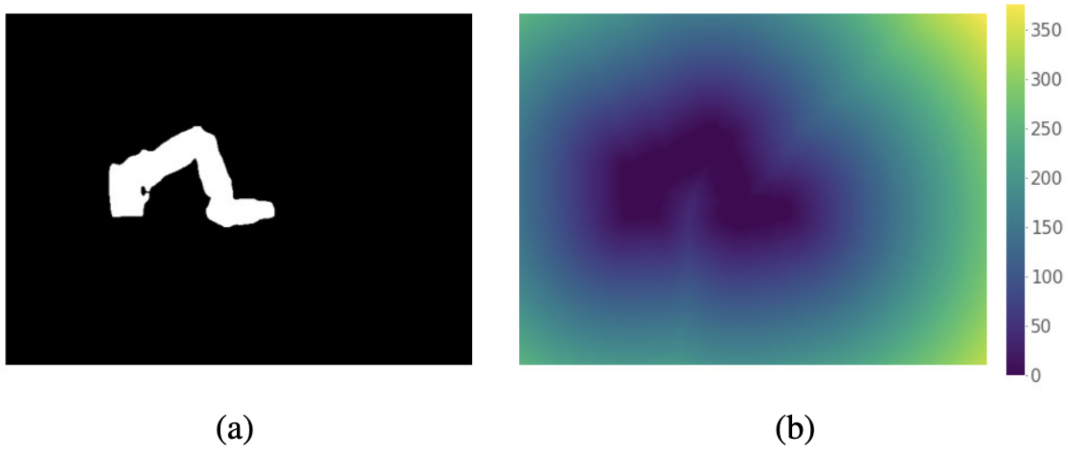

Figure 3.5. Visualization of the distance map, with (a) the reference binary mask and (b) the corresponding distance map.

Since the reconstructed mask should have the same appearance as the observed mask, we introduce the appearance loss to force them to have the same number of positive pixels:

$$\mathcal{L}_{app} = \left\| \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \mathbb{S}(i,j) - \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \mathbb{M}^{ref}(i,j) \right\| \tag{3.14}$$

The appearance loss is effective for preventing the robot pose from being too far or too close to the camera, through regulating the size of the rendered mask. Finally, the objective loss for pose estimation is defined as:

$$\mathcal{L}_{pose} = \lambda_{mask}\mathcal{L}_{mask} + \lambda_{dist}\mathcal{L}_{dist} + \lambda_{app}\mathcal{L}_{app} \tag{3.15}$$

[1]https://pythonhosted.org/scikit-fmm

where $\lambda_{mask}, \lambda_{dist}, \lambda_{app}$ are weights for the loss functions.

3.1.3 Experiments and Results

Datasets and Evaluation Metrics

Tendon-driven Octopus Arm dataset. Our experimental evaluations are conducted on a physical prototype of tendon-driven octopus arm, visible in Fig. 3.1. It consists of a tapered cylinder of Ecoflex 00-30 (Smooth-On, Inc., Macungie, PA, USA) of length 200 mm, base and tip radius of 10 and 6 mm, respectively. It contains four channels for tendon actuation. The tendons are rigidly attached at the tip, and connected to spools actuated by harmonic drive motors at the base. The motors displace the tendons, leading to motions of the octopus arm. We collected the images using the ZED camera and the ground-truth robot shape is obtained with stereo reconstruction. For evaluation, we compare the center line of the reconstructed robot shape with the ground truth shape. The 2D and 3D errors of the center line are computed using the Euclidean distance of discrete points.

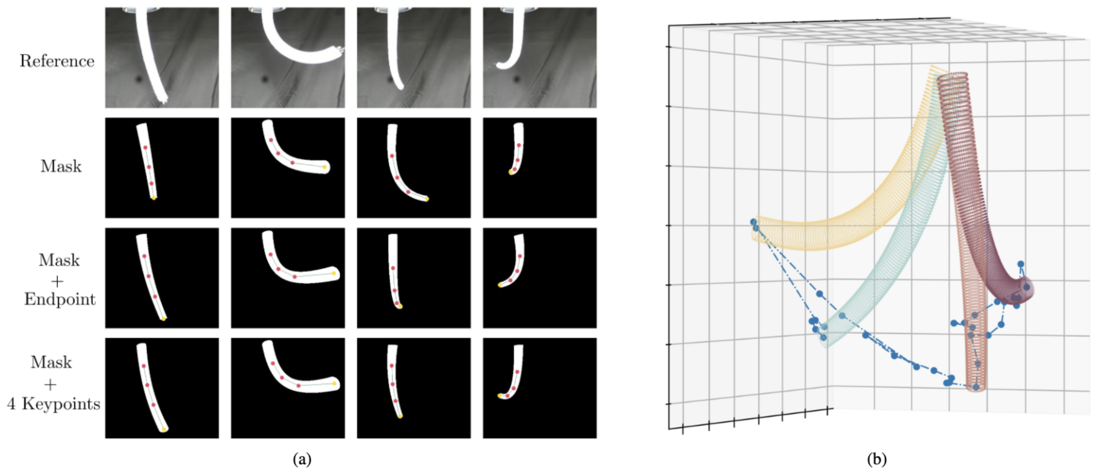

Figure 3.6. Reconstruction results for the Octopus Arm dataset. (a) The reference RGB images and shape reconstruction results for different losses are shown for 4 example frames that cover a large range of motion and (b) the reconstructed 3D robot shape of the picked frames and the entire robot trajectory.

Table 3.1. 2D and 3D Error (mean e and standard deviation σ) of Shape Reconstruction on real Octopus Arm Dataset.

Losses	e_{2D} (pixel)	σ_{2D}	e_{3D}	σ_{3D}
Mask	12.462	8.690	8.720	2.534
Mask + endpoint	3.898	2.216	7.299	3.900
Mask + 4 keypoints	**3.276**	**0.785**	**6.915**	**2.096**

Baxter dataset. The Baxter dataset from [79] provides 100 image frames of 20 different robot poses. The ground-truth end-effector position in 2D and 3D are provided. This dataset includes the challenging scenarios where the robot manipulator is self-occluded. The Percentage of Correct Keypoints (PCK) metric of the end-effector will be calculated according to [79], where the end-effector position in the camera frame is calculated based on the estimated robot pose.

Shape Reconstruction for Soft Continuum Robot

Implementation details. The RGB images are pre-processed to binary masks and the 2D center-line are extracted from the reference binary mask using the *scikit-image* (https://scikit-image.org) package, which implements the fast skeletonization method [147]. We arbitrarily predefined 4 keypoints along the center-line for loss computation, as shown in Fig. 3.4. For computing the mesh vertices, s is discretized to 100 and θ is discretized to 40 number of evenly spaced points. For the loss function, we set $\lambda_{mask} = 1$ and $\lambda_{keypoint} = 100$. We initialize the control points randomly but make sure the initialized mesh is within the camera frustum. The optimization loop is run for 200 iterations with a learning rate of 0.2.

Evaluation on the Octopus Arm Dataset. We evaluate our shape reconstruction method with different loss functions described in Section 12. For the keypoint loss, we experimented with only using the endpoint and using all 4 keypoints. We report the averaged 2D and 3D center-line error and the results are shown in Table 3.1. We can see that the error is dropped significantly by combining the mask loss and keypoint loss with only the endpoint. Considering more keypoints further improves our performance of shape reconstruction as they provide more guidance for optimization. The qualitative results are shown in the Fig. 3.6, where we show the rendered silhouette images of the reconstructed robot mesh (left) and, the reconstructed 3D robot shape, and the robot trajectory (right).

Pose Estimation for Robot Manipulator

Implementation details. To segment the robot from the background, we trained the DeepLabV3 [20] with 10K synthetic image data generated using the CoppeliaSim [110]. We applied Domain Randomization [128] so that the trained network can generalize to the real images. We initialize the primitive shape meshes for each link with the length, width, and height that are described in the robot description file. The deformed vertices $\mathbf{v}_{\texttt{deformed}}$ are initialized to zeros and the robot poses are initialized randomly but within the camera frustum. For loss function, we set the weights as $\lambda_{mask} = 1, \lambda_{dist} = 1, \lambda_{app} = 1$ and $\gamma = 100$ when computing the distance map. We optimize parameters for 500 iterations with a learning rate of 1e-2 for camera pose parameters and 1e-4 for the deformed vertices.

Evaluation on Baxter Dataset. We evaluate our method of robot pose estimation on the Baxter dataset and compare it against the state-of-the-art methods [69, 79], as shown in Table 3.2. We applied our method with two different shape primitive meshes, the box and cylinder. We also report the performance of using the robot CAD model with differentiable rendering. The Percentage of Correct Keypoints (PCK, higher is better) results are reported for both 2D and 3D at different thresholds. The experimental results show that our method of using primitive shapes outperforms the state-of-the-art methods and achieves comparable performance with using the high-resolution robot CAD model.

Table 3.2. Comparison of our methods with the state-of-the-art methods on robot pose estimation.

	PCK2D				PCK3D			
	@50 pixel	@100 pixel	@150 pixel	@200 pixel	@100 mm	@200 mm	@300 mm	@400 mm
DREAM [69]	0.33	0.52	0.62	0.64	0.32	0.43	0.54	0.66
Optimized Keypoints [79]	0.69	0.88	0.93	0.95	0.47	0.74	0.86	0.90
Ours (box)	0.65	**0.94**	**0.95**	0.95	**0.8**	**0.95**	0.95	0.95
Ours (cylinder)	**0.80**	0.91	0.93	0.95	0.71	0.93	0.94	0.95
Ours (CAD)	0.74	0.90	0.94	**1.0**	0.78	0.93	**0.97**	**1.0**

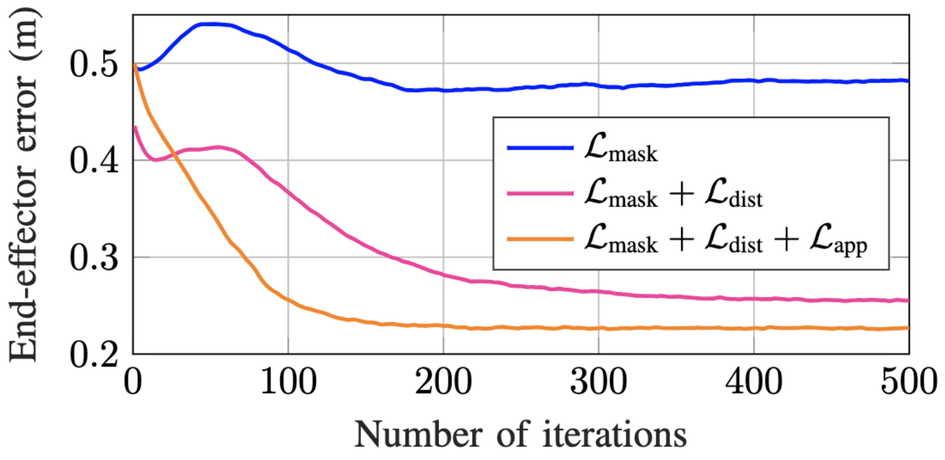

Figure 3.7. Ablation study of the loss functions for robot pose estimation. Notice that the averaged error is large because of outliers, and 95% of the estimates have less than 4 cm error as shown in Table 3.2.

Ablation Study on Loss Functions. Here, we study the effectiveness of the loss functions proposed in Section 12 using the Baxter dataset. We experiment with our robot pose estimation method (cylinder) with different loss combinations and calculate 3D end-effector error using the Euclidean Distance. We plot the average 3D end-effector error at each iteration in Fig. 3.7. With only the mask loss, the algorithm suffers from bad initialization and cannot converge robustly. The distance loss helps the convergence by propagating the gradient information to every image pixel. Finally, by combining all three losses, we achieve a more robust convergence for robot pose estimation.

3.1.4 Conclusion

In this work, we demonstrate the capability of measuring robot pose and configuration state parameters directly from a camera, as shown in Fig. 3.1. The method works via the technique of differentiable rendering, and can be effective both in rigid-link robot manipulators as well as soft continuum robots. we show that several definitions for optimization losses are useful to overcome the local minima when applying differentiable rendering to the objective of robot pose estimation. We evaluated our method on relatively unstructured environments of continuum and rigid robots showing its efficacy in pose estimation. Ultimately, this work helps to enable robot state estimation and tracking *in the wild*, with greater opportunities in useful dataset curation, behavioral cloning and visual learning.

3.2 Tracking Snake-Like Robots

Unlike their stationary counterparts, mobile robots are designed to navigate through the physical world in environments that are often too treacherous for humans such as the deep sea [65] and even other planets [121]. With mobile robots acting as surrogates for humans, exploration for research and search and rescue missions in extreme environments are conducted without risking human lives [135]. A growing class of mobile robots involves ambulatory systems. These ambulatory mobile robots (AMRs) have specialized articulated robotic designs for enhanced mobility and stability on uneven ground techniques in order to navigate broader terrains. AMRs include but are not limited to quadruped robots [5], flying drones [1], and snake-like and serpentine robots [138, 105].

To ensure the safe operation of AMRs in complex environments, various sensors are integrated into their systems. These sensors aid in localizing the robot and understanding its surroundings, though this can introduce increased complexity in real-world deployments. A more streamlined approach involves tracking AMRs using cameras. Cameras, given their ease of installation and portability, are better for navigating challenging terrains. For example, in the Mars 2020 NASA mission, where the Mars Helicopter utilized onboard cameras to scout the landscape and guide the Perseverance rover's exploration. As we look to the future, exploratory and search-and-rescue missions likely involve collaborative efforts between multiple robots, and the ability to track one robot using a camera mounted on another will be crucial.

In this work, we address the problem of tracking snake-like robots from a single

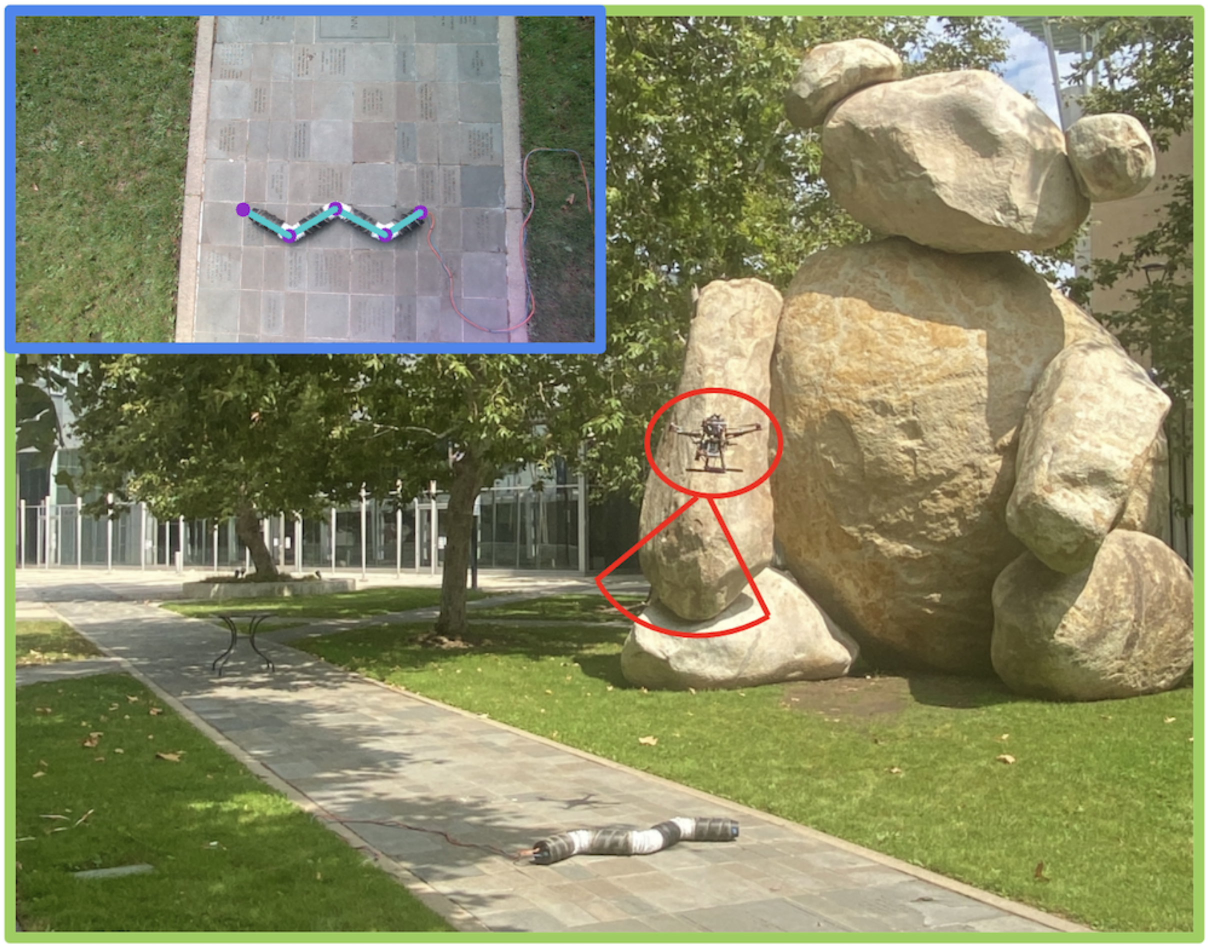

Figure 3.8. A snake-like robot, Arcsnake [105], is tracked on camera in the outdoor environment by a hovering drone.

camera. Along the lines of the Mars Helicopter's mission, we aim to bring robot state estimation from camera data to snake-like robots, and by extension, other AMRs, to aid in future exploratory missions. By estimating the pose and state of an AMR, drones can provide more detailed guidance when providing mapping of the environment [132]. Our focus is on snake robots that draw inspiration from biological snakes [97] and are currently funded by NASA for exploration on extraterrestrial planetary bodies [12]. Toward this end, we recognize a fundamental need for being able to track AMRs using only a monocular camera. These techniques will also become foundational in the future to deploying robots in search-and-rescue missions or leveraging autonomous robot teams for work in the remote wilderness.

The overall tracking approach involves first a method for automatic robot mask generation. Leveraging this mask, we present a tracking technique that seamlessly integrates differentiable rendering with the Kalman filter, ensuring precise online state estimation. We conduct experiments in both laboratory and outdoor environments (Fig. 3.8). Through both qualitative and quantitative evaluations, we demonstrate the effectiveness of our method in different scenarios. Our contributions are threefold:

- We present the first work on marker-less state estimation for a snake robot from a single monocular camera.
- Our method combines differentiable rendering with a Kalman filter, and simultaneously estimates the joint angle and the pose of a snake robot.
- Validation of the effectiveness of the algorithm on a snake robot in both structured and unstructured environments, achieving a localization accuracy of 0.05 m for the robot base position and 0.11 rad on the robot's joint states.

3.2.1 Related Works

For a broader category of mobile robots, the primary focus of state estimation has been on localizing the robot within its surroundings. For instance, Milella et al. [86] utilizes visually distinctive features on stereo images for localization. Several other works [40, 152, 27] have proposed methods that take into account the environment dynamics and potential measurement errors to enhance localization accuracy.

However, in the realm of snake robots, state estimation becomes even more intricate due to the need to consider joint angles for accurate 3D space modeling. Historically, state estimation for snake robots has relied on the robot's internal proprioceptive sensors, as highlighted by works like Rollinson et al. [111, 112]. Then, the filtering methods, like the

Unscented Kalman Filter and Extended Kalman Filter [57, 131], have been employed to account for the measurement error for real-time estimation.

In this work, we seek to estimate both the position and joint angle of the snake robot using only images. This approach not only simplifies the estimation process but also enhances the robot's adaptability in outdoor scenarios.

3.2.2 Methodology

```
Algorithm 3: Online State Estimation
   Input   : Initialized robot state $\mathbf{x}_{0|0}, \Sigma_{0|0}$
   Output  : Estimated robot state $\mathbf{x}_{t|t}, \Sigma_{t|t}$
1  while receive new image $\mathbb{I}_t$ do
       // Motion Model
2      $\mathbf{x}_{t|t-1}, \Sigma_{t|t-1} \leftarrow motionModel(\mathbf{x}_{t-1|t-1}, \mathbf{v}_{t-1}, \Sigma_{t-1|t-1})$
       // Observation from Image
3      $\mathbb{M}_t^{ref} \leftarrow f_{seg}(\mathbb{I}_t)$
4      $\mathbf{m}_t \leftarrow computeMoments(\mathbb{M}_t^{ref})$
       // Observation Model
5      $\mathcal{M}_{t|t-1} \leftarrow reconstructMesh(\mathbf{x}_{t|t-1})$
6      $\mathbb{M}_{t|t-1}^{pred} \leftarrow renderPrediction(\mathbf{x}_{t|t-1}, \mathcal{M}_{t|t-1})$
7      $\hat{\mathbf{m}}_t \leftarrow computeMoments(\mathbb{M}_{t|t-1}^{pred})$
8      $H_t = \frac{\partial \hat{\mathbf{m}}_t}{\partial \mathbf{x}_{t|t-1}}$
       // Compute the Residual
9      $\mathbf{y}_t = \mathbf{m}_t - \hat{\mathbf{m}}_t$
       // Update Belief
10     $K_t = \Sigma_{t|t-1} H_t^\top (H_t \Sigma_{t|t-1} H_t^\top)^{-1}$
11     $\mathbf{x}_{t|t} = \mathbf{x}_{t|t-1} + K_t \mathbf{y}_t$
12     $\Sigma_{t|t} = (I - K_t H_t) \Sigma_{t|t-1}$
       // Refine with Image Loss
13     for number of refinement steps do
14         $\mathcal{M}_{t|t} \leftarrow reconstructMesh(\mathbf{x}_{t|t})$
15         $\mathbb{M}_{t|t}^{pred} \leftarrow renderPrediction(\mathbf{x}_{t|t}, \mathcal{M}_{t|t})$
16         $\mathcal{L}_t \leftarrow computeLoss(\mathbb{M}_{t|t}^{pred}, \mathbb{M}_t^{ref})$
17         $\mathbf{x}_{t|t} = \mathbf{x}_{t|t} - \lambda \frac{\partial \mathcal{L}_t}{\partial \mathbf{x}_{t|t}}$
       // Update Velocity
18     $\mathbf{v}_t \leftarrow computeVelocity(\mathbf{x}_{t|t}, \mathbf{x}_{t-1|t-1})$
```

The overall proposed approach follows an online state estimation method combining differentiable rendering of a robot mask, with image moment prediction, a robot motion model, and a Kalman filter to estimate the joint angle and the pose of a mobile robot from a single camera. The method includes, additionally, refinement steps and velocity update steps to enhance the accuracy of the estimation, as well as model transfer techniques to reduce computation and memory costs so that the method can run on modest hardware. The details follow in the next section, and Algorithm 3 outlines the main steps of the method.

Motion Model with Belief Propagation

For AMR navigation, the robot state, denoted by $\mathbf{x}_t$, can encapsulate various attributes such as joint angles, camera-to-robot transformations, and other necessary parameters at time t. In this work, we define the robot state as $\mathbf{x} := [\theta, \mathbf{q}, \mathbf{b}]$, where $\theta \in \mathbb{R}^N$ is the robot joint angle (N is the number of joints), $\mathbf{q}$ is the quaternion, and $\mathbf{b}$ is the translational vector for the first link of the robot. The quaternion and the translational vector are parametrizations of the $\mathbf{T}_b^c \in SE(3)$, which is the robot pose in the camera frame.

The next state of the robot is predicted with a motion model, based on its previous state and velocity. This prediction phase provides a rough direction for belief propagation. We will model the robot's motion using a simple linear relationship:

$$\mathbf{b}_{t|t-1} = \mathbf{b}_{t-1|t-1} + \mathbf{v}_{t-1}\Delta t \tag{3.16}$$

where we try to predict the position of the robot $\mathbf{b}_{t|t-1}$ at time t by considering the previous robot position $\mathbf{b}_{t-1|t-1}$, the velocity $\mathbf{v}_{t-1}$, and the time step Δt. We will make the assumption that there is negligible process noise (i.e., imperfections in the system's motion model are negligible as compared to observation noise), leading to the following expression for the propagation of the covariance matrix:

$$\Sigma_{t|t-1} = F_t \Sigma_{t-1|t-1} F_t^\top \tag{3.17}$$

In this case, F_t is the identity matrix, reflecting our assumption that the motion model follows a linear relationship without any non-linear or stochastic effects.

Automatic Mask Generation for Segmentation

The proposed state estimation algorithm requires segmenting the robot from images, but manually labeling the robot masks can be highly time-consuming. Recently, the zero-shot generalizable segmentation model, Segment Anything Model (SAM) [63], allows automatic robot mask generation with simple bounding box prompts.

Given the binary robot mask of the previous frame, $\mathbb{M}_{t-1} \in \mathbb{R}^{H \times W}$, the bounding box prompt for the current frame, $\mathcal{B}_t := (u_{min}, v_{min}, u_{max}, v_{max})$, is estimated by a mask-to-box operation,

$$(u_{min}, v_{min}) = \min\{(u, v) \,|\, \mathbb{M}_{t-1}[u, v] \neq 0\} \tag{3.18}$$

$$(u_{max}, v_{max}) = \max\{(u, v) \,|\, \mathbb{M}_{t-1}[u, v] \neq 0\} \tag{3.19}$$

Then, the SAM is utilized to generate the robot mask of the current frame, given the bounding box prompt $\mathcal{B}_t$, as shown in Fig. 3.9. To ensure the robustness of the bounding box prompt, the robot mask is dilated before performing the mask-to-box operation.

Using SAM for robot mask generation can, however, be slow as SAM is not optimized for real-time application (around 0.5 seconds per frame using a single Nvidia GeForce RTX 4090 GPU). To achieve real-time performance, we utilize the robot masks generated from SAM to train a lightweight neural network for segmentation. Specifically, we employ DeepLabV3+ [21], a popular semantic segmentation architecture, to segment the robot

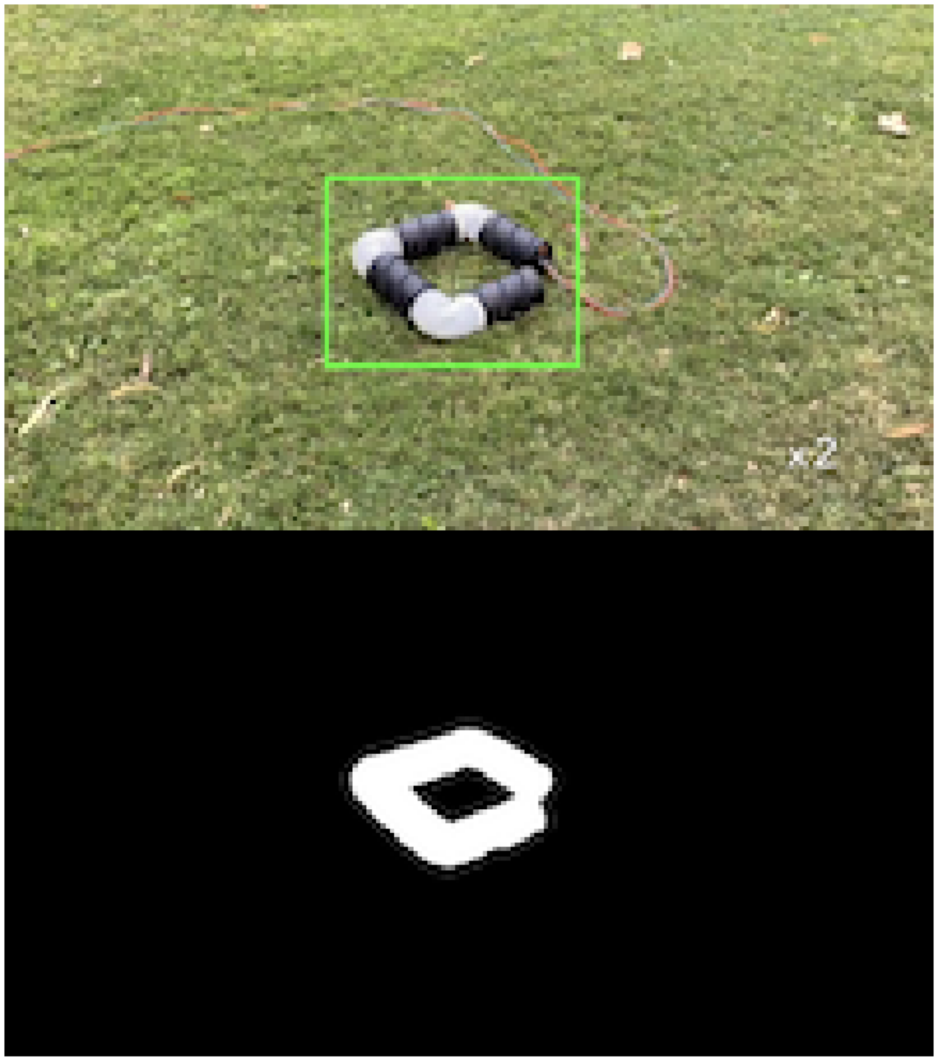

Figure 3.9. Example of the bounding box prompt generated by mask-to-box operation (top) and the corresponding robot mask generated using SAM (bottom).

from RGB images during the online estimation process. By training DeepLabV3+ with the generated masks, we ensure that our system can segment the robot in real-time with modest memory and computation requirements, effectively enabling realistic deployment in the wild.

Observation Model for Belief Propagation

In this section, we introduce the mapping from the predicted robot states $\mathbf{x}_{t|t-1}$ to the observation of image moment [13] $\hat{\mathbf{m}}_t$ in the proposed algorithm 3.

Given the predicted robot states $\mathbf{x}_{t|t-1}$, which includes joint angle and robot pose, we first reconstruct the robot mesh by interconnecting individual robot body parts through forward kinematics. For a snake-like (serpentine) robot, we approximate each individual robot body part as a cylinder with the dimension mentioned in [117, 105]. Given a mesh vertex $\mathbf{r}^n \in \mathbb{R}^3$ on the n-th robot link, this vertex undergoes a transformation into the robot base frame considering the joint angle:

$$\bar{\mathbf{r}}^b = \mathbf{T}_n^b(\theta)\bar{\mathbf{r}}^n \tag{3.20}$$

where $\bar{\cdot}$ represents the homogeneous representation of a point (i.e. $\bar{\mathbf{r}} = [\mathbf{r}, 1]^T$), and $\mathbf{T}_n^b(\theta)$ is the coordinate frame transformation obtained from the forward kinematics [31].

Having the reconstructed robot mesh and the predicted robot base-to-camera transformation, $\mathbf{T}_b^c$, the PyTorch3D differentiable renderer [101] comes into play to produce a virtual-model-derived, or rendered robot mask. By referencing techniques similar to those in [78], a differentiable silhouette renderer paired with a perspective camera is employed. The *SoftSilhouetteShader* is specifically leveraged to compute pixel values that form the robot mask.

With the rendered robot mask, $\mathbb{M}$, the image moments become computable as:

$$M_{ij} = \sum_u \sum_v u^i v^j \mathbb{M}(u, v) \tag{3.21}$$

Then, we derive the centroid, which is our observation for belief propagation, by:

$$\hat{\mathbf{m}} = \begin{bmatrix} \frac{M_{10}}{M_{00}} & \frac{M_{01}}{M_{00}} \end{bmatrix}^\top \tag{3.22}$$

We employ pytorch autograd [95] to track the gradient of each step and compute the observation matrix H by collecting the derivatives of the image moment $\hat{\mathbf{m}}$ with respect to the robot states $\mathbf{x}_{t|t-1}$.

Finally, an Extended Kalman Filter (EKF) [57] is employed to update the belief of the robot states (lines 9-12 in Algorithm 3), which ensures that our belief about the robot states is continually refined as more observations come in.

Image Loss Refinement and Velocity Estimation

While image moments have historically proven useful in object tracking [13, 142], their efficacy diminishes in the complex arena of robot state estimation. This is because they encapsulate only limited details of the robot mask. Consequently, a direct method that compares the estimated and reference robot masks provides an enhancement to state estimation accuracy.

We predict the robot mask from estimated robot states using the same differentiable rendering pipeline as described in Section 18. To measure the difference between this prediction and the reference mask, we employ an image loss function, which sums the squared differences between the predicted mask $\mathbb{M}^{pred}$ and the reference mask $\mathbb{M}^{ref}$ across the image dimensions:

$$\mathcal{L} = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \left(\mathbb{M}^{pred}(i,j) - \mathbb{M}^{ref}(i,j) \right)^2 . \tag{3.23}$$

We refine the mean of the robot states by applying back-propagation on this image loss (line 17 in Algorithm 3), bringing the estimation closer to the true state.

As a final step, in service of the next belief propagation timestep, we derive the velocity from the updated position:

$$\mathbf{v}_t = \frac{\mathbf{b}_{t|t} - \mathbf{b}_{t-1|t-1}}{\Delta t} \tag{3.24}$$

This velocity is used for the motion model in forthcoming iterations, as it feeds into predictions for the robot's future states.

3.2.3 Experiments and Results

To comprehensively assess the efficacy of our proposed state estimation algorithm, we collected datasets of a snake robot operating in both structured and unstructured environments. These datasets facilitated both qualitative and quantitative evaluations of the state estimation method.

The snake robot hardware is described in [117, 105] and is the evolutionary precursor to the NASA Extant Exobiology Life Surveyor (EELS) robot [12] that is anticipated to serve a science research vehicle for both earth science missions as well as extraterrestrial planetary exploration on Saturn's moon, Enceladus, or Jupiter's moon, Europa.

Snake-Lab Dataset: We introduced the Snake-Lab Dataset for evaluating the accuracy of the joint angle estimation and robot pose estimation. This dataset was acquired in a lab setting using an Intel® Realsense™ camera at a resolution of (1280, 720). The robot's joint angles were recorded using electromagnetic sensors and were synchronized with the captured images. Additionally, the robot's spatial position was determined using the depth capabilities of the camera. For evaluation metrics, we employed the Euclidean distance for position estimation and the L_1 norm for joint angle estimation.

Snake-Outdoor Dataset: To examine the robustness of our algorithm in less structured environments, we collected the Snake-Outdoor dataset. This dataset comprises three videos: the first two were recorded using a hand-held camera at a resolution of (1280, 720), while the third was captured via a drone camera, which has no direct connection to the snake robot system. Given the absence of ground truth for the robot's state in this setting, we adopted the Intersection-over-Union metric (IoU):

$$\text{IoU} = \frac{|\mathbb{M}^{ref} \cap \mathbb{M}^{pred}|}{|\mathbb{M}^{ref} \cup \mathbb{M}^{pred}|} \tag{3.25}$$

to compare the ground-truth robot mask $\mathbb{M}^{ref}$ with our algorithm's estimated mask $\mathbb{M}^{pred}$.

Implementation Details

To train DeepLabV3+, we collected around 1500 images, captured at a resolution of (1280, 720) and the ground truth segmentation masks were generated using Segment Anything Model [63]. We used the Adam optimizer [62] for gradient descent with 20 epochs and 8 batch size. The initial learning rate was set to 0.0001 and was decayed by a factor of 0.1 at the 10th epoch.

During the online estimation, we resize the raw image to a resolution of (640, 360). Both the observed robot mask and the rendered robot mask are processed at this resolution. For the refinement step, we set the learning rate to 0.005 and also used the Adam optimizer for gradient descent. All computational experiments were executed on a system equipped with an Intel® Core™ i9-11900F Processor and NVIDIA GeForce RTX 4090. To strike a balance between accuracy and processing speed, we perform 10 refinement iterations for each incoming image, ensuring optimal performance while sustaining an estimation speed of 1 FPS.

Experiment on Snake-Lab dataset

We present the qualitative results on the Snake-Lab dataset in Fig. 3.11, and the quantitative evaluation of our state estimation algorithm is presented in Table 3.3. We also plot the estimated joint trajectory with sensor readings in Fig. 3.10.

The results are segmented based on different scenarios: static conditions, moving

Table 3.3. Average Position and State Estimation Error on Snake-Lab Dataset

	Position error (m)	Joint state error (rad)
Static	0.0278	0.0605
Moving camera	0.0647	0.0849
Moving robot	0.0587	0.1352
Overall	0.0540	0.1125

camera, and moving robot. Under static conditions, where both the camera and the robot remain stationary, both the joint angle error and position error are the lowest, indicating that the algorithm performs exceptionally well in stable environments. Moving the camera or robot slightly affects the algorithm's accuracy. This could be attributed to the dynamic nature of the camera and the robot's movements, which might introduce complexities in state estimation. The overall average position error and joint angle error across all scenarios are 0.0540 m and 0.1125 rad, respectively. These results affirm the robustness of our state estimation algorithm, even in varying conditions. However, it's evident that dynamic factors, such as camera or robot movement, introduce some challenges, leading to increased errors.

Experiment on Snake-Outdoor dataset

Table 3.4 presents the quantitative evaluation of our state estimation algorithm on the Snake-Outdoor dataset. The results are organized based on the number of refinement steps taken, which are 1, 5, and 10. The performance metric used is the Intersection-over-Union (IoU) for each video, and the speed of the algorithm in frames per second (FPS) is also provided. From the Table 3.4, we can see a clear trade-off between accuracy and speed. As the number of refinement steps increases, there is a noticeable improvement in the Mean IoU, but the speed decreases. With 10 refinement steps, the algorithm operates at 1

Table 3.4. Quantitative evaluation on Snake-Outdoor dataset. We compute the IoU between the estimated robot mask and the ground-truth robot mask. We also report the processing speed under different settings.

	Number of refinement steps		
	1	5	10
Video 1 (Mean IoU)	0.4659	0.7632	0.8665
Video 2 (Mean IoU)	0.2456	0.3584	0.7690
Video 3 (Mean IoU)	0.3088	0.4394	0.8210
Speed (FPS)	3.5	1.5	1

FPS, which might be a limiting factor for real-time applications. However, the significant boost in accuracy might justify this trade-off in scenarios where precision is critical.

We also present qualitative results in Fig 3.12, showing the estimated skeleton and the predicted robot mask overlaid on the images. We can observe the estimated skeleton aligns with the robot's actual structure, providing a clear and intuitive understanding of the algorithm's performance in real-world, outdoor settings.

3.2.4 Conclusion

In this work, we present a novel method for state estimation of snake robots using a single camera. The proposed approach combines differentiable rendering with the Kalman filter, fusing temporal information with a rendering-based optimization technique to improve the estimation process, which enhances the method's adaptability in outdoor scenarios. The results demonstrate the efficacy of our approach on a snake robot, validating its performance in both structured and unstructured environments. We believe this technique opens up possibilities for expanded capabilities for ambulatory mobile robot deployment and navigation in complex environments, making it a promising solution for future mobile robot applications.

For future works, an exciting avenue is the exploration of how our method can be adapted for collaborative robotics, where multiple robots work in tandem. This could involve state estimation in scenarios where robots share sensory data to navigate or perform tasks (e.g. drone-assisted routing in different landscapes).

3.3 Acknowledgements

Chapter 3, in part, is a reprint of the material from J. Lu, F. Liu, C. Girerd and M. C. Yip, "Image-based Pose Estimation and Shape Reconstruction for Robot Manipulators and Soft, Continuum Robots via Differentiable Rendering," in *IEEE Conference on Robotics and Automation*, 2023. The dissertation author was one of the primary authors of this paper. Chapter 3, in part, is also a reprint of the material from J. Lu, F. Richter, S. Lin and M. C. Yip, “Tracking Snake-like Robots in the Wild using only a Single Camera,” in *IEEE Conference on Robotics and Automation*, 2024. The dissertation author was the primary author of this paper.

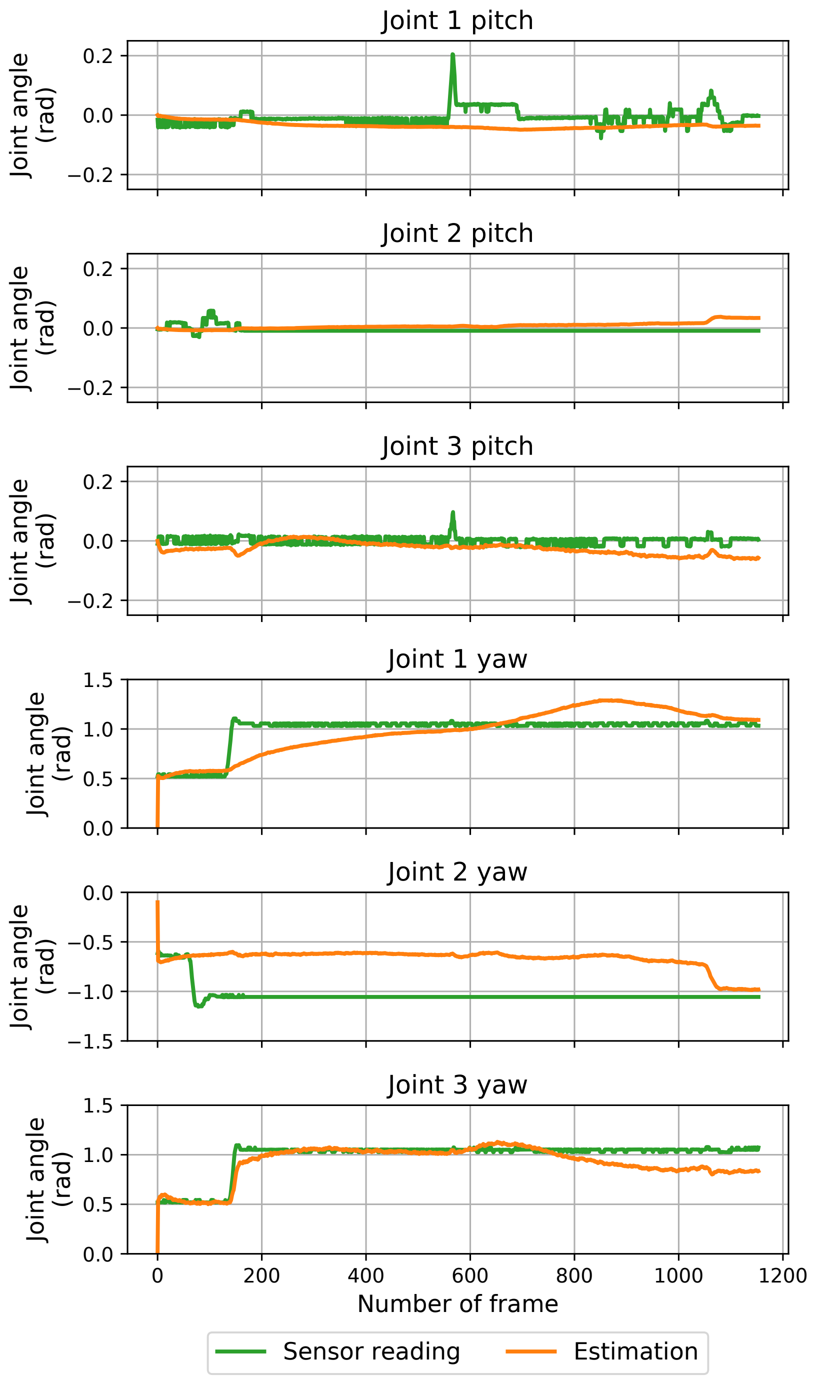

Figure 3.10. Plots of estimated joint trajectory vs. sensor reading for the Snake-Lab dataset in the moving robot scenario. For each joint, we plot the pitch and yaw angle separately. Note that the snake robot uses magnetic encoders for sensor readings and are slightly noisy due misalignment between the encoder and magnet from vibrations during the experiment.

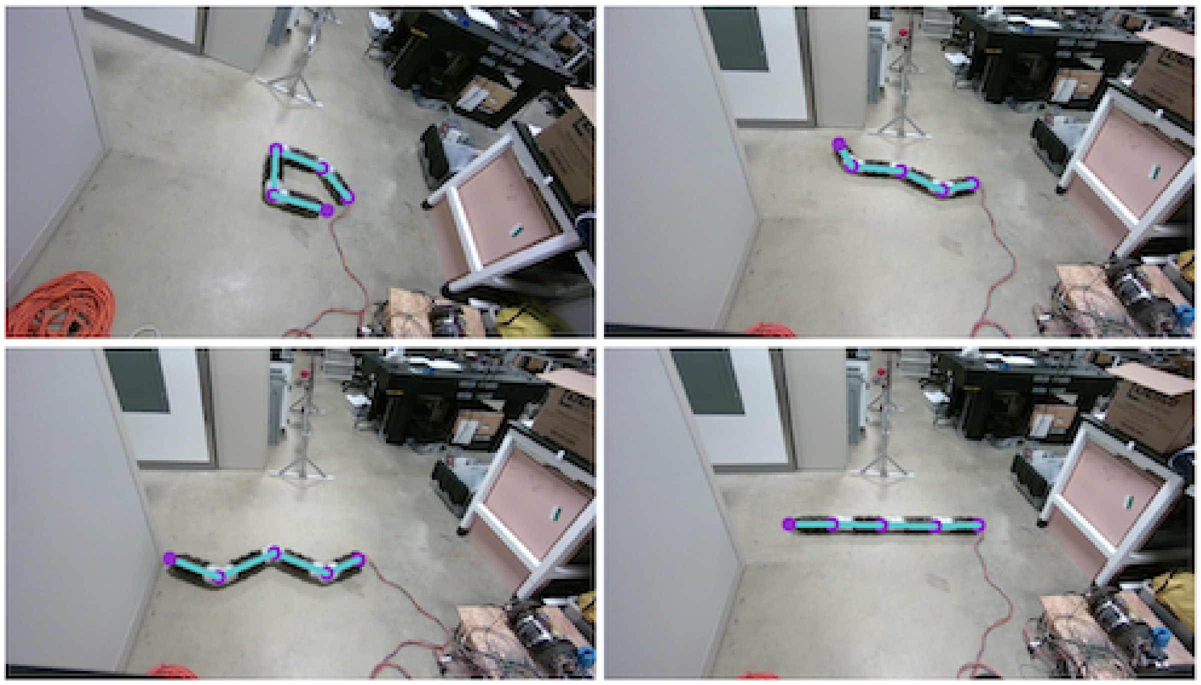

Figure 3.11. Qualitative results on Snake-Lab dataset. We derive the skeleton from the estimated robot pose and joint angle, and visualize it by projecting the skeleton on images.

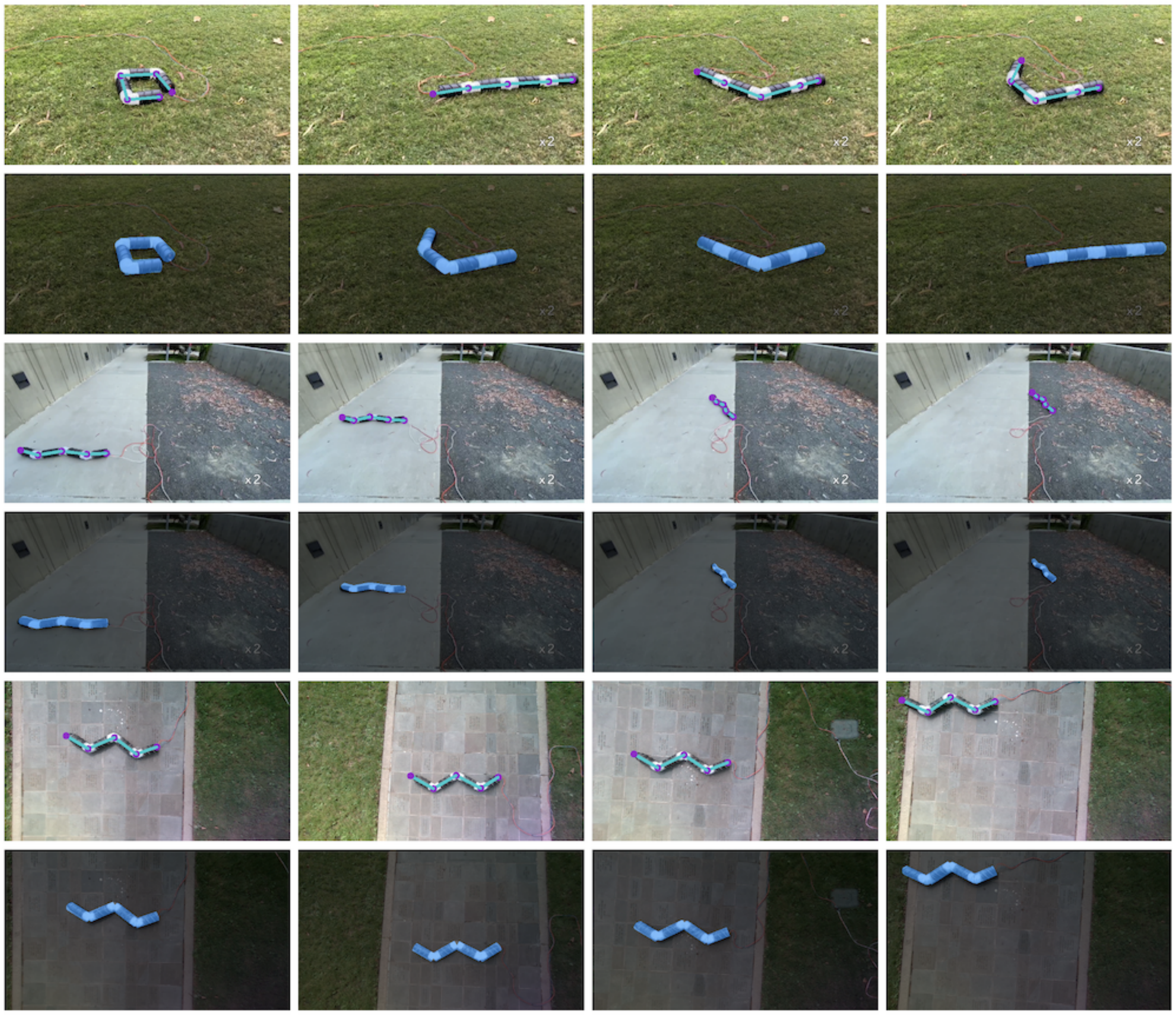

Figure 3.12. Qualitative results on Snake-Outdoor dataset. We show the estimated skeleton and predicted robot mask overlaid on images. Rows 1-2 correspond to video 1, rows 3-4 correspond to video 2, and rows 5-6 correspond to video 3. Notably, there's a precise alignment of the skeleton and mask with the robot as shown in the images.

Chapter 4

Combining Keypoint and Rendering for Pose Estimation

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) [17]. At a high level, PBVS converts 3D environmental information inferred from the visual data (e.g. 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 [82] to tissue manipulation in surgery [73].

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 [39], AprilTag [90]). 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 [70, 38]. 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

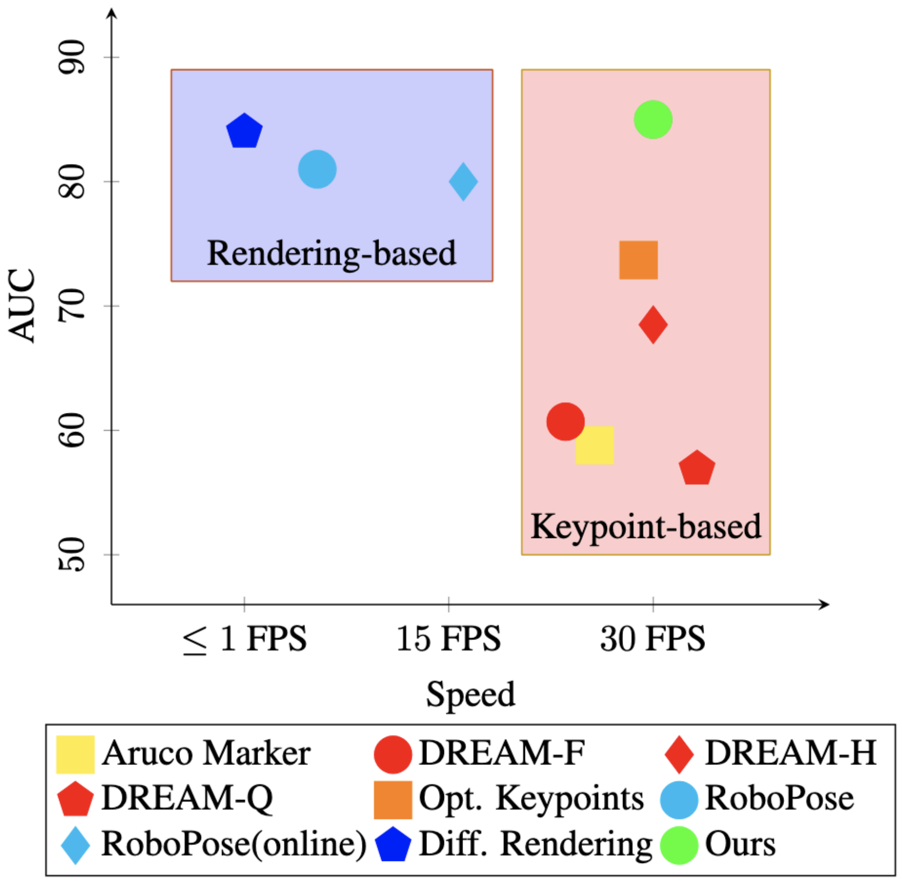

Figure 4.1. Comparison of speed and accuracy (based on AUC metric) for existing image-based robot pose estimation methods.

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 [69, 79, 106, 68, 67] and rendering-based methods [66, 42]. 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 [149].

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 [74]. 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 [59, 75]. 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 4.1, 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.

4.1 Related Works

4.1.1 Camera-to-Robot Pose Estimation

The classical way to calibrate the camera-to-robot pose is to attach the fiducial markers [39, 90] 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 [93, 34, 52, 48].

Early works on markerless articulated pose tracking utilize a depth camera for 3D observation [116, 96, 85, 33]. For a high degree-of-freedom articulated robot, Bohg et al. [6] 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 [136] 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 (eg. PnP solver) [68, 69, 79, 153]. To further improve the performance, the segmentation mask and edges are utilized to refine the robot pose [67, 42]. Labbé et al. [66] 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.

4.1.2 Domain Adaptation for Sim-to-Real Transfer

In computer vision and robotics, Domain Randomization (DR) [128] 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 [129, 7, 45, 127, 49], and pose estimation [122, 83, 130, 68, 69, 79, 153, 66]. 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 [134]. This technique has shown recent success in computer vision [46, 23, 115] and robotic applications [8, 55, 41]. 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.

4.2 Methodology

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 4.2.1 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 4.2.2 which utilizes a keypoint detector and a PnP solver to estimate the pose of the robot from image data in real-time.

4.2.1 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:

$$\theta_{bb}, \theta_{kp}, \theta_{seg} = argmin_{\theta_{bb},\theta_{kp},\theta_{seg}} \mathcal{L}[f_{seg}(\mathbb{I}|\theta_{bb}, \theta_{seg}), \mathcal{R}(f_{pose}(\mathbb{I}|\mathbf{q}, \theta_{bb}, \theta_{kp})|\mathbf{K})] \tag{4.1}$$

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 (4.1) 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.

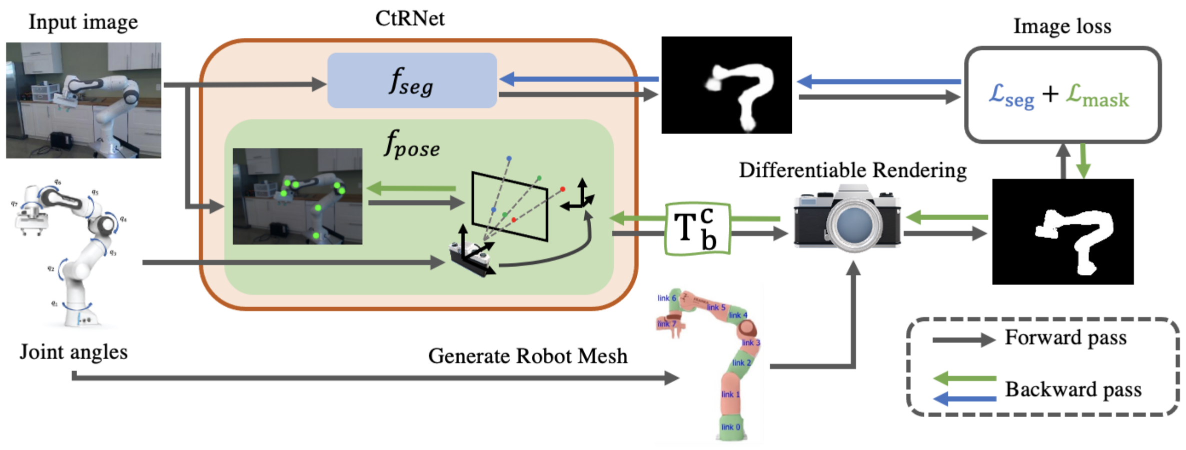

Figure 4.2. The overview of our proposed self-supervised training framework for sim-to-real transfer. The CtRNet contains a foreground segmentation module and a pose estimation module, which output a robot mask and a camera-to-robot pose respectively. The output pose is transformed into a silhouette image through a differentiable renderer. The image loss is back-propagated to train the keypoint detector and fine-tune the segmentation.

Overview. The overview of the self-supervised training pipeline is shown in the 4.2. 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 4.3. 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

$$\mathbb{M} = f_{seg}(\mathbb{I}|\theta_{bb}, \theta_{seg}) \qquad \mathbf{T}_b^c = f_{pose}(\mathbb{I}|\mathbf{q}, \theta_{bb}, \theta_{kp}) \tag{4.2}$$

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 (4.1) 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 [101]. 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 [101] 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 *SoftSilhouetteShader* is applied to compute pixel values of the rendered image using the sigmoid blending method [75].

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

$$\overline{\mathbf{v}}^c = \mathbf{T}_b^c \mathbf{T}_n^b(\mathbf{q}) \overline{\mathbf{v}}^n \tag{4.3}$$

where $\bar{\cdot}$ 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 [31].

Objective loss function. The objective loss in (4.1) 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 [66]. Mathematically the loss is expressed as

$$\mathcal{L}_{mask} = \sum_{i=1}^{H} \sum_{j=1}^{W} \left(\mathbb{S}(i,j) - \mathbb{M}(i,j)\right)^2 \tag{4.4}$$

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:

$$\mathcal{L}_{seg} = -\frac{w}{H * W} \sum_{i=1}^{H} \sum_{j=1}^{W} [\mathbb{M}(i,j) \log \mathbb{S}(i,j) + (1 - \mathbb{M}(i,j)) \log(1 - \mathbb{S}(i,j))]. \tag{4.5}$$

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:

$$w = exp\left(-sO(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c)\right) \tag{4.6}$$

where s is a scaling constant, O is the reprojection loss in the PnP solver (explained in 4.2.2), $\{\mathbf{o}_i | \mathbf{o}_i \in \mathbb{R}^2\}_{i=1}^n$ 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.

4.2.2 Camera-to-Robot Pose Estimation Network

The overview of the proposed Camera-to-Robot Pose Estimation Network, CtRNet, is shown in Fig. 4.3. Given an input RGB image, we employ ResNet50 [43] as the backbone network to extract the latent features. The latent features are then passed

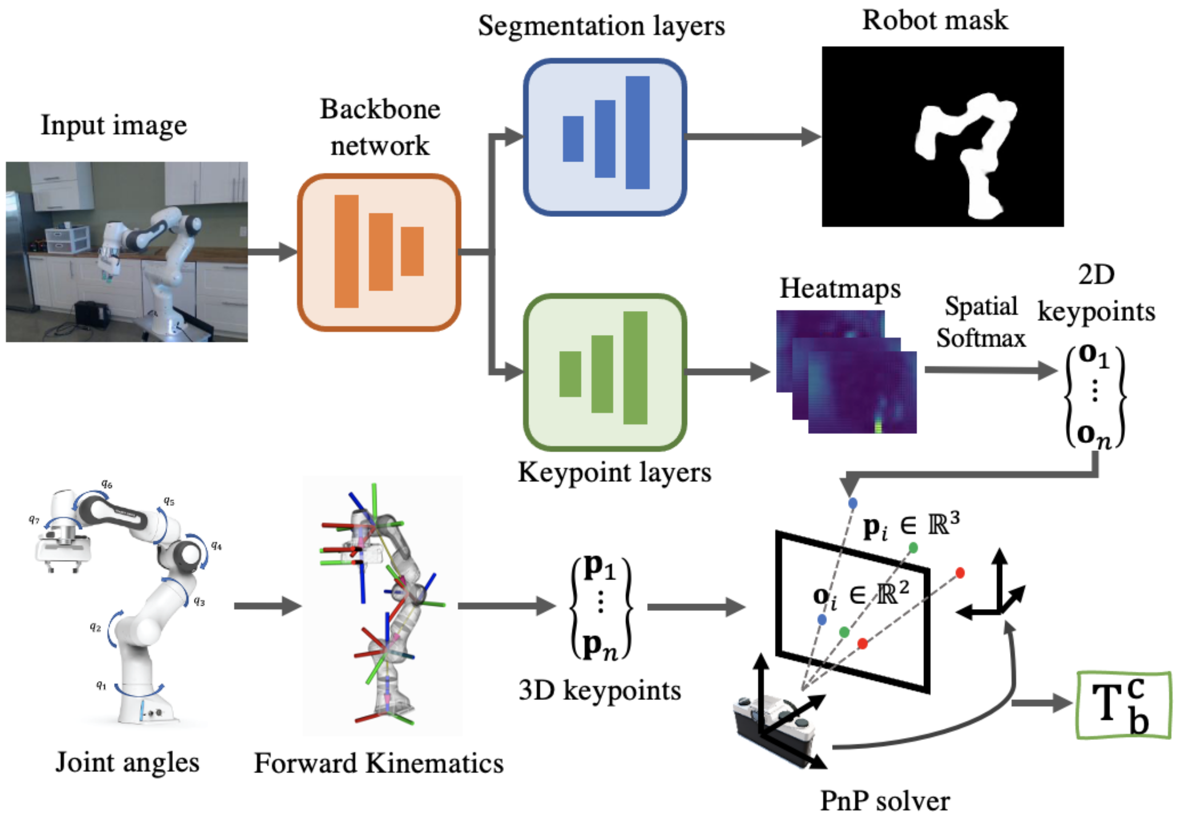
Segmentation layers
Robot mask
Input image
Backbone network
Heatmaps
Spatial Softmax
2D keypoints
$\{\mathbf{o}_1 \vdots \mathbf{o}_n\}$
Keypoint layers
$\mathbf{p}_i \in \mathbb{R}^3$
$\mathbf{o}_i \in \mathbb{R}^2$
$\{\mathbf{p}_1 \vdots \mathbf{p}_n\}$
3D keypoints
$\mathbf{T}^{\mathbf{c}}_{\mathbf{b}}$
Joint angles
Forward Kinematics
PnP solver

Figure 4.3. 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.

through the Atrous Spatial Pyramid Pooling layers [20] 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 [36] 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:

$$\overline{\mathbf{p}}_i = \mathbf{T}_i^b(\mathbf{q})\overline{\mathbf{t}},\ for\, i = 1, ..., n \tag{4.7}$$

where $\mathbf{t} = [0, 0, 0]$. With the 2D and 3D corresponding keypoints, we can then apply a PnP solver [70] 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 [18], the implicit function theorem [64] 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:

$$\mathbf{T}_b^{c*} = g(\mathbf{o}, \mathbf{p}, \mathbf{K}) \tag{4.8}$$

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:

$$\mathbf{T}_b^{c*} = argmin_{\mathbf{T}_b^c} O(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c) \tag{4.9}$$

with

$$O(\mathbf{o}_i, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c) = \sum_{i=1}^{n} ||\mathbf{o}_i - \pi(\mathbf{p}_i|\mathbf{T}_b^c, \mathbf{K})||_2^2 \tag{4.10}$$

$$= \sum_{i=1}^{n} ||\mathbf{r}_i||_2^2 \tag{4.11}$$

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 $\mathbf{T}_b^c$:

$$\frac{\partial O}{\partial \mathbf{T}_b^c}(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c)|_{\mathbf{T}_b^c=\mathbf{T}_b^{c*}} = \mathbf{0}. \tag{4.12}$$

Following [18], we construct a constrain function F to employ the implicit function theorem:

$$F(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c) = \frac{\partial O}{\partial \mathbf{T}_b^c}(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^{c*}) = \mathbf{0}. \tag{4.13}$$

Substituting the 4.10 and 4.11 to 4.13, we can derive the constraint function as:

$$\begin{aligned} F(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c) &= \sum_{i=1}^{n} \frac{\partial ||\mathbf{r}_i||_2^2}{\partial \mathbf{T}_b^c} && (4.14) \\ &= -2 \sum_{i=1}^{n} \mathbf{r}_i^T \frac{\partial \pi}{\partial \mathbf{T}_b^c}(\mathbf{p}_i | \mathbf{T}_b^{c*}, \mathbf{K}). && (4.15) \end{aligned}$$

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:

$$\frac{\partial g}{\partial \mathbf{o}}(\mathbf{o}, \mathbf{p}, \mathbf{K}) = -\left(\frac{\partial F}{\partial \mathbf{T}_b^c}(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c)\right)^{-1} \left(\frac{\partial F}{\partial \mathbf{o}}(\mathbf{o}, \mathbf{p}, \mathbf{K}, \mathbf{T}_b^c)\right). \tag{4.16}$$

4.3 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 pretraining procedure and explore how the number of pretraining 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.

4.3.1 Datasets and Evaluation Metrics

DREAM-real Dataset. The DREAM-real dataset [69] 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 [139],

$$ADD = \frac{1}{n}\sum_{i=1}^{n} ||\widetilde{\mathbf{T}}_b^c \overline{\mathbf{p}}_i - \mathbf{T}_b^c \overline{\mathbf{p}}_i||_2 \tag{4.17}$$

where $\widetilde{\mathbf{T}}_b^c$ indicates the ground-truth camera-to-robot pose. We also report the area-under-the-curve (AUC) value, which integrates the percentage of ADD over different thresholds. A higher AUC value indicates more predictions with less error.

Baxter Dataset. The Baxter dataset [79] 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 [79].

4.3.2 Implementation details

The entire pipeline is implemented in PyTorch [95]. We initialize the backbone network with ImageNet [32] 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

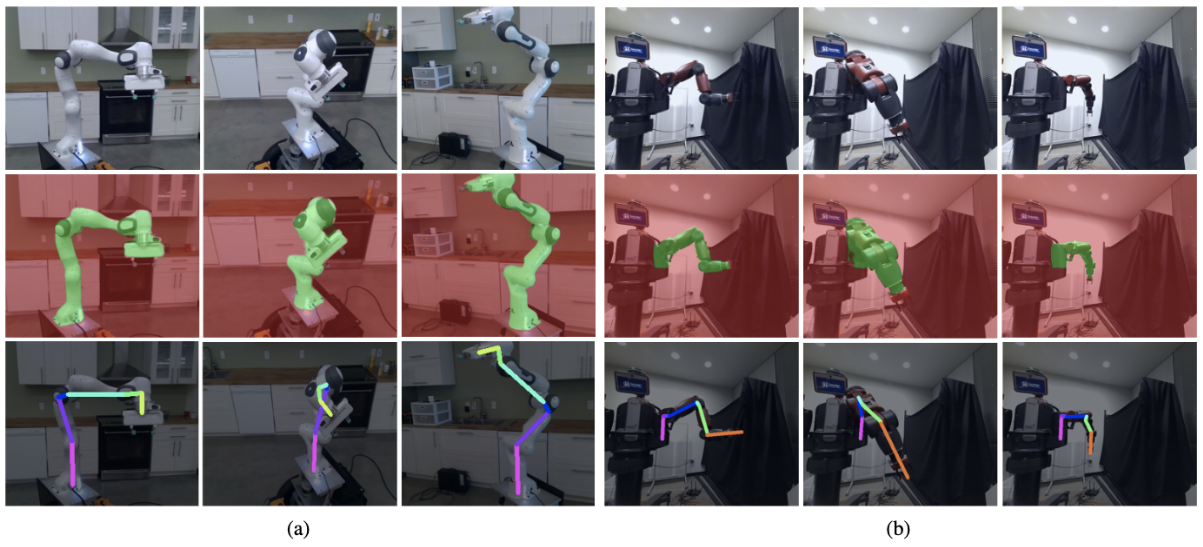

Figure 4.4. Qualitative results of CtRNet foreground segmentation and pose estimation on (a) DREAM-real dataset and (b) Baxter dataset. The first row shows the input RGB image, the second row shows the foreground segmentation, and the third row shows the projected robot skeleton based on the estimated robot pose.

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 4.6 is set to 0.1 for DREAM-real dataset and 0.01 for Baxter dataset, mainly accounting for the difference in resolution.

4.3.3 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 Figure 4.4. We compared our method with the state-of-the-art keypoint-based method DREAM [69] and the rendering-based method RoboPose [66]. The results for DREAM and RoboPose are compiled from the implementation provided by [66]. In Table

4.1, 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 Figure 4.4 shows some of the qualitative results. We compared our method with several keypoint-based methods (Aruco Marker [39], DREAM [69], Optimized Keypoints [79]). We also implemented Differentiable Rendering for robot pose estimation, where the robot masks are generated with the pretrained foreground segmentation. The 2D PCK results and 3D ADD results are reported in Table 4.2. 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.

Table 4.1. 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	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 [69]	Keypoint	VGG19	68.912	11.413	24.359	491.911	76.130	2.077	61.930	95.319	60.740	113.029
DREAM-Q [69]	Keypoint	VGG19	52.382	78.089	37.471	54.178	77.984	0.027	57.087	67.248	56.988	59.284
DREAM-H [69]	Keypoint	ResNet101	60.520	0.056	64.005	7.382	78.825	0.024	69.054	25.685	68.584	17.477
RoboPose [66]	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 4.2. Comparison of our methods with the state-of-the-art methods on Baxter dataset.

Method	Category	PCK (2D)		Mean 2D Err.	ADD (3D)		Mean 3D Err.
		@50 pixel $\uparrow$	AUC $\uparrow$	(pixel) $\downarrow$	@100 mm $\uparrow$	AUC $\uparrow$	(mm) $\downarrow$
Aruco Marker [39]	Keypoint	0.49	57.15	286.98	0.30	43.45	2447.34
DREAM-Q [69]	Keypoint	0.33	44.01	1277.33	0.32	40.63	386.17
Opt. Keypoints [79]	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 4.3. Ablation study for the number of pretraining samples.

$N_{pretrain}$	Mean ADD (mm) $\downarrow$	AUC ADD $\uparrow$
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

4.3.4 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. Figure 4.5 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 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 Table 4.3. 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.

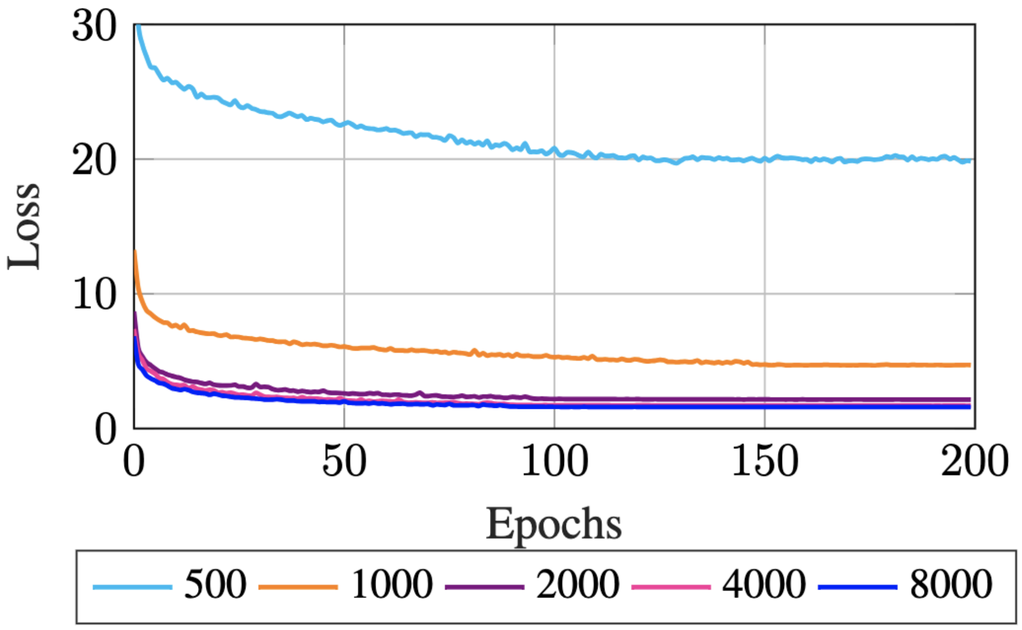

Figure 4.5. The training loss vs. number of epochs for the self-supervised training with different numbers of pretraining samples. More pretraining samples results in better convergence. The improvement saturates after having more than 2000 pretraining samples as only marginal improvement by adding more samples.

4.3.5 Visual Servoing Experiment

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

Table 4.4. Mean and standard deviation of the translational error and rotational error for the visual servoing experiment.

Method	Loop Rate	Trans. Err. (m)	Rot. Err. (rad)
DREAM [69]	30Hz	0.235 ± 0.313	0.300 ± 0.544
Diff. Rendering	1Hz	0.046 ± 0.062	0.036 ± 0.066
CtRNet	30Hz	**0.002** ± **0.001**	**0.002** ± **0.001**

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 [69] and a Differentiable Renderer for PBVS. For DREAM, the pretrained 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 Figure 4.6. 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 Table 4.4. 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 Figure 4.7. 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. 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 ($\leq$ 5 seconds), thanks to the fast and robust robot pose estimation. We show more qualitative results in the Supplementary Materials.

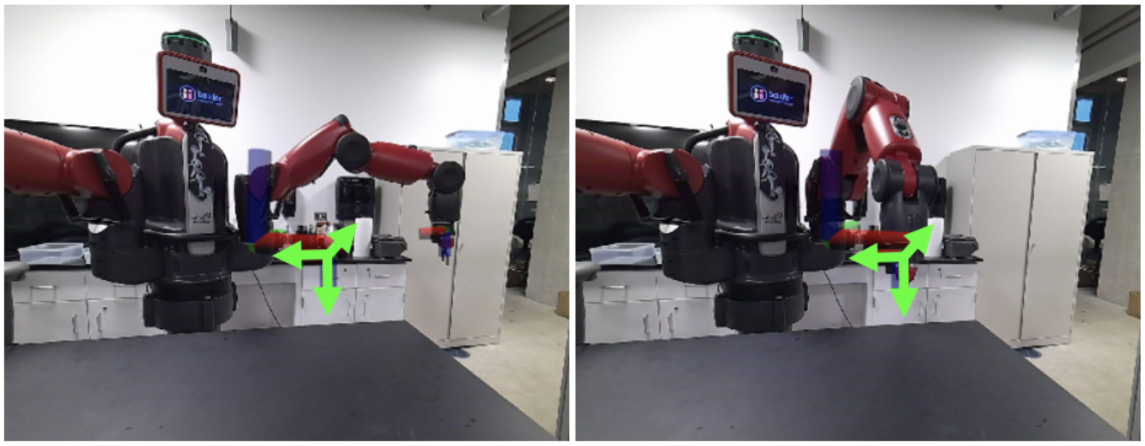

Figure 4.6. Snapshots of PBVS. The goal is to move the end-effector to the target pose (green). The figure on the right shows the robot configuration upon the convergence of PBVS.

4.4 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 Figure 4.1 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 [106] and mobile robot manipulators [137]. For future work, we would like to extend our method to more robots and explore vision-based control in an unstructured environment.

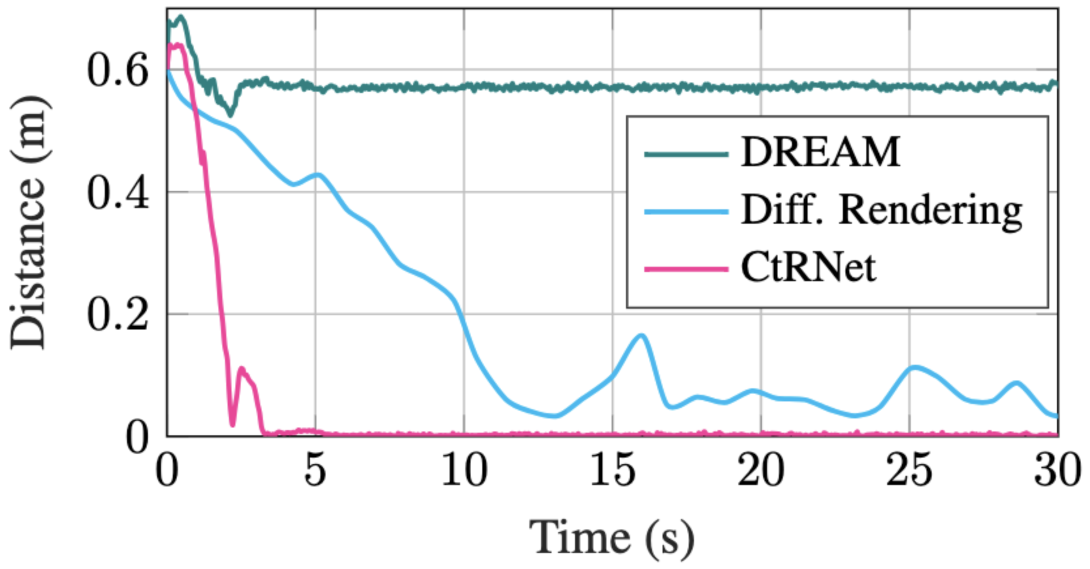

Figure 4.7. The plot of end-effector distance-to-goal over time on a selected PBVS trail.

4.5 Acknowledgments

Chapter 4, in part, is a reprint of the material from J. Lu, F. Richter, M. C. Yip, "Markerless Camera-to-Robot Pose Estimation via Self-supervised Sim-to-Real Transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. The dissertation author was the primary author of this paper.

Chapter 5

Robot Pose Estimation in the Wild

Estimating the Camera-to-Robot transform is crucial for manipulation, as it links the visual feedback from the camera to the space where the robot is operating, enabling accurate model-based robot arm manipulation with visual observations. Calibrating the Camera-to-Robot transform requires a significant amount of effort. Traditional calibration methods, such as [39, 90, 35], usually place fixed fiducial markers on the end-effector, collect images of several robot joint angles, and compute the transformation. These techniques have proved their advantage in generalizability and availability for different environments and robots. However, such a procedure requires modification to the robotic system, which is not always possible, such as in instances where a dataset has already been collected [91, 61]. Furthermore, the accuracy of the fiducial marker calibration approach is limited to the accuracy of the fiducial location relative to the robot.

The recent development of deep learning methods makes the markerless robot pose estimation possible, which can generally be divided into keypoint-based methods [79, 80, 69] and rendering-based methods [78, 42, 66]. Contrary to classical approaches that need fiducial markers, deep learning-based pose estimation methods don't require cumbersome physical setups for calibration. Instead, they utilize deep neural networks for feature extraction or segmentation. The robot pose is then estimated using the keypoint features or the segmented robot masks.

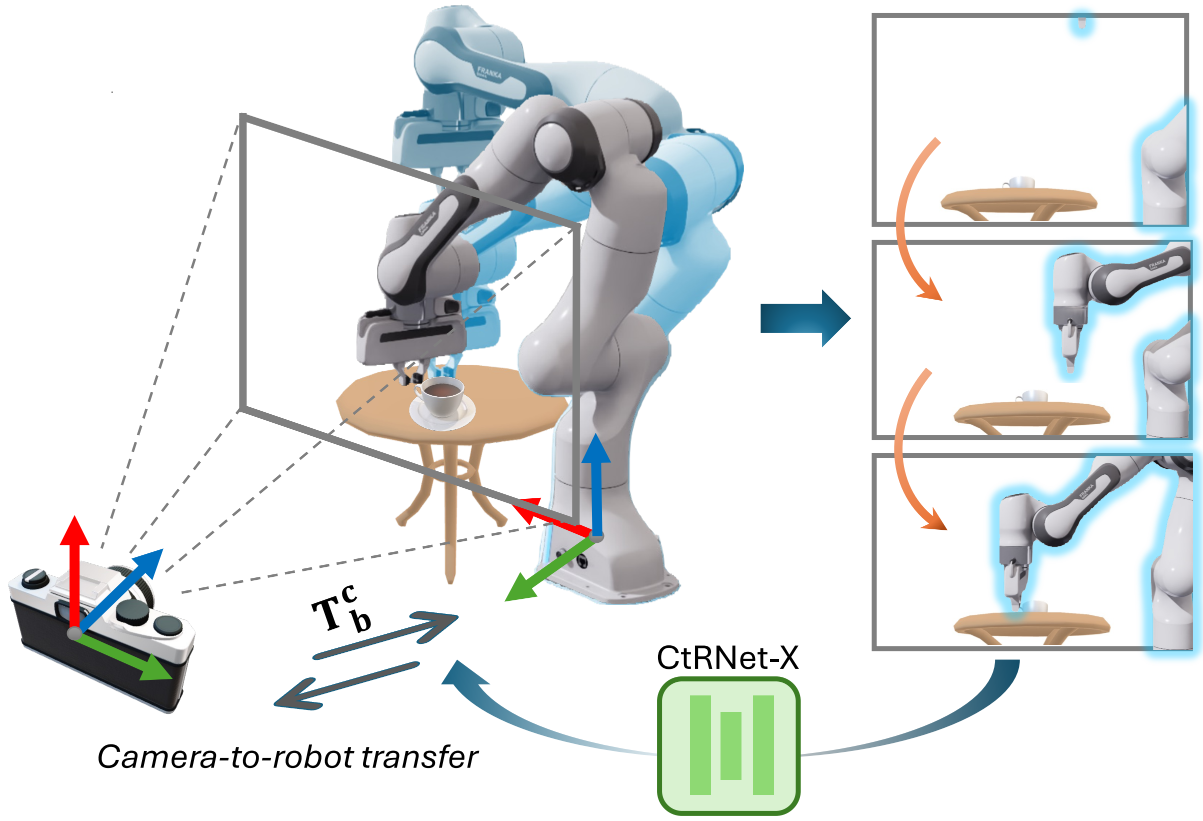

Figure 5.1. In real-world robot manipulation scenarios, the camera does not always capture all the robot links, and the visibility of robot links changes from time to time. Our method leverages the limited available visual features within the camera view and achieves state-of-the-art performance on robot pose estimation.

Despite considerable efforts to improve camera-to-robot pose estimation's flexibility, existing works have made a strong assumption that the entire robot arm is fully visible from the camera view. However, in real-world manipulation scenarios, the operating space is often limited, thus setting spatial constraints on camera placement [71, 106, 91, 61]. Additionally, the shape and size of target objects further complicate the trade-off between the capturing robot body and the manipulation targets. In such scenarios, the camera placement is typically driven more by the demands of the manipulation tasks instead of the need to observe all robot joints motion, as shown in Fig. 5.1. Consequently, video sequences with partially visible robot arm make up the majority of robotics manipulation datasets, such as Open X-Embodiment [91] and DROID [61], as shown in Fig. 5.2. Existing methods often fail in situations where robots are only partially visible due to these real world constraints. Therefore, being able to estimate the robot pose with partial views is important from the practical aspect of robot manipulation scenarios where the camera can only capture a portion of the robot.

In this work, we introduce a novel framework for Camera-to-Robot Pose Estimation that extends the markerless robot pose estimation to partially visible scenarios. Our method integrates the Vision-Language Model (VLM) to detect the visible robot components and dynamically select the keypoints from visible robot links for the pose estimation. Moreover, we also improve keypoint detection performance by introducing the distribution-aware coordinate representation [146] to our previous development for the pose estimation network [80]. We evaluate our framework on both fully visible and partially visible setups and achieve state-of-the-art performance on the public robot pose dataset and our self-collected dataset. In summary, our contributions are threefold:

- We present a framework for markerless camera-to-robot pose estimation for more practical manipulation setups, where often only parts of the robot can be observed from a camera.
- We show the benefits of using the vision-language foundation model with few-shot learning for robot part detection and integrate it into the pose estimation framework.
- We show that our method achieves state-of-the-art performance compared to the existing methods on the benchmarking dataset while demonstrating the capability of estimating accurate camera extrinsic information for large-scale robot manipulation datasets.

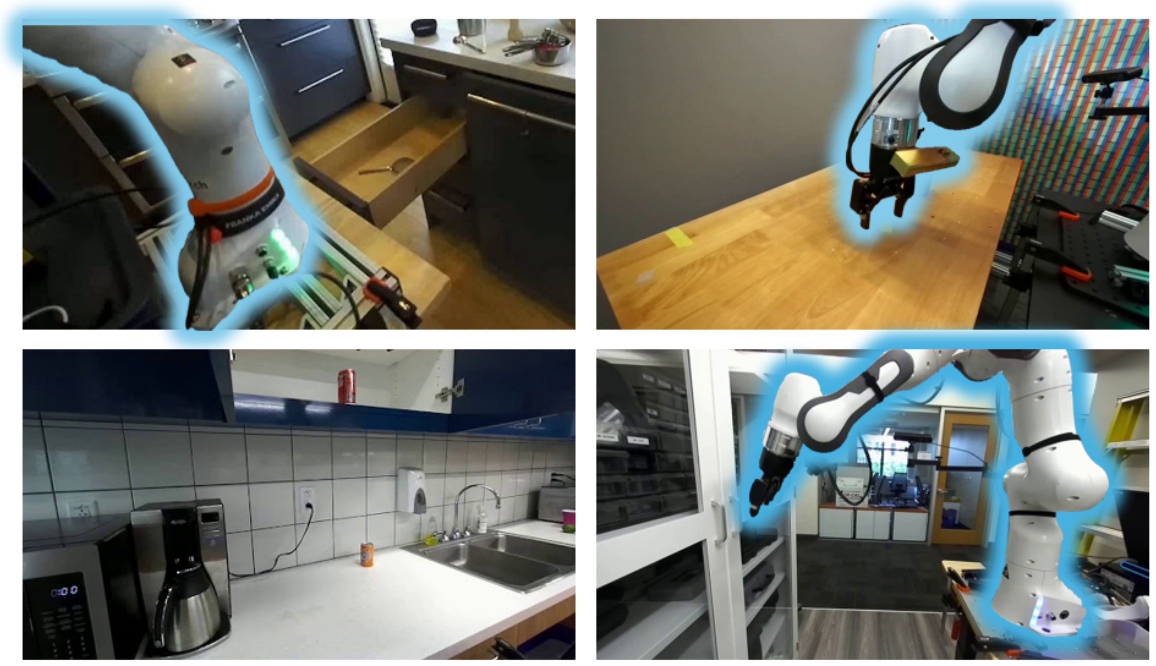

Figure 5.2. Sample images are from DROID robot learning dataset [61]. Often, only certain parts of the robot are visible in the camera view, and sometimes none of them are visible.

5.1 Related Works

Camera-to-Robot Pose Estimation. Traditionally, the camera-to-robot pose is calibrated using the fiducial markers [39, 90]. For articulated robots, the fiducial markers provide 2D point features on the robots, the 3D position of the markers can be calculated using robot kinematics, and the robot pose can be derived by solving a Perspective-n-Point problem [93, 34, 52, 48].

As the field evolved, there was a shift towards markerless pose estimation. Initial efforts in this direction utilized depth cameras to localize articulated robots [116, 96, 85, 33]. With the rise of Deep Neural Networks (DNNs), a new paradigm emerged. DNNs, with their advantages of extracting point features without the need for markers, have significantly enhanced the performance of markerless pose estimation for articulated robots [68, 69, 79, 153]. Beyond keypoint-based methods, recent works [66, 78, 22] have demonstrated the potential of rendering-based methods. Benefiting from the dense correspondence provided by robot masks, rendering-based methods achieve state-of-the-art performance on robot pose estimation, but with compromise on the processing speed due to iterative render-and-compare. Most recently, [80] proposed CtRNet, which uses robot masks to supervise the keypoint detector, achieving comparable performance to rendering-based methods while maintaining real-time inference speed. Nonetheless, existing methods focus on scenarios where the robot manipulator is fully observable. In real-world manipulations, it's non-trivial to set up the camera and the manipulator such that all the robot links stay within the camera view during the episodes, thereby diminishing the generalizability of existing methods when dealing with less constrained, real-world environments. In contrast, our proposed method overcomes this limitation by integrating a vision-language foundation model to detect the visibility of different robot parts and dynamically select the keypoints from visible robot parts for pose estimation.

Robot Part Detection. With the rapid development of deep learning, convolutional neural networks (CNNs) have demonstrated superior performance in object detection. The introduction of residual connections in ResNet [43] makes it easier to construct deeper CNN architectures, hence achieving human-level performance on image recognition. However, deep neural networks typically require very large datasets to learn, which demands substantial hardware resources as well as labeling efforts. Furthermore, data is often not available due to not only the nature of the problem or privacy concerns but also the cost of data preparation [94].

Recent developments of Vision-Language Models (VLMs), such as CLIP [100], have achieved remarkable performance on open-vocabulary object detection through training on large-scale datasets collected from the Internet. Recent research has focused on customizing the training and fine-tuning the CLIP model for specific downstream tasks. For example, CoOp [150] optimizes the tokenized prompt vectors while freezing the model, reducing training time and maintaining the model's performance. Some following works [9], [19],

[81], [143], [151] further improve prompt learning's capability. Additionally, Parameter-Efficient Fine-Tuning (PEFT) like Bitfit [145] and Clip-adapter [37] focuses on VLM model optimization while attempting to minimize the number of training parameters, balancing training time and performance. However, the existing work mainly considers individual and common object classification, whereas their performance on fine-grained object parts detection is still unexplored. In this work, we explore several methods to tackle this challenge and demonstrate an effective way of fine-grained robot parts detection with few-shot learning samples.

5.2 Methodology

In this section, we introduce our framework for camera-to-robot pose estimation. The inference pipeline of our framework is shown in Fig. 5.3. Our framework builds upon CtRNet [80], extending it to handle partially visible scenarios. We utilize the vision-language foundation model, CLIP [100], to identify the visible robot parts in the image frame hence selecting which keypoints to use for robot pose estimation. Moreover, we also incorporate the Distribution-Aware coordinate Representation of Keypoint method (DARK [146]) into our framework to further enhance the performance of keypoint detection. In Section 5.2.1, we detail our approach to fine-tune the CLIP model for robot part detection. In Section 5.2.2, we first provide an overview of CtRNet and then introduce our improvements.

5.2.1 Few-shot Learning for Robot Parts Detection

VLM Fine-tuning. Different from classifying the individual objects within images, our problem lies in detecting the components of the robot. Although vision-language foundation models demonstrate the capability of zero-shot object classification, they do not perform well in detecting robot parts. This is because the training data from the Internet lacks fine-grained semantic labels (e.g. robot end-effector, robot base). In order

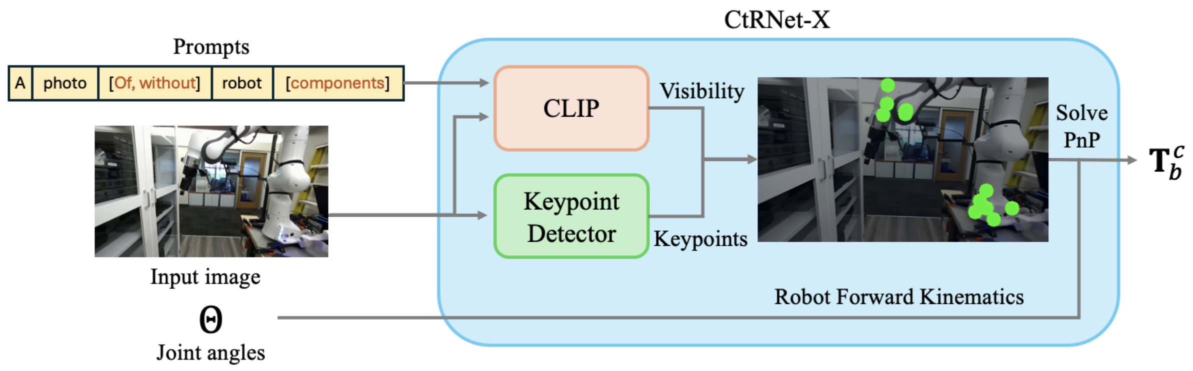

Figure 5.3. Model inference pipeline. CtRNet-X estimates camera-to-robot transform given the images and the corresponding joint angles. The framework uses a set of structured prompts and the fine-tuned CLIP model to detect which robot parts are visible and dynamically adjusts the keypoint selection. The keypoint detector outputs 2D keypoints, and the corresponding 3D keypoints are obtained from the robot forward kinematics. Finally, a PnP solver is utilized to estimate the camera-to-robot transformation matrix given the selected keypoint correspondence.

to fine-tune the VLM to detect the robot parts in the images, we collected a small number of samples from the robot learning dataset, DROID [61], and investigated the few-shot transfer capability of the popular vision-language foundation model, CLIP [100].

Training a large model like CLIP with a small dataset can be challenging. To address this, we employ the parameter-efficient fine-tuning method, Low-Rank Adaptation (LoRA [50]), and a strategic prompting method for fine-tuning the CLIP. LoRA provides inspiration on freezing the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture. We add the LoRA module on both text and image encoders of the CLIP. For a forward linear passes $h = W_0 x$, we apply LoRA such that

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{5.1}$$

where the W_0 is the pre-trained weight matrix of the self-attention module of CLIP, and ΔW is the trainable weight matrix. Specifically, $W_0 \in \mathbb{R}^{d \times k}$, $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$, where the r represents the low intrinsic dimension and $r \ll \min(d, k)$. During the training, we only optimize the weights of these low-rank matrices B and A which have significant fewer parameters, while freezing the pre-trained weight. This approach allows for fast and efficient fine-tuning.

Prompt Strategy. The prompt plays a crucial role in VLMs as a good prompt strategy can improve the performance of the model without extra effort. A typical prompt for CLIP is formulated as *A photo of* {*object*}. For classification, the {*object*} is replaced with different class labels, and the CLIP will select the class based on the cosine similarity between the image embedding and text embedding. In our scenario, the {*object*} is defined as the name of the robot components (e.g. robot base, robot end-effector). However, since the text of different robot parts is semantically similar, we have found that it is more effective to separate the queries for different components. Specifically, for each robot part, we input a pair of prompts (*A photo of robot* {*component*}, *A photo without robot* {*component*}). CLIP conducts binary classification for each component individually, thereby eliminating ambiguity when choosing from semantically close text embeddings.

To investigate the few-shot capability of VLM for robot part detection, we conducted a comparison of different few-shot learning approaches. These included parameter-efficient fine-tuning method (LoRA [50]), prompt learning method (CoOp [150]), and traditional image classification using ResNet [43] on the dataset we scraped from DROID.

5.2.2 Camera-to-Robot Pose Estimation

CtRNet Overview. The Camera-to-Robot Pose Estimation Network (CtRNet [80]) is the pioneer method for end-to-end robot pose estimation. The CtRNet includes a segmentation network, a keypoint detection network, and a differentiable Perspective-n-Point solver (BPnP [18]). During the inference time, given the image frames and corresponding joint angles, the keypoint detector predicts the 2D keypoint coordinates on the robot manipulator, and the PnP solver estimates the robot pose with 2D-3D keypoint associations. The CtRNet is pre-trained on the synthetic dataset with ground-truth labels of segmentation masks and keypoint coordinates and is fine-tuned in the real-world data without labels in a self-training manner. In the self-training phase, a differentiable renderer is utilized to compute the robot mask based on pose estimation, and the masks obtained from the segmentation network are leveraged to provide image-level supervision to optimize the keypoint detector.

Model Training. In this work, we follow the training strategy of the CtRNet with modified keypoint placement. The CtRNet defines the keypoint at the location of each robot joint. To ensure the framework has a sufficient number of keypoint to estimate the pose for each image frame with partial view, we place N ($N \geq 4$) number of keypoints for each robot link. During the pre-training phase, CtRNet uses coordinate regression for training the keypoint detector, which minimizes the L_2 distance between the ground-truth and predicted keypoint coordinates. Inspired by DARK [146], we adapt heatmap regression for training the keypoint detector. The heatmap provides spatial support around the ground-truth location, taking into account contextual clues and the inherent ambiguity of the target position. Importantly, this approach can effectively reduce the risk of overfitting in the model during training, similar to the concept of class label smoothing regularization. The heatmap regression minimizes the per-pixel L_2 distance between the predicted and ground-truth heatmap. In this work, we assume the heatmap should follow the Gaussian distribution. To supervise the heatmap prediction, the ground truth keypoint coordinates are encoded into the Gaussian heatmap, $\mathcal{D}$, as

$$\mathcal{D}(u,v) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(u-u^*)^2 + (v-v^*)^2}{2\sigma^2}\right) \tag{5.2}$$

where u, v are pixel coordinates in the heatmap, u^*, v^* are the ground truth keypoint coordinates, and σ is the predefined spatial variance. After pre-training on the synthetic dataset, we conduct self-supervised training on the real-world data without labels. The objective of self-training is to optimize the neural network parameters by minimizing the difference between the segmentation robot mask and the robot mask rendered based on the predicted pose. We follow the same self-training strategy as the CtRNet and the details can be found in [80].

Model Inferencing. During the inference phase, we integrate the distribution-aware coordinate decoding [146] to extract 2D coordinates of keypoints from the predicted heatmap. We assume the predicted heatmap follows a 2D Gaussian distribution:

$$\mathcal{G}(\mathbf{x};\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{5.3}$$

where $\mathbf{x}$ is the pixel location, $\boldsymbol{\mu}$ is the Gaussian mean and Σ is the distribution's covariance. In order to estimate $\boldsymbol{\mu}$, we follow the maximum likelihood estimation principle and transform the distribution function to the Log-likelihood function:

$$\mathcal{P}(\boldsymbol{x};\boldsymbol{\mu},\Sigma) = \ln(\mathcal{G}) = -\ln(2\pi) - \frac{1}{2}\ln(|\Sigma|) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \tag{5.4}$$

Following [146], we can approximate $\boldsymbol{\mu}$ using the maximum activation $\mathbf{m}$ of the predicted heatmap, as the maximum activation generally represents a good coarse prediction that approaches $\boldsymbol{\mu}$. Then, we can approximate $\mathcal{P}(\boldsymbol{\mu})$ using the Taylor series expansion evaluated at $\mathbf{m}$:

$$\mathcal{P}(\boldsymbol{\mu}) = \mathcal{P}(\boldsymbol{m}) + \mathcal{D}'(\boldsymbol{m})(\boldsymbol{\mu}-\boldsymbol{m}) + \frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{m})^\top \mathcal{D}''(\boldsymbol{m})(\boldsymbol{\mu}-\boldsymbol{m}) \tag{5.5}$$

Solving the above equation, we can obtain the equation to estimate $\boldsymbol{\mu}$ as

$$\boldsymbol{\mu} = \boldsymbol{m} - \left(\mathcal{D}''(\boldsymbol{m})\right)^{-1}\mathcal{D}'(\boldsymbol{m}) \tag{5.6}$$

where the $\mathcal{D}''(\boldsymbol{m})$ and $\mathcal{D}'(\boldsymbol{m})$ are the first and second image derivative of the predicted heatmap at the maximum activation $\mathbf{m}$. The detailed derivation can be found in [146].

For robot manipulation tasks, the robot and camera positions often remain fixed throughout the episode. We can leverage this temporal consistency to improve the accuracy of estimation. Instead of estimating the robot pose for each single frame, we estimate it based on a batch of image frames from a single episode. To perform batch estimation, we first use CLIP to predict which parts of the robot are visible in each image frame and select the keypoints that appear on the visible parts. Since we have more than enough keypoints to solve for the pose, we prioritize the keypoints with high confidence. We evaluate each keypoint based on the maximum activation value in the heatmap, denoted as $|\mathbf{m}|$, and filter out the keypoints with a maximum activation value below a certain threshold. Finally, we combine all the reliable keypoints from multiple frames and use the PnP solver to estimate the robot pose.

5.3 Experiments and Results

5.3.1 Implementation Details

We implemented the framework in Pytorch. The CLIP for robot parts detection is fine-tuned using an NVIDIA RTX 3090 GPU, while the pre-training and self-training of the keypoint detector are conducted on an NVIDIA RTX A6000 GPU. For few-shot learning comparisons, the network or prompt parameters are trained for 200 epochs, given the small sample size and the models converge quickly. For fine-tuning the CLIP using LoRA, we apply low-rank matrices on the query, key and value matrices for both image and text encoders with $r = 2$.

For the keypoint detector, we pre-train the neural network on synthetic data from the DREAM dataset [69] and follow the training parameters from the CtRNet [80]. The standard deviation of the Gaussian heatmap is set to 6 pixels. In the robot manipulation scenarios, the most common visible components are the robot end-effector and robot base. Hence, we simplify our framework to specifically recognize the robot end-effector and robot base. We place 6 keypoints for each of the links representing the robot end-effector and robot base (see example in Fig. 5.3). For self-training, we utilize the DREAM-real dataset and our self-collected Panda manipulation dataset.

5.3.2 Few-shot Learning for Robot Parts Detection

To examine the performance of VLM for robot parts detection, we compare popular fine-tuning methods, including Low-Rank Adaption [50], prompt learning [150], and full-fine-tuning. Moreover, we also include classical object recognition technique using ResNet [43] with ImageNet pretrained weights. For simplicity, we conduct the experiment on detecting the robot end-effector and robot base links. We train the networks on the training dataset scraped from DROID [61] and evaluate the performance on a test dataset that has various unseen environments. We experiment with 3 random seeds and report the average top-1 accuracy for each category. The results are shown in Table 5.1. We found that increasing the number of learning samples improves performance in general. However, the improvement becomes marginal when the training dataset becomes larger. Fine-tuning the CLIP with LoRA achieves better performance overall and requires less training time when learning with 32 shots.

Table 5.1. Comparsion of top-1 accuracy (%) and training time (s) of different fine-tuning methods on robot parts detection with few-shot learning. We evaluate the detection accuracy for the robot base (Base) and robot end-effector (EE). The results are based on the average of 3 seeds. Fine-tuning CLIP using LoRA achieves good performance with short training time.

Method	0 shots			4 shots			8 shots			16 shots			32 shots		
	EE (%)	Base (%)	Time (s)	EE	Base	Time	EE	Base	Time	EE	Base	Time	EE	Base	Time
ResNet50 [43]	-	-	-	62.76	62.23	**31.17**	70.56	69.43	**46.14**	79.43	79.46	**72.90**	91.13	77.23	138.91
ResNet152 [43]	-	-	-	55.00	61.10	70.23	62.76	61.66	85.16	71.66	67.23	121.90	91.66	75.56	203.34
CoOp [150]	75.00	63.33	-	**82.76**	67.76	69.70	**86.70**	73.33	87.10	87.23	72.23	138.82	93.33	68.90	159.79
CLIP (Full Fine-tuning)	75.00	63.33	-	76.66	72.20	316.06	82.23	72.76	330.69	86.13	75.00	365.27	90.00	80.00	450.81
CLIP (LoRA [50])	75.00	63.33	-	76.66	**81.13**	40.70	81.10	**80.56**	57.89	**92.76**	**80.56**	89.51	**96.70**	**87.23**	**108.33**

5.3.3 Experiment on DREAM-real Dataset

We benchmark the robot pose estimation performance of CtRNet-X on the DREAM-real [69] dataset. The DREAM-real dataset consists of real-world images of the Franka Emika Panda robot arm captured from three different camera setups with total of around 57k image frames. We evaluate our method, together with other state-of-the-art robot pose estimation methods, on a single-frame setup. We adopt average distance (ADD) metric to evaluate the pose estimation accuracy,

$$ADD = \frac{1}{n}\sum_{i=1}^{n}\left\|\tilde{\mathbf{T}}^c_b\mathbf{p}_i - \mathbf{T}^c_b\mathbf{p}_i\right\|_2 \tag{5.7}$$

where $\tilde{\mathbf{T}}^c_b$ and $\mathbf{T}^c_b$ stand for the ground truth and estimated pose respectively. The area-under-the-curve (AUC) value and mean ADD are reported in Table 5.2. Benefiting from the advanced distribution-aware coordinate decoding method, CtRNet-X achieves higher AUC and lower mean errors compared to existing methods.

5.3.4 Experiment on Panda Manipulation Dataset

The DREAM-real dataset only includes images where the robot arm is fully visible. To better evaluate our performance in real-world robot manipulation scenarios, we collected the Panda manipulation dataset. This dataset contains scenarios where only certain parts of the robot are visible during manipulation, and the robot arm is sometimes in and out of the image frame.

This dataset includes 60 video episodes: 30 robot-in-view episodes, where the visibility of the robot parts remains the same throughout the episode, and 30 robot-in-and-out episodes, where the visibility of the robot parts changes throughout the episode. Each episode comes with synchronized robot joint angles and ground-truth camera-to-robot transformation. The camera-to-robot transformation is carefully calibrated using a checkerboard. To reduce calibration errors, we verify the calibration results by projecting

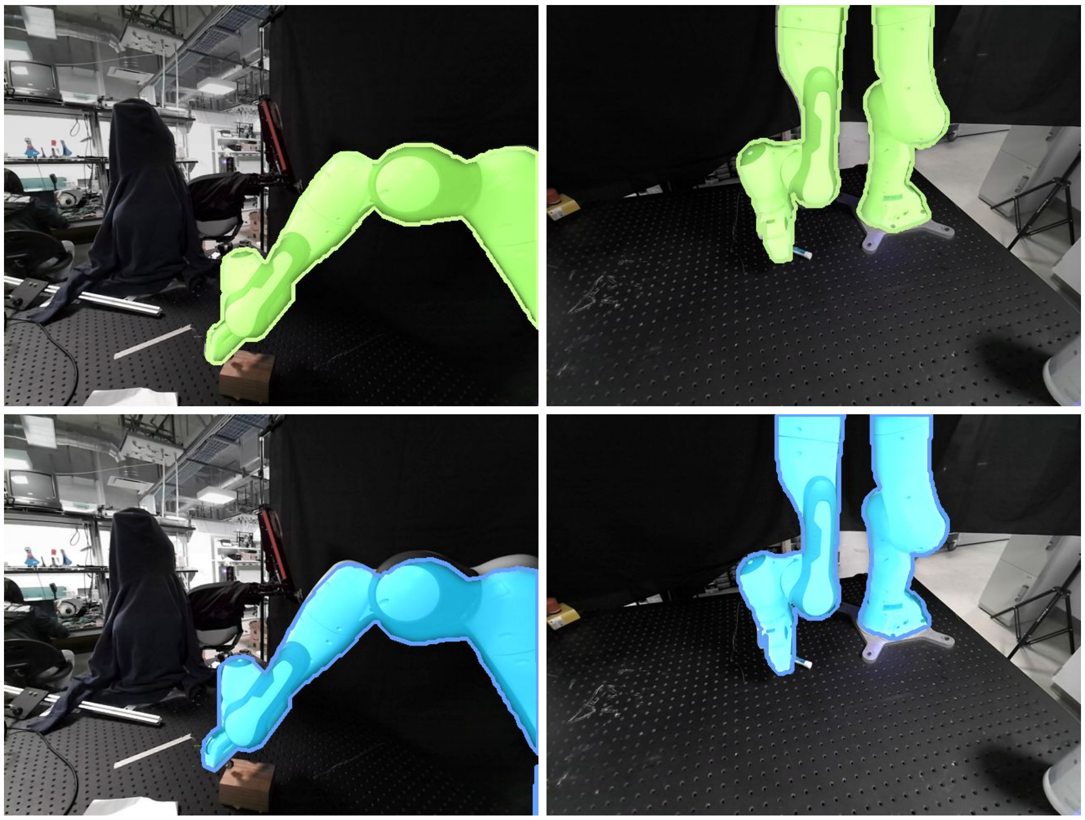

Figure 5.4. Qualitative results on Panda manipulation dataset. The first row is rendered robot masks using ground-truth extrinsic calibration (green) and the second row is the rendered robot masks using the pose from CtRNet-X (blue).

the points at each link and overlaying the robot masks to ensure alignment with the images. We compare our method with the original CtRNet [80], and report the ADD metric in Table 5.3. Additionally, we have provided the qualitative results in Fig. 5.4. Our method demonstrates significant improvements in partial-view robot pose estimation compared to the previous method. By leveraging temporal consistency through batch estimation, CtRNet-X further improves accuracy by a significant margin.

5.3.5 Experiment on DROID Dataset

In this section, we demonstrate that our method can be used to obtain accurate extrinsic calibration for large robot learning datasets. DROID [61] is a large-scale in-the-wild robot manipulation dataset. We randomly sampled video episodes from the DROID and applied CtRNet-X to obtain the camera-to-robot transformation. We show some of the qualitative results in Fig. 5.5, where we render the robot masks based on the estimated robot pose.

To quantitatively evaluate our pose estimation performance on DROID, we use Segment Anything [63] to obtain the ground-truth robot masks and compute the Intersection over Union (IoU) for the rendered robot masks. Due to labeling intensity for a hand-labeled ground-truth, we randomly selected 10 video episodes from DROID, which in total contain 3232 image frames, to label the ground-truth robot masks. The CtRNet-X achieves the average IoU of 0.8356. We noticed that using the extrinsic information provided by the dataset, the average IoU of the rendered robot mask is 0.0186, demonstrating that the extrinsic calibration is prone to having errors which highlights the necessity for accurate extrinsic calibration using our method.

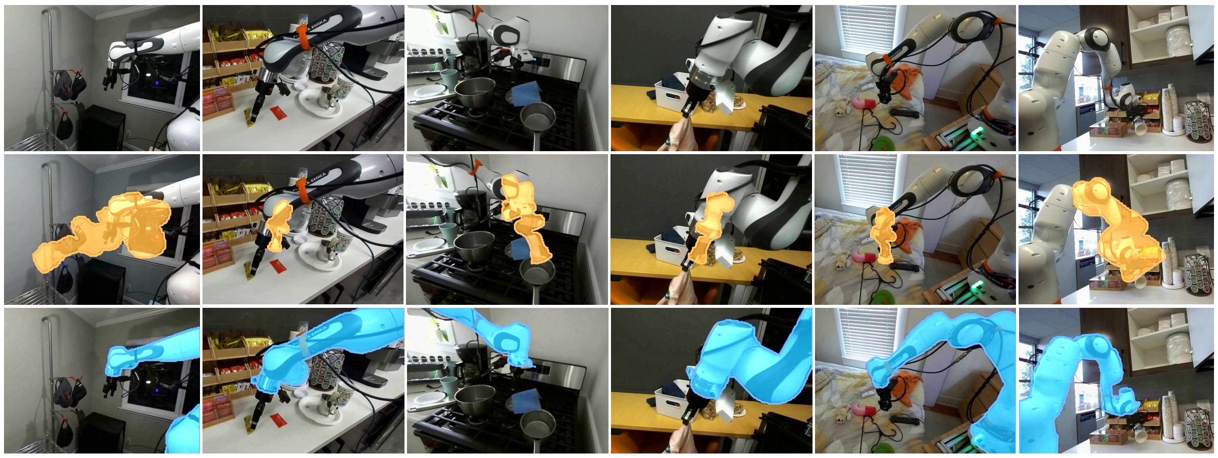
Figure 5.5. Qualitative results of our method on the real-world manipulation dataset DROID [61]. The first row is the raw image frames, the second is the robot masks rendered based on the estimation of the original CtRNet (orange), and the third row is the robot masks rendered based on the estimation of the CtRNet-X (blue). As shown above, CtRNet fails under real-world conditions whereas our method exhibits greater generalizability.

Table 5.2. Performance comparison of different methods on DREAM-real dataset. We report the overall keypoint accuracy for the mean ADD and AUC of ADD.

Method	Category	AUC ↑	Mean (m) ↓
DREAM-F [69]	Keypoint	60.740	113.029
DREAM-Q [69]	Keypoint	56.988	59.284
DREAM-H [69]	Keypoint	68.584	17.477
RoboPose [66]	Rendering	80.094	0.020
CtRNet [80]	Keypoint	85.962	0.020
CtRNet-X	Keypoint	**86.231**	**0.014**

Table 5.3. Qualitative results on Panda Manipulation Dataset. We report the overall mean and AUC of ADD with both single-frame and batch estimation.

Method	robot in view		robot in-and-out	
	AUC ↑	Mean (m) ↓	AUC ↑	Mean ↓
CtRNet (single frame)	16.764	0.381	35.944	0.335
CtRNet-X (single frame)	60.317	0.059	59.828	0.056
CtRNet-X (batch)	**70.817**	**0.038**	**79.665**	**0.022**

5.4 Conclusion

We propose CtRNet-X, an end-to-end image-based robot pose estimation framework that can generalize to real-world robot manipulation scenarios. We employ the VLM, CLIP, for robot parts detection and dynamically select the keypoints of the visible robot parts. The robot pose is then estimated using a PnP solver with selected 2D and 3D keypoint correspondence. We evaluate our method on the public robot pose dataset and self-collected manipulation dataset, demonstrating the superiority of our method in both fully and partially visible scenarios. Admittedly, the performance of the framework would be limited by the visible robot parts since we only utilize the robot end-effector and robot base links for pose estimation. However, our approach can be extended to finer granularity by including more robot parts. In the future, we will extend our method to different robots, incorporate kinematic uncertainty [106], and investigate the performance in more complex environments.

5.5 Acknowledgments

Chapter 5, in part, has been submitted for publication of the material as it may appear in J. Lu, Z. Liang, T. Xie, F. Ritcher, S. Lin, S. Liu, M. C. Yip, "CtRNet-X: Camera-to-Robot Pose Estimation in Real-world Conditions Using a Single Camera" in *IEEE Conference on Robotics and Automation*, 2025. The dissertation author was one of the primary authors of this paper.

Chapter 6

Conclusion

This dissertation has explored novel approaches for image-based robot pose estimation, with a focus on enhancing flexibility and accuracy in camera-to-robot pose estimation for robotic manipulators. The proposed methods, which bypass the need for fiducial markers, address longstanding challenges in robot pose estimation by introducing deep learning-based, markerless solutions suitable for dynamic and unstructured environments. This work contributes both theoretical advancements and practical implementations to the field of robot perception, demonstrating methods that can be applied across various robotic systems, such as robot manipulators, surgical robots, and snake-like robots.

6.1 Summary of Contributions

The first contribution of this work is the development of a keypoint-based approach to robot pose estimation, as discussed in Chapter 2. This approach leverages deep neural networks to detect robot keypoints from visual data. By optimizing the selection of keypoints and applying Domain Randomization for sim-to-real transfer, this approach demonstrates robust performance in robot pose estimation across diverse tasks, with significant improvements in accuracy and speed over traditional methods.

A second contribution is the introduction of a rendering-based method in Chapter 3, which utilizes differentiable rendering to iteratively refine pose estimates by minimizing the difference between rendered and observed data. This method addresses pose estimation challenges in scenarios with limited feature visibility. By utilizing dense image-level supervision, this method achieves state-of-the-art performance and demonstrates enhanced robustness.

By integrating the above methods with probabilistic filtering techniques such as Particle Filters and Kalman Filters, this dissertation demonstrates the capability of online pose tracking. This capability is particularly valuable for real-time applications like surgical task automation and space exploration. These results indicate the methods' adaptability and reliability in dynamic settings where precise, continuous tracking is essential.

To overcome the limitations of sim-to-real domain adaptation, Chapter 4 proposed a self-supervised training framework that enhances the generalization capability of the model. This approach integrates keypoint-based and rendering-based methods, leveraging their complementary strengths to improve robustness in real-world environments, without the need for human-labeled data.

Finally, this work extends the camera-to-robot pose estimation framework to scenarios where robots are only partially visible within the camera's field of view. By incorporating a vision-language foundation model to detect visible robot parts, this framework broadens the scope of practical applications, allowing for reliable pose estimation even in cluttered and partially occluded environments.

6.2 Future Work

While this dissertation has made substantial strides in advancing markerless robot pose estimation, several promising avenues for future work remain: 1) Extension to multi-camera and multi-robot systems: Future research could extend these approaches to scenarios involving multiple cameras and robots, where accurate, synchronized pose estimation across several viewpoints or robot agents is required. Such applications are relevant for collaborative robotics and environments where robots interact closely with humans or other robots. 2) Adaptive models for variable environments: Incorporating adaptive models that respond to changes in lighting, background, and environmental factors would increase the robustness of the pose estimation methods. Exploring meta-learning and active learning paradigms could further improve model adaptability. 3) Application to soft robotics and deformable objects: While this work primarily focuses on rigid-body manipulators, extending the techniques to soft robots or deformable objects is a promising direction. This would involve refining the keypoint-based or rendering-based approach to account for non-rigid shapes, enabling applications in fields such as medical robotics and agriculture.

The advances presented in this dissertation reflect the potential of deep learning in transforming robot pose estimation from a labor-intensive process to a more flexible and robust solution. By demonstrating the feasibility and effectiveness of markerless approaches in diverse robotic applications, this work contributes to the ongoing evolution of intelligent robotic systems. It is anticipated that the findings and methodologies developed here will serve as a foundation for further research, inspiring continued innovation in robotic perception, automation, and human-robot collaboration.

Bibliography

[1] Abdul Aabid, Bisma Parveez, Nagma Parveen, Sher Afghan Khan, JM Zayan, and O Shabbir. Reviews on design and development of unmanned aerial vehicle (drone) for different applications. *J. Mech. Eng. Res. Dev*, 45(2):53–69, 2022.

[2] Ali AlBeladi, Girish Krishnan, Mohamed-Ali Belabbas, and Seth Hutchinson. Vision-based shape reconstruction of soft continuum arms using a geometric strain parametrization. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11753–11759, 2021.

[3] Herbert Bay et al. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[4] Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414, 2019.

[5] Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

[6] 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.

[7] 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.

[8] 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.

[9] Adrian Bulat and Georgios Tzimiropoulos. Lasp: Text-to-text optimization for language-aware soft prompting of vision & language models. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 23232–23241, 2023.

[10] Simone Buoncompagni et al. Saliency-based keypoint selection for fast object detection and matching. *Pattern Recognition Letters*, 62:32–40, 2015.

[11] Paolo Cabras, Florent Nageotte, Philippe Zanne, and Christophe Doignon. An adaptive and fully automatic method for estimating the 3d position of bendable instruments using endoscopic images. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 13(4):e1812, 2017. e1812 RCS-16-0177.R2.

[12] Kalind Carpenter, Andrew Thoesen, Darwin Mick, Justin Martia, Morgan Cable, Karl Mitchell, Sarah Hovsepian, Jay Jasper, Nikola Georgiev, Rohan Thakker, Ara Kourchians, Brian Wilcox, Michael Yip, and Hamid Marvi. *Exobiology Extant Life Surveyor (EELS)*, pages 328–338. ASCE Library, 2021.

[13] François Chaumette. Image moments: a general and useful set of features for visual servoing. *IEEE Transactions on Robotics*, 20(4):713–723, 2004.

[14] Francois Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The confluence of vision and control*, pages 66–78. Springer, 2007.

[15] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[16] François Chaumette and Seth Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics & Automation Magazine*, 14(1):109–118, 2007.

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

[18] 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.

[19] Guangyi Chen, Weiran Yao, Xiangchen Song, Xinyue Li, Yongming Rao, and Kun Zhang. Plot: Prompt learning with optimal transport for vision-language models. *arXiv preprint arXiv:2210.01253*, 2022.

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

[21] Liang Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. Europ. Conf. Comput. Vis.*, pages 801–818, 2018.

[22] Linghao Chen, Yuzhe Qin, Xiaowei Zhou, and Hao Su. Easyhec: Accurate and automatic hand-eye calibration via differentiable rendering and space exploration. *IEEE Robotics and Automation Letters*, 2023.

[23] 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.

[24] Stephanie Cheung and Robert Rohling. Enhancement of needle visibility in ultrasound-guided percutaneous procedures. *Ultrasound in medicine & biology*, 30(5):617–624, 2004.

[25] Zih-Yun Chiu, Florian Richter, Emily K Funk, Ryan K Orosco, and Michael C Yip. Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7737–7743. IEEE, 2021.

[26] Cristina Garcia Cifuentes et al. Probabilistic articulated real-time tracking for robot manipulation. *IEEE Robotics and Automation Letters*, 2(2):577–584, 2016.

[27] Etienne Colle and Simon Galerne. A robust set approach for mobile robot localization in ambient environment. *Autonomous Robots*, 43:557–573, 2019.

[28] Peter I Corke and Seth A Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, 17(4):507–515, 2001.

[29] Jordan M. Croom, D. Caleb Rucker, Joseph M. Romano, and Robert J. Webster. Visual sensing of continuum robot shape using self-organizing maps. In *2010 IEEE International Conference on Robotics and Automation*, pages 4591–4596, 2010.

[30] Nikhil Das and Michael Yip. Learning-based proxy collision detection for robot motion planning applications. *IEEE Transactions on Robotics*, 36(4):1096–1114, 2020.

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

[32] 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.

[33] 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.

[34] 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.

[35] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE, 2005.

[36] 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.

[37] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *International Journal of Computer Vision*, 132(2):581–595, 2024.

[38] 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.

[39] 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.

[40] Dani Goldberg and Maja J Matarić. Maximizing reward in a non-stationary mobile robot environment. *Autonomous Agents and Multi-Agent Systems*, 6:287–316, 2003.

[41] 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.

[42] 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.

[43] 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.

[44] John Hill. Real time control of a robot with a mobile camera. In *Proc. 9th Int. Symp. on Industrial Robots*, pages 233–245, 1979.

[45] 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.

[46] 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.

[47] Matthias Hoffmann, Alexander Brost, Carolin Jakob, Felix Bourier, Martin Koch, Klaus Kurzidim, Joachim Hornegger, and Norbert Strobel. Semi-automatic catheter reconstruction from two views. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 584–591. Springer, 2012.

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

[49] 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.

[50] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[51] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.

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

[53] Tomas Jakab et al. Unsupervised learning of object landmarks through conditional image generation. In *Advances in neural information processing systems*, pages 4016–4027, 2018.

[54] Stephen James et al. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

[55] 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.

[56] Alexander B. Jung et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020.

[57] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. 1961.

[58] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020.

[59] 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.

[60] P. Kazanzides et al. An open-source research kit for the da vinci® surgical system. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6434–6439, May 2014.

[61] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[62] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[63] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

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

[65] Clayton Kunz et al. Deep sea underwater robotic exploration in the ice-covered arctic ocean with auvs. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3654–3660. IEEE, 2008.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] Vincent Lepetit et al. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[71] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[72] Shiyao Li and Guangbo Hao. Current trends and prospects in compliant continuum robots: A survey. *Actuators*, 10(7), 2021.

[73] 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.

[74] 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.

[75] 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.

[76] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[77] Jingpei Lu, Ambareesh Jayakumari, Florian Richter, Yang Li, and Michael C Yip. Super deep: A surgical perception framework for robotic tissue manipulation using deep learning for feature extraction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4783–4789. IEEE, 2021.

[78] Jingpei Lu, Fei Liu, Cédric Girerd, and Michael C Yip. Image-based pose estimation and shape reconstruction for robot manipulators and soft, continuum robots via differentiable rendering. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 560–567. IEEE, 2023.

[79] 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.

[80] Jingpei Lu, Florian Richter, and Michael C Yip. Markerless camera-to-robot pose estimation via self-supervised sim-to-real transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21296–21306, 2023.

[81] Yuning Lu, Jianzhuang Liu, Yonggang Zhang, Yajing Liu, and Xinmei Tian. Prompt distribution learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5206–5215, 2022.

[82] 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.

[83] 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.

[84] Alexander Mathis et al. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9):1281, 2018.

[85] 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.

[86] Annalisa Milella, Giulio Reina, Roland Siegwart, et al. Computer vision methods for improved mobile robot state estimation in challenging terrains. *J. Multim.*, 1(7):49–61, 2006.

[87] Peter Mountney et al. Three-dimensional tissue deformation recovery and tracking. *IEEE Signal Processing Magazine*, 27(4):14–24, 2010.

[88] Prerana Mukherjee, Siddharth Srivastava, and Brejesh Lall. Salient keypoint selection for object representation. In *2016 Twenty Second National Conference on Communication (NCC)*, pages 1–6. IEEE, 2016.

[89] Volker Nannen and Gabriel Oliver. Grid-based spatial keypoint selection for real time visual odometry. In *ICPRAM*, pages 586–589, 2013.

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

[91] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[92] Chrysi Papalazarou, Peter MJ Rongen, et al. 3d catheter reconstruction using non-rigid structure-from-motion and robotics modeling. In *Medical Imaging 2012: Image-Guided Procedures, Robotic Interventions, and Modeling*, volume 8316, pages 622–629. SPIE, 2012.

[93] 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.

[94] Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint arXiv:2203.04291*, 2022.

[95] 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.

[96] 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.

[97] Kristin Y Pettersen. Snake robots. *Annual Reviews in Control*, 44:19–44, 2017.

[98] Jiang Qian and Jianbo Su. Online estimation of image jacobian matrix by kalman-bucy filter for uncalibrated stereo vision feedback. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, volume 1, pages 562–567. IEEE, 2002.

[99] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. IEEE, 2009.

[100] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[101] 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.

[102] Rob Reilink, Stefano Stramigioli, and Sarthak Misra. Pose reconstruction of flexible instruments from endoscopic images using markers. In *2012 IEEE International Conference on Robotics and Automation*, pages 2938–2943, 2012.

[103] Florian Richter et al. Augmented reality predictive displays to help mitigate the effects of delayed telesurgery. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 444–450. IEEE, 2019.

[104] Florian Richter et al. Open-sourced reinforcement learning environments for surgical robotics. *arXiv preprint arXiv:1903.02090*, 2019.

[105] Florian Richter, Peter V Gavrilov, Hoi Man Lam, Amir Degani, and Michael C Yip. Arcsnake: Reconfigurable snakelike robot with archimedean screw propulsion for multidomain mobility. *IEEE Transactions on Robotics*, 38(2):797–809, 2021.

[106] 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.

[107] III Robert J. Webster and Bryan A. Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.

[108] Rethink Robotics. Baxter. *Retrieved Jan*, 10:2014, 2013.

[109] E. Rohmer et al. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com.

[110] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.

[111] David Rollinson, Austin Buchan, and Howie Choset. State estimation for snake robots. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1075–1080. IEEE, 2011.

[112] David Rollinson, Howie Choset, and Stephen Tully. Robust state estimation with redundant proprioceptive sensors. In *Dynamic Systems and Control Conference*, volume 56147, page V003T40A005. American Society of Mechanical Engineers, 2013.

[113] Arthur C Sanderson and Lee E Weiss. Adaptive visual servo control of robots. In *Robot vision*, pages 107–116. Springer, 1983.

[114] Mark Sandler et al. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[115] 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.

[116] 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.

[117] Dimitri A Schreiber, Florian Richter, Andrew Bilan, Peter V Gavrilov, Hoi Man Lam, Casey H Price, Kalind C Carpenter, and Michael C Yip. Arcsnake: An archimedes' screw-propelled, reconfigurable serpentine robot for complex environments. In

2020 IEEE International Conference on Robotics and Automation (ICRA), pages 7029–7034. IEEE, 2020.

[118] James A Sethian. Fast marching methods. *SIAM review*, 41(2):199–235, 1999.

[119] Mili Shah, Roger D Eastman, and Tsai Hong. An overview of robot-sensor calibration methods for evaluation of perception systems. In *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, pages 15–20, 2012.

[120] Chaoyang Shi, Xiongbiao Luo, Peng Qi, Tianliang Li, Shuang Song, Zoran Najdovski, Toshio Fukuda, and Hongliang Ren. Shape sensing techniques for continuum robots in minimally invasive surgery: A survey. *IEEE Transactions on Biomedical Engineering*, 64(8):1665–1678, 2016.

[121] Steven W Squyres et al. Athena mars rover science investigation. *Journal of Geophysical Research: Planets*, 108(E12), 2003.

[122] 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.

[123] Supasorn Suwajanakorn et al. Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *Advances in neural information processing systems*, pages 2059–2070, 2018.

[124] Amy Tabb and Khalil M Ahmad Yousef. Solving the robot-world hand-eye (s) calibration problem with iterative methods. *Machine Vision and Applications*, 28(5):569–590, 2017.

[125] Hashem Tamimi et al. Localization of mobile robots with omnidirectional vision using particle filter and iterative sift. *Robotics and Autonomous Systems*, 54(9):758–765, 2006.

[126] James Thewlis et al. Unsupervised learning of object landmarks by factorized spatial embeddings. In *Proceedings of the IEEE international conference on computer vision*, pages 5916–5925, 2017.

[127] 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.

[128] Josh Tobin et al. 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.

[129] 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.

[130] 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.

[131] Rudolph Van Der Merwe, Eric A Wan, Simon Julier, et al. Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation. In *Proceedings of the AIAA guidance, navigation & control conference*, volume 3, page 08. Providence, RI Providence, RI, 2004.

[132] Lukas von Stumberg, Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. From monocular slam to autonomous drone exploration. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2017.

[133] Michael Waine, Carlos Rossa, Ronald Sloboda, Nawaid Usmani, and Mahdi Tavakoli. 3d shape visualization of curved needles in tissue from 2d ultrasound images using ransac. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4723–4728. IEEE, 2015.

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

[135] Julian Whitman, Nico Zevallos, Matt Travers, and Howie Choset. Snake robot urban search after the 2017 mexico city earthquake. In *2018 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, pages 1–6. IEEE, 2018.

[136] 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.

[137] 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.

[138] Cornell Wright, Aaron Johnson, Aaron Peck, Zachary McCord, Allison Naaktgeboren, Philip Gianfortoni, Manuel Gonzalez-Rivero, Ross Hatton, and Howie Choset. Design of a modular snake robot. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2609–2614. IEEE, 2007.

[139] 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.

[140] Shan Xu, Gaofeng Li, Dezhen Song, Lei Sun, and Jingtai Liu. Real-time shape recognition of a deformable link by using self-organizing map. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 586–591, 2018.

[141] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2878–2890, 2012.

[142] Gang Yao, Ryan Saltus, and Ashwin Dani. Image moment-based extended object tracking for complex motions. *IEEE Sensors Journal*, 20(12):6560–6572, 2020.

[143] Hantao Yao, Rui Zhang, and Changsheng Xu. Visual-language prompt tuning with knowledge-guided context optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6757–6767, 2023.

[144] Michael Yip and Nikhil Das. Robot autonomy for surgery. In *The Encyclopedia of MEDICAL ROBOTICS: Volume 1 Minimally Invasive Surgical Robotics*, pages 281–313. World Scientific, 2019.

[145] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

[146] Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. Distribution-aware coordinate representation for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7093–7102, 2020.

[147] Tongjie Y Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.

[148] Yuting Zhang et al. Unsupervised discovery of object landmarks as structural representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2694–2703, 2018.

[149] 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.

[150] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[151] Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15659–15669, 2023.

[152] Leila Zouaghi, Alexander Alexopoulos, Achim Wagner, and Essameddin Badreddin. Probabilistic online-generated monitoring models for mobile robot navigation using modified petri net. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 594–599. IEEE, 2011.

[153] 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.