

6D Object Pose Estimation for Manipulation via Weak Supervision

Chuer Pan

CMU-RI-TR-22-36

July 2022



The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:

David Held, CMU RI, *chair*
Shubham Tulsiani, CMU RI
Mohit Sharma, CMU RI

*Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics.*

Copyright © 2022 Chuer Pan. All rights reserved.

To my parents, grandparents, and all the people who inspired me.

Abstract

6D object pose estimation is essential for robotic manipulation tasks. Existing learning-based pose estimators often rely on training from labeled absolute poses with fixed object canonical frames, which 1) requires datasets with annotations of object absolute pose that are resource-intensive to collect; 2) is hard to generalize to novel configurations and unseen objects. Instead, we propose to investigate the utilization of relative poses between: i) a single object in different orientations; ii) pairs of interacting objects in manipulation tasks. In this thesis, we show that by using relative poses as weak supervision, we can achieve better label-efficiency and generalizability to novel object configurations and unseen objects.

In the first part of this thesis, we investigate the problem of learning an image-based object pose estimator self-supervised by relative object poses. However, local rotation averaging problems can be difficult to optimize in training due to the closed nature of the rotational manifold of $SO(3)$. To tackle this, we propose a new algorithm that utilizes Modified Rodrigues Parameters to stereographically project 3D rotations from the closed manifold of $SO(3)$ to the open manifold of \mathbb{R}^3 , allowing optimization to be done on an open manifold improving the convergence speed. Empirically, we show that the proposed algorithm is able to converge to a consistent relative orientation frame much faster than algorithms that purely operate in the $SO(3)$ space, and subsequently enabling training pose estimators self-supervised by relative poses.

In the second part, we study the problem of learning task-specific relative pose between interacting objects to solve manipulation tasks. For example, hanging a mug on a rack requires us to reason about relative pose between objects. We conjecture that the relative pose between objects is a generalizable notion of a manipulation task that can transfer to new objects in the same category. We define this as “cross-pose”, and propose a vision-based method that learns to estimate it for a variety of manipulation tasks. Finally, we empirically show that our system is able to generalize to unseen objects in both simulation and the real world from very few demonstrations.

Acknowledgments

I would like to thank my supervisor Prof David Held for his gracious support and guidance throughout my Master's journey. From Dave I have learned the arts of critical thinking, effective experimental design and analysis, and project management. Dave's zest for research, specifically for critically thinking about ways to tackle hard problems is contagiously inspiring, and every brainstorming session with Dave has been an absolute pleasure. Most importantly, I have learned from Dave on how to approach and tackle research problems with both creativity and insight.

I also thank my Ph.D. mentor, Brian Okorn, for his immense guidance and mentorship throughout my Master's journey. I am extremely grateful for having the opportunity to work with Brian on some very interesting projects during my Master's journey. I have learned from Brian on the delicate art of tackling hard problems with both rigor and insight. His unstoppable perseverance in face of obstacles during a project is contagious, and is something I hope will stay with me in spirit for my journey ahead.

I would like to thank Prof Martial Herbert for his valuable feedbacks and advice on the high-level directions of both projects of this thesis. I am grateful for having the opportunity to join his weekly discussion with Brian, and have learned from him the arts of how to critically examine and formulate a research problem, among many other valuable skills.

I would like to thank Prof Shubham Tulsiani and Mohit Sharma for taking time out of their busy schedules to serve on my thesis committee. I am grateful for their valuable feedback and suggestions on the high-level research directions, as well as the presentation of this thesis.

I would also like to thank Prof Zachary Manchester for his insightful and pivotal advice and feedback during the early stages of our MRP project, which pointed us to the right direction to tackle and explain the observations we encountered.

I would also like to thank Prof Keenan Crane, Prof Frank Dellaert, Mark Gillespie, Prof Richard Hartley, Valentin Peretroukhin, Prof David Rosen and Prof Denis Zorin for taking time out their busy schedule to meet with us and providing enormously helpful feedback and advice on the MRP project.

I would like to thank my wonderful collaborators, Ben Eisner and Harry

Zhang, for their contributions of PartNet-Mobility placement experiments for the TAX-Pose project.

Next, I would like to thank the members of R-PAD, it has been an honor and pleasure to have the opportunity to be labmates with all of them. I learned a lot from each and every one of them and feel lucky to be part of such an inclusive and caring lab. And I will definitely miss all the fearless questions during our weekly lab meetings from everyone!

I would especially like to thank my dear friend Colin Li for encouraging and inspiring me to pursue a path in robotics research; I would not have been here if it were not for him, who constantly encouraged me to step out of my comfort zone!

I would like to give special thanks to my partner, Chao Cao for his love and support. I am constantly learning to view my work and life from new perspectives from Chao, and cannot begin to express my gratitude and appreciation for having him by my side!

Finally, I would like to thank my family for their unconditional love and support behind all my decisions. Their constant love, support, and encouragement throughout my Master is an unflinching source of power that fueled me to push through when things got difficult, especially during the pandemic.

Contents

1	Introduction	1
2	Deep Projective Rotation Estimation through Relative Supervision	3
2.1	Introduction	3
2.2	Related Work	5
2.2.1	Averaging and Consensus Estimation	5
2.2.2	Supervised Orientation Estimation	6
2.3	Problem Definition	7
2.4	Preliminaries	8
2.4.1	3D Rotation Representation	8
2.4.2	Lie Group & Lie Algebra	13
2.4.3	Exponential & Logarithmic Map for $SO(3)$	13
2.4.4	Intuitive Implications of Compactness	15
2.5	Baselines	17
2.5.1	Quaternion Averaging	17
2.5.2	$SO(3)$ Averaging	17
2.6	Method	18
2.6.1	Iterative Modified Rodrigues Projective Averaging	19
2.7	Intuitive Example	21
2.7.1	Examples of Critical Points	23
2.8	Experiments	26
2.8.1	Direct Parameter Optimization	28
2.8.2	Neural Network Optimization	30
2.9	Limitations & Future Directions	32
2.10	Conclusions	32
3	Task-Specific Cross-Pose Estimation for Robot Manipulation	35
3.1	Introduction	35
3.2	Related Work	37
3.3	Problem Definition	39
3.4	Method	40
3.4.1	Cross-Pose Estimation via Soft Correspondence Prediction	42
3.4.2	Supervision	44
3.4.3	Overall Training Procedure	46

3.5	Experiments	47
3.5.1	PartNet-Mobility Placement	47
3.5.2	Mug Hanging	50
3.5.3	Failure Cases	56
3.6	Limitations & Future Directions	57
3.7	Conclusions	58
4	Conclusions	59
A	Appendix	61
A.1	Additional Local Rotation Averaging Experiment Results on Structure from Motion Dataset	61
A.2	Definition and Properties of Cross-Pose	67
A.3	Translational Equivariance	71
A.4	Weighted SVD	74
A.5	PartNet-Mobility Objects Placement Task Details	75
A.5.1	Dataset Preparation	75
A.5.2	Metrics	75
A.5.3	Motion Planning	76
A.5.4	Baselines Description	79
A.5.5	Per-Task Results	81
	Bibliography	83

List of Figures

- 2.1 **Illustration of stereographic projection in 2D.** The point of projection is the South Pole of the unit circle, \mathbb{S} , denoted as q_S . The projection line is the horizontal equator of the unit circle, \mathbb{R} . We denote the points on the unit circle as \mathbf{q} 's, and the points on the projection line as ψ 's. Geometrically, for an arbitrary point on the unit circle, \mathbf{q}_i , the corresponding projected point on the projection line is the point of intersection of the line joining itself \mathbf{q}_i and the point of projection \mathbf{q}_S , denoted as ψ_i in the figure. 12

- 2.2 **A simple illustration on the effects of compact vs open manifold on the convergence behaviour of optimization algorithm.** Given a 2D circle, and 3 points that exist on the 2D circle where their true values should overlap, (i.e., their ground truth values represent the same point on the 2D circle), the left shows the valid final configuration. Now assign random position on the circle to the three points as initialization, assuming we have access to the underlying ground truth relative geodesic distance between all pairs of points, which in this case, are all 0, we would like to make pairwise updates to minimize the difference between their current relative geodesic distance and their ground truth relative geodesic distance (0). To execute each update, we take a pair of points, and move both points in the direction that minimize their relative geodesic distance, and repeatedly do this pairwise update for all pairs, until convergence or if maximum number of steps is reached. The first row illustrates the configurations of doing pairwise updates for all pairs of points on a compact manifold of a closed circle, and the second row shows the exact same update procedure but applied on an open manifold of a curved line. In this case, we see that optimization on the compact manifold results in failure in convergence, while optimization on an open manifold results in progress in convergence. 16

2.3	Illustration of Iterative Modified Rodrigues Projective Averaging. Projection of relative supervision, q_i^j , shown in red, from back-projected rotation $\hat{q}_j := \phi^{-1}(\hat{\psi}_j)$ to \hat{q}_j into the MRP space update, ϕ_Δ , shown in green. While \tilde{q}_i could have been selected as the the goal rotation, it would have induced a much larger movement in the projected space.	19
2.4	An illustration of the intuition behind how MRP reduces the dimension of the set of problematic local optima through stereographic projection. (a) Take two orientation quaternions in \mathbb{S}^3 , shown in red and blue. We see that all points along the orange equator are valid averages of these two quaternions in \mathbb{S}^3 ; (b) We can reduce the dimension of the set of valid averages by using MRP to project these orientations into the hyperplane of \mathbb{R}^3 ; (c) And instead, take the average of the two projected quaternion as seen in red and blue circles, which gives us the average as the orange circle in \mathbb{R}^3 ; (d) The average in the projected space can be projected back to the original orientation space when needed. Thus we have successfully <i>reduced</i> the dimension of the set of problematic local optima from an equator of points to one point on the original manifold of 3D rotations (in this specific case, the unit quaternion sphere).	23
2.5	Relative rotation consensus results for direct parameter optimization. Relative rotation consensus with direct optimization of rotation parameters over 50 unique environments with 100 random generated orientations each (left) and Alamo 1DSfM [89] (right). Median average-pair-wise angular error ($^\circ$) between each estimated rotations is shown, with shaded region representing the first and third quartile for each method. The max average-pair-wise angular error for each algorithm at each iteration is shown as a dashed line.	26
2.6	Experiment setup for image-based orientation estimation using neural network optimization. Given the YCB drill model, the model is randomly rendered at two orientations, producing a pair of images, (I_i, I_j) , with ground truth relative rotation between them as R_j^i . We then learn a function $f(\cdot) : \mathcal{I} \rightarrow SO(3)$ parameterized by a ResNet18 model that takes in an image of an object, I_i , and outputs an absolute orientation prediction \hat{R}_i such that the geodesic distance between the ground truth relative rotation R_j^i and the relative rotation induced from the pair of orientation prediction (\hat{R}_i, \hat{R}_j) is minimized.	30

2.7	Illustration of curriculum used for training. With a constant anchor orientation denoted by the dashed line, the curriculum range is defined as the orientation range which is θ degree away from this constant anchor orientation. In the figure, this curriculum range is defined by the area enclosed by the orange lines. This curriculum range is increased whenever the training error drops below a given error threshold, until θ reaches 180° . The pair of orientations for training is always selected to be within $\alpha = 30^\circ$ of each other, whilst both located within the curriculum range. For example, the blue and red triangles represent some valid orientation pairs.	32
2.8	Curriculum Angle (left) and Average Pairwise Error (right), sampled over the full orientation space for three training sessions with each method. Median average-pairwise angular error ($^\circ$) is shown with shaded areas representing the first and third quartile over all training sessions. The max average-pairwise angular error for each algorithm at each iteration is shown as a dashed line.	33
2.9	Neural net optimization results. Estimated rotation frame learned for the YCB [92] drill model using Iterative Modified Rodrigues Projective Averaging and relative rotations (\mathbf{x} , \mathbf{y} , \mathbf{z}) (left). Results for rotations estimated by neural networks given images of the YCB drill [92] rendered at each of 100 random rotations with various supervisions, (right). Median average-pairwise angular error ($^\circ$) is shown with shaded areas representing the first and third quartile over all training sessions. The max average-pairwise angular error for each algorithm at each iteration is shown as a dashed line.	34
3.1	TAX-Pose in action. Top: PartNet-Mobility Placement Task. Bottom: Mug Hanging Task. The model is trained using demonstrations of anchor and action objects in their ground-truth cross-pose for the task. The model first observes the initial configuration of the objects, estimates correspondence between the pair of objects, and then calculates the desired pose. The desired pose is then used to guide robot motion planning.	36
3.2	Visualization of the properties of cross-pose. If we transform both \mathbf{P}_A (mug) and \mathbf{P}_B (rack) objects by the same transform, then the relative pose between these objects is unchanged (the mug is still “on” the rack) so the cross-pose is unchanged.	40

3.3	TAX-Pose training overview. Our method takes as input two point clouds given a specific task and outputs the cross-pose between them for the task. TAX-Pose first learns point clouds features using two DGCNN networks and two transformers. Then the learned features will be input to two point residual networks to predict per-point soft correspondence, correspondence residuals and SVD weights between the two objects. Then the desired cross-pose can be inferred analytically using weighted singular value decomposition.	41
3.4	Computation of corrected virtual correspondence. Given a pair of objects \mathcal{A}, \mathcal{B} , a per-point soft correspondence $v_i^{A \rightarrow B}$ is first computed. Next to allow the predicted correspondence to lie beyond object’s convex hull, these soft correspondences are adjusted with correspondence residuals, $r_i^{A \rightarrow B}$, which results in the corrected virtual correspondence, $\tilde{v}_i^{A \rightarrow B}$	44
3.5	Illustration of weighted SVD. Given a point cloud, \mathbf{P}_A , the associated corrected virtual correspondence, $\tilde{\mathbf{V}}_B$, and the per-point weights predicted, α_A , we extract a $SE(3)$ transformation, \mathbf{T} . We can then apply to the original point cloud \mathbf{P}_A to result in a desirable cross-pose for the task.	45
3.6	Real-world experiments illustration. Left: work-space setup for physical experiments. Center: Octomap visualization of the perceived anchor object.	49
3.7	Visualization of mug hanging task (upright pose). Mug hanging task is consisted of two stages, given a mug that is randomly initialized on the table, the model first predicts a $SE(3)$ transform from gripper end effector to the mug rim $\mathbf{T}_{g \rightarrow m}$, then grasp it by the rim. Next, the model predicts another $SE(3)$ transform from the mug to the rack $\mathbf{T}_{m \rightarrow r}$ such that the mug handle gets hanged on the the mug rack.	50
3.8	Failure of <i>grasp</i> prediction. Predicted TAX-Pose for the gripper misses the rim of mug.	57
3.9	Failure of <i>place</i> prediction. Predicted TAX-Pose for mug results in the mug handle misses the rack hanger completely.	57
3.10	An illustration of unsuccessful TAX-Pose predictions for mug hanging. In both subfigures, red points represent the anchor object, blue points represent action object’s starting pose, and green points represent action object’s predicted pose.	57

A.1	Optimization results for all 1DSfM [89] datasets, ordered by number of cameras (N). Median average-pairwise angular error ($^{\circ}$) is shown with shaded areas representing the first and third quartile over all training sessions. The max average-pairwise angular error for each algorithm at each iteration is shown as a dashed line.	62
A.2	Failure of "In" prediction. Predicted TAX-Pose violates the physical constraints by penetrating the oven base too much.	77
A.3	Failure of "Left" prediction. Predicted TAX-Pose violates the physical constraints by being in collision with the leg of the drawer.	77
A.4	An illustration of unsuccessful real-world TAX-Pose predictions. In both subfigures, red points represent the anchor object, blue points represent action object's starting pose, and green points represent action object's predicted pose.	77
A.5	A visualization of all categories of anchor objects and associated semantic tasks, with action objects in ground-truth TAX-Poses used in simulation training.	80

List of Tables

2.1	Number of iteration steps until convergence and Normalized Area Under Curve (nAUC) over 50 unique environments of 100 randomly generated orientations. 300K optimization steps are taken for each experiment.	27
2.2	Percentage of experiments converged and final angular errors over 50 unique environments of 100 randomly generated orientations. 300K optimization steps are taken for each experiment.	27
2.3	Final results on 1DSfM [89] datasets after 20K iterations	27
2.4	Final results for image based rotation estimation.	29
3.1	Goal Inference Rotational and Translational Error Results (\downarrow). Rotational errors ($\mathcal{E}_{\mathbf{R}}$) are in degrees ($^{\circ}$) and translational errors ($\mathcal{E}_{\mathbf{t}}$) are in meters (m). The lower the better.	48
3.2	Real-world goal placement success rate	49
3.3	Mug on Rack Simulation Task Success Results	52
3.4	# Demos vs. Overall Success	53
3.5	Mug Hanging Ablations Results	56
A.1	Final Mean Absolute Error ($^{\circ}$) on all 1DSfM [89] datasets after 20K iterations	63
A.2	Final Median Absolute Error ($^{\circ}$) on all 1DSfM [89] datasets after 20K iterations	64
A.3	Final Mean Normalized AUC on all 1DSfM [89] datasets after 20K iterations	65
A.4	Final Mean Relative Error ($^{\circ}$) on all 1DSfM [89] datasets after 20K iterations	66
A.5	Dataset sizes and observation accuracies ($^{\circ}$) for all 1DSfM [89] datasets	67
A.6	Goal Inference Rotational and Translational Error Results (\downarrow) for the “ In ” Goal. Rotational errors ($\mathcal{E}_{\mathbf{R}}$) are in degrees ($^{\circ}$) and translational errors ($\mathcal{E}_{\mathbf{t}}$) are in meters (m). The lower the better.	80
A.7	Goal Inference Rotational and Translational Error Results (\downarrow) for the “ On ” Goal. Rotational errors ($\mathcal{E}_{\mathbf{R}}$) are in degrees ($^{\circ}$) and translational errors ($\mathcal{E}_{\mathbf{t}}$) are in meters (m). The lower the better.	80

A.8	Goal Inference Rotational and Translational Error Results (\downarrow) for the “ Left ” Goal. Rotational errors ($\mathcal{E}_{\mathbf{R}}$) are in degrees ($^{\circ}$) and translational errors ($\mathcal{E}_{\mathbf{t}}$) are in meters (m). The lower the better.	81
A.9	Goal Inference Rotational and Translational Error Results (\downarrow) for the “ Right ” Goal. Rotational errors ($\mathcal{E}_{\mathbf{R}}$) are in degrees ($^{\circ}$) and translational errors ($\mathcal{E}_{\mathbf{t}}$) are in meters (m). The lower the better. . . .	81
A.10	Combined per-task results for real-world goal placement success rate.	82

Chapter 1

Introduction

6D Object pose estimation is the core component to enabling functional, robust robotic manipulation system. Deep learning has offered a way to develop image-based or pointcloud-based pose estimators; however, such estimators often require training on a large labeled dataset, which can be time-intensive to collect. In this thesis work, we explore different approaches to obtain accurate, generalizable 6D object pose estimator for manipulation tasks with minimal supervision required.

In the first part the work, we explore whether self-supervised learning from unlabeled image data can be used to alleviate this issue. By taking advantage of the Specifically, we assume access to estimates of the relative orientation between neighboring poses, such that can be obtained via a local alignment method. While self-supervised learning has been used successfully for translational object keypoints, in this work, we show that naively applying relative supervision to the rotational group $SO(3)$ will often fail to converge due to the non-convexity of the rotational space. To tackle this challenge, we propose a new algorithm for self-supervised orientation estimation which utilizes Modified Rodrigues Parameters to stereographically project the closed manifold of $SO(3)$ to the open manifold of \mathbb{R}^3 , allowing the optimization to be done in an open Euclidean space. We empirically validate the benefits of the proposed algorithm for rotational averaging problem in two settings: (1) direct optimization on rotation parameters, and (2) optimization of parameters of a convolutional neural network that predicts object orientations from images. In both settings, we demonstrate that our proposed algorithm is able to converge to a

1. Introduction

consistent relative orientation frame much faster than algorithms that purely operate in the $SO(3)$ space.

In the second part of this work, we explore whether self-supervised learning from unlabeled point cloud data can be used to alleviate this issue. Specifically, we take a step back from the problem scope of 6D object pose estimation, and shift our focus back to “manipulation” by thinking critically about what kind of pose estimator is really necessary and best-suited for manipulation tasks. Specifically, we think about the problem of,

How do we imbue robots with the ability to efficiently manipulate unseen objects and transfer relevant skills based on demonstrations?

End-to-end learning methods often fail to generalize to novel objects or unseen configurations. Instead, we conjecture that the task-specific pose relationship between relevant parts of interacting objects is a generalizable notion of a manipulation task that can transfer to new objects in the same category; examples include the relationship between the pose of a lasagna relative to an oven or the pose of a mug relative to a mug rack. We call this task-specific pose relationship “cross-pose” and provide a mathematical definition of this concept. We propose a vision-based system that learns to estimate the cross-pose between two objects for a given manipulation task. The estimated cross-pose is then used to guide a downstream motion planner to manipulate the objects into the desired pose relationship (placing the lasagna into the oven or the mug onto the mug rack). We train a cross-pose estimator in simulation and we demonstrate the capability of our system to generalize to unseen objects in both simulation and the real world, deploying our policy on a Franka Emika with no finetuning.

Chapter 2

Deep Projective Rotation Estimation through Relative Supervision

2.1 Introduction

Pose estimation is a critical component for a wide variety of computer vision and robotic tasks. It is a common primitive for grasping, manipulation, and planning tasks. For motion planning and control, estimating an object’s pose can help a robot avoid collisions or plan how to use the object for a given task. The current top performing methods for pose estimation use machine learning to estimate the object’s pose from an image; however, training these estimators tends to rely on direct supervision of the object orientation [43, 83, 92]. Obtaining such supervision can be difficult and requires either time-consuming annotations or synthetic data, which might differ from the real world. In this work, we explore whether self-supervised learning can be used to alleviate this issue by training an object orientation estimator from unlabeled data. Specifically, we assume that we can estimate the relative rotation of an object between neighboring object poses in a self-supervised manner. Such relative supervision can be easily obtained in practice, for example through a local registration method such as ICP [96] or camera pose estimation.

2. Deep Projective Rotation Estimation through Relative Supervision

Relative self-supervision has been previously used for representation learning in estimating translational keypoints [50, 71, 72]. These methods use only relative supervision to ensure that the keypoints are consistent across views of the object, and do not directly supervise the keypoint locations. In this work, we explore whether such relative self-supervision can similarly be used in estimating object orientations. We show that naively applying such relative supervision to rotations on the $SO(3)$ manifold will often fail to converge. Unlike self-supervised learning of translational keypoints, the rotational averaging problem [32] is inherently non-convex, with many local optima. While there exist global optimization algorithms which jointly optimize all pairs of rotations for this problem [23, 84], they are not easily integrated into the iterative, stochastic gradient descent methods used to train neural network-based pose estimators.

To address this issue, we propose a new algorithm, Iterative Modified Rodrigues Projective Averaging, which uses Modified Rodrigues Parameters to map from the closed manifold of $SO(3)$ to the open space of \mathbb{R}^3 . In doing so, we obtain faster convergence with a lower likelihood of falling into local optima. Our experiments show that our method converges faster and more consistently than the standard $SO(3)$ optimization and can easily be integrated into a neural network training pipeline. Additionally, we include an intuitive theoretical example describing how, while not all local optima are removed, the dimensionality of a set of problematic configurations is greatly reduced when optimizing using our algorithm, as compared to optimizing in the space of $SO(3)$.

The primary contributions of this work are:

- We propose a new algorithm, *Iterative Modified Rodrigues Projective Averaging*, which is an iterative method for learning rotation estimation using only relative rotation supervision and can be applied to neural network optimization.
- We empirically investigate the convergence behavior of our algorithm as compared to optimizing on the $SO(3)$ manifold.
- We demonstrate that our algorithm can be used to train a neural network-based pose estimator using only relative supervision.

2.2 Related Work

2.2.1 Averaging and Consensus Estimation

Consensus methods, sometimes referred to as averaging methods, have a long history of research. The goal of these methods is, given a distributed set of estimates, to produce a consistent prediction of a value using relative information. While there are iterative algorithms with good convergence properties in Euclidean space [7, 16, 22, 51, 62], optimizing over the closed manifold of $SO(3)$ can be more difficult, as the region is non-convex, with many local minima.

Rotation consensus, or what is commonly known as the rotation averaging problem, can be solved under certain conditions. Manton [49] describes an iterative method for computing the mean of a set of rotations, through Riemannian optimization, though this algorithm is not guaranteed to produce a unique solution if the rotations do not lie in a ball of radius $r \leq \frac{\pi}{2}$ with respect to angular distance. Similarly, Tron et al. [76], proposes a Riemannian optimization method via tangent space updates in $SE(3)$ and proves that it will converge to local minima, with later evaluation of these minima [77, 78]. In the case where all orientations are in a planar configuration, Piova et al. [56] provides a distributed method of localizing orientations. Hartley et al. [31, 32] describe several methods of finding a consistent set of rotations, though their convergence is similarly not guaranteed outside of a radius $r \leq \frac{\pi}{2}$ ball in $SO(3)$. Wang and Singer [84] find an exact solution to this problem, using a combination of a semidefinite programming relaxation and a robust penalty function. More recently, Shonan Rotation Averaging [23] shows that projecting to higher dimensional spaces allows for the recovery of a globally optimal solution using semidefinite programming.

These existing rotation averaging solvers either 1) require solving a semidefinite programming problem; or 2) are global methods that require optimizing over the entire set of relative orientations at once. These two settings are not well-applicable for learning-based orientation estimators that are learned using neural networks. Specifically, integrating semidefinite programming as differentiable trainable layers in a neural net is still an active research area [2, 82]. Moreover, global rotation averaging solvers will lead to memory explosion issue when applied directly on a neural net model. This is due to the fact that deep learned models commonly rely

on back propagation to optimize its model weights. Specifically, back propagation requires the model to compute and store every intermediate gradients of the loss with respect to each weight per input to update the model weights. And doing so for the entire set of relative orientation dataset is practically infeasible and will lead to memory explosion. This is in fact the reason why in learning settings, it's common practice to train model in a mini-batch manner, where the model is optimized over a mini-batch of the whole dataset during each update step, as in the case of stochastic gradient descent.

Therefore, existing rotation averaging methods are infeasible for our problem setup which requires doing local iterative updates with a differentiable neural network.

2.2.2 Supervised Orientation Estimation

Past work has explored using a neural network to predict an object's orientation. Traditionally, these methods rely on supervising the rotations using a known absolute orientation, whether in the form of quaternions [40, 41, 92], axis-angle [25], or Euler angles [69]. More recently, 6D [43, 97], 9D [44], and 10D [55] representations have been developed for continuity and smoothness. Recently, Terzakis et al. [74] reintroduced Modified Rodrigues Parameters, a projection of the unit quaternion sphere \mathbb{S}^3 to \mathbb{R}^3 commonly used in attitude control literature [19], to a range of common computer vision problems. Terzakis et al. [74] does not, however, address the unique problems found in the rotation averaging problem.

Some methods, such as DeepIM [45], have posed the rotation estimation problem purely as a relative problem, computing the transform to rotate from one object pose to another. Similarly, $se(3)$ -TrackNet [87] tracks object pose using a Lie Algebra-based orientation update. While these methods do remove the need for absolute supervision, the resulting estimates are only useful when compared to an anchor image with an absolute orientation given. In practice, obtaining an absolute pose can be useful for both planning and joint learning of orientation representation and control. For this reason, we seek to estimate an absolute pose using relative supervision.

Recently, there has been research into mapping the Riemannian optimization to the Euclidean optimization used for network training [5, 11, 14, 17, 73]. These methods focus on applying tangent space gradients from losses in 3D transformation groups.

Specifically, Projective Manifold Gradient Layer [17] ensures that the gradients take into account any projection operations, such that the gradients point towards the nearest valid representation in the projection’s preimage. While this does map the Riemannian optimization into a Euclidean problem, it does not solve the problems caused by the closed manifold of $SO(3)$, as this does not alter the underlying topology of this manifold.

2.3 Problem Definition

We formally describe the problem of self-supervised orientation estimation below. We assume that we are given a set of inputs observations $\{I_1, \dots, I_N\}$, of an object where, in each input observation I_i , the object is viewed from an unknown orientation R_i . These inputs could be in the form of images, point clouds, or some other object representation. While we do not know the absolute object orientations R_i in any reference frame, we assume that we do know a subset of the relative rotations R_i^j , possibly from a local registration method like ICP, between the object in images I_j and I_i , such that $R_i = R_i^j R_j$. Our goal is to learn a function $f(I_i)$ that estimates an orientation of the object in each image, $f(I_i) = \hat{R}_i$ that minimizes the pairwise error of all input pairs, with respect to the geodesic distance metric $d(R_i, R_j) = \|\log(R_i^\top R_j)\|^2$. Given a set of rotations $\mathcal{R} = \{R_1, \dots, R_N\}$, the core optimization objective is thus:

$$\min_{\hat{R}_i, \hat{R}_j \in \mathcal{R}} \sum_{i,j} d(R_i, R_i^j R_j) \quad (2.1)$$

Note that this optimization does not have a unique solution, since the solution $\hat{R}_i := S R_i, \forall i$ minimizes this error for any constant rotation S .

In many robotics tasks, relative rotations can be accurately estimated only when their magnitude is small as many registration algorithms, such as ICP, requires a good initialization near the optimum. Following this observation, we assume that we can only accurately supervise relative rotations when they are small in magnitude. This leads to a local neighborhood structure where each rotation R_i is connected to R_j only in a local neighborhood around R_i , when $d(R_i, R_j) < \epsilon$, and the set of all R_j ’s connected to R_i form the neighborhood set of \mathcal{N}_i . While the algorithms described in this manuscript do not rely on this angle ϵ , it can be scaled as needed based on the

accuracy of the relative rotation estimation method (e.g. ICP, camera ego motion estimation).

Our eventual goal is to represent the function $f(I_i) = \hat{R}_i$ as a neural network. Thus, we restrict the methods with which we compare to iterative methods that are updated using only a sampled subset of the rotations (as opposed to methods that perform a global optimization over the entire set of rotations $\{R_1, \dots, R_N\}$). This requirement is to match the conditions required by stochastic gradient descent, the primary method of training neural networks.

2.4 Preliminaries

2.4.1 3D Rotation Representation

Every rotation in 3D can be defined by two elements,

- An axis of rotation through the origin
- An angle of rotation about the axis of rotation defined

For consistency, in this work, we follow the right-hand rule by assuming that the angle of rotation always goes counter-clockwise about the axis of rotation.

There exist many different parameterization of 3D rotation, each with its own pros and cons, as discussed below.

Rotation Matrix

Rotations are linear transformations of \mathbb{R}^3 , and therefore can be represented by matrices given that a basis in \mathbb{R}^3 is chosen. If we choose a basis formed by 3 orthonormal vectors in \mathbb{R}^3 , then this leads to a set of orthogonal 3×3 matrices that uniquely describe every 3D rotation, called “Special Orthogonal Group” in dimension 3, defined as

$$SO(3) \triangleq \{R \in \mathbb{R}^{3 \times 3} | R^\top R = \mathbb{I}_{3 \times 3}, \det(R) = 1\} \quad (2.2)$$

forms a matrix Lie group. The set of all rotation matrices $SO(3)$ is commonly referred to as the 3D rotation group. A rotation matrix has 9 variables with 3 degrees of freedom (DoF); the rest of the 6 variables are constrained via orthonormality.

Rotation matrices are arguably the most intuitive and fundamental representation of 3D rotation, with all useful rotation operations can be carried out via standard matrix operations. Specifically, rotation composition can be carried out via standard matrix multiplication, rotation can be applied to 3D point via matrix-vector multiplication, and the inverse can be obtained via matrix transportation. However, the high-dimensional nature of rotation matrix means that it could be very cumbersome to maintain and impose the orthonormality constraints in practice.

Euler Angles

Euler angles are 3-dimensional rotation representation defined as the composition of three elemental rotations, which are rotations around the orthogonal axes of the Cartesian coordinate system. The three rotations are usually denoted by the angle of rotation in radians, such as (ϕ, θ, ψ) or (α, β, γ) . Since rotation composition is non-commutative, the order of the axes of rotations matters for the definition of any set of Euler angles. There exist in total 12 possible sets of Euler angle conventions, depending on the order of the 3 rotation axes. The axes can be of two types, either of the form A-B-A, or A-B-C, where each unique letter represents one of the XYZ axes. The three elemental rotations can be defined with respect to a static global coordinate system, called *extrinsic*, or with respect to the moving body frame, called *intrinsic*. The intrinsic Euler angles of the form A-B-C are usually referred to as yaw-pitch-roll, commonly used in aerospace.

Although there is great merit in the intuitive geometric interpretation of the representation of Euler angles, the literature has generally agreed that the downsides of Euler angles make them far from ideal candidates for rotation representation in practice [28, 74]. First, notice that even under a particular order of axes out of the 12 possible orders, Euler angles are not unique, since an unique 3D rotation can be represented by many non-unique Euler angles representations. More critically, Euler angles suffer from singularities known as gimbal lock, where at certain configurations when two of the three rotation axes align, which renders the last rotation axis useless, resulting in the loss of 1 DoF. More concretely, every rotation in Euler angles representation is less than or equal to $\pi/2$ away from the nearest singularity.

Axis-Angle

Every rotation in 3D can be represented by rotating around an axis of rotation that can be represented by a unit \mathbb{R}^3 vector \mathbf{u} , by a certain rotation angle θ . The multiplication of the unit vector and the angle of rotation gives rise to the angle-axis representation, a 3-dimensional rotation representation defined as,

$$\omega \triangleq \theta \mathbf{u} \in \mathbb{R}^3 \quad (2.3)$$

Notice that for any rotation represented by θu , the same rotation can be achieved by rotating around u by $-(2\pi - \theta)$, thus we see that the axis-angle is not a unique mapping of the $\text{SO}(3)$ group, and in fact singularities exist at $\theta = 2k\pi, \forall k \in \mathbb{N}_+$.

Unit Quaternions

The discovery of the group of quaternions have historically been of fundamental importance to the study of 3D rotations [3]. Quaternions are 4-dimensional real vectors. The set of all quaternion is denoted as \mathbb{H} and a quaternion is denoted by $\mathbf{q} = (\rho, v_1, v_2, v_3)$:

$$\mathbb{H} \triangleq \{\mathbf{q} | \mathbf{q} = \rho + v_1 i + v_2 j + v_3 k\} \quad (2.4)$$

where

- $\rho, v_1, v_2, v_3 \in \mathbb{R}$
- i, j, k are the quaternion basis units satisfying $i^2 = j^2 = k^2 = ijk = -1$

Quaternions can also be expressed in the form of,

$$\mathbf{q} = \rho + \mathbf{v}^\top \begin{bmatrix} i \\ j \\ k \end{bmatrix} \quad (2.5)$$

where $\rho \in \mathbb{R}$ is the scalar part and $\mathbf{v} \in \mathbb{R}^3$ is the vector part. Unit quaternions live on the unit hypersphere, \mathbb{S}^3 in \mathbb{R}^4 defined as

$$\mathbb{S}^3 \triangleq \{\mathbf{q} \in \mathbb{H} | \|\mathbf{q}\| = \sqrt{\rho^2 + v_1^2 + v_2^2 + v_3^2} = 1\} \quad (2.6)$$

Being a four-dimensional representation, with the added degree of redundancy, unit quaternions do not suffer from singularities like the other three-dimensional representations. However, its representation is not unique; any quaternion \mathbf{q} and its inverse $-\mathbf{q}$ represent the same 3D rotation, any such pair is called antipodal points. In other words, the unit quaternion group is a double cover of the 3D rotational group $SO(3)$. Furthermore, quaternions require an extra parameter, leading to a non-minimal parameterization, and require maintenance of the unit norm constraint.

Modified Rodrigues Parameters

Stereographic projection is a bijective conformal mapping to map from a sphere to a plane. It is a well-defined, one-to-one mapping except at the point of projection, where it is mapped to points of infinity on the projected plane. Modified Rodrigues parameters utilize stereographic projection to map from the unit quaternion hypersphere \mathbb{S}^3 to the open Euclidean hyperplane of \mathbb{R}^3 , with singularities at $\pm 2\pi$. This parameterization has been well established and is a popular choice for attitude control in the aerospace literature [20, 64, 67, 79].

Stereographic projection projects points from the hypersphere onto the hyperplane by designating a *point of projection* on the hypersphere and a *projection hyperplane*. Then it shoots lines from the point of projection to all points on the hypersphere, and the corresponding projected point for an arbitrary point on the hypersphere is defined as the point of intersection between the projection plane and the line joining itself and the point of projection.

Figure 2.1 shows a geometric illustration of a 2D stereographic projection, projected from a 2D unit sphere \mathbb{S} to a 1D Euclidean line \mathbb{R} at the equator of the sphere.

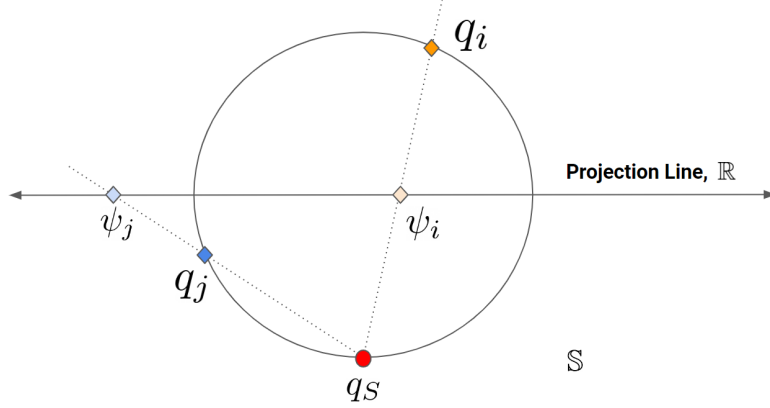


Figure 2.1: **Illustration of stereographic projection in 2D.** The point of projection is the South Pole of the unit circle, \mathbb{S} , denoted as q_S . The projection line is the horizontal equator of the unit circle, \mathbb{R} . We denote the points on the unit circle as \mathbf{q} 's, and the points on the projection line as ψ 's. Geometrically, for an arbitrary point on the unit circle, \mathbf{q}_i , the corresponding projected point on the projection line is the point of intersection of the line joining itself \mathbf{q}_i and the point of projection \mathbf{q}_S , denoted as ψ_i in the figure.

Modified Rodrigues parameters (MRP), result from the application of stereographic projection in 4D, mapping from the 4D unit sphere \mathbb{S}^3 to the 3D Euclidean hyperplane \mathbb{R}^3 . In fact, Figure 2.1 is a simplified illustration of the mapping between the quaternions on the 4D unit sphere and the corresponding MRP in the lower-dimensional space of 2D, since visualizing the 4D topology on 2D paper is almost impossible. In MRP, the South Pole of the unit quaternion sphere ($\mathbf{q}_S = -1 + 0i + 0j + 0k$) is the point of projection, and the projection plane is the 3D plane that slides through the unit sphere at the imaginary equator, spanned by the quaternion basis units, i, j, k . MRPs are denoted by $\psi \in \mathbb{R}^3$, which can be obtained from quaternion $q = \rho + \mathbf{v}^\top [i \ j \ k]^\top = [\rho \ \mathbf{v}] \in \mathbb{H}$ by,

$$\psi = \frac{\mathbf{v}}{1 + \rho} \quad (2.7)$$

Inversely, given a modified Rodrigues parameter, ψ , the unprojected corresponding unit quaternion is given by

$$q = \frac{1 - \|\psi\|^2}{1 + \|\psi\|^2} + \frac{2}{1 + \|\psi\|^2} \psi^\top \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} \frac{1 - \|\psi\|^2}{1 + \|\psi\|^2} & \frac{2}{1 + \|\psi\|^2} \psi^\top \end{bmatrix} \quad (2.8)$$

Modified Rodrigues parameters are a minimal parameterization of rotation of just 3 dimensions. It also allows for the representation of rotations on a noncompact hyperplane, rather than staying on the compact manifold of $SO(3)$, which is of key significance for our work, as shown later.

2.4.2 Lie Group & Lie Algebra

Matrix Lie group is a differential manifold that locally resembles a Euclidean space, which is composed of $n \times n$ invertible matrices. Every Lie group is associated with a Lie algebra, which is a vector space that is the tangent space of the Lie group at identity. The 3D rotation group $SO(3)$ forms a matrix Lie group, and its associated Lie algebra is $\mathfrak{so}(3)$.

The Lie algebra $\mathfrak{so}(3)$ is characterized by the vector space of 3×3 skew-symmetric matrices [32], which are usually denoted as $\Omega \in \mathbb{R}^{3 \times 3}$, that can be represented by a vector $\omega = [\omega_1, \omega_2, \omega_3]^\top \in \mathbb{R}^3$ containing the entries of the skew-symmetric matrix,

$$\Omega = [\omega]_\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \quad (2.9)$$

2.4.3 Exponential & Logarithmic Map for $SO(3)$

To map elements from a Lie algebra to its associated matrix Lie group, one utilizes the *exponential map*. Inversely, *logarithmic map* maps from a Lie group to its Lie algebra.

Exponential Map Formally, if G is a matrix Lie group with Lie algebra \mathfrak{g} , then the exponential map for G is defined as the map [30],

$$\exp(\cdot) : \mathfrak{g} \rightarrow G \quad (2.10)$$

For the matrix Lie group of $SO(3)$, there are two common methods for computing the exponential map from $\mathfrak{so}(3)$ to $SO(3)$. One way is to compute the matrix exponential of the skew-symmetric matrix $\Omega \in \mathfrak{so}(3)$, and map it to its corresponding

3D rotation matrix $R \in SO(3)$ as,

$$R = \exp(\Omega) = \sum_{k=0}^{\infty} \frac{\Omega^k}{k!} \quad (2.11)$$

Computing matrix exponentials are expensive and can only be computed approximately in practice, so this method of computing the exponential map is actually not commonly used in practice.

Another way to compute the exponential map is to map from the axis-angle representation $\omega \in \mathbb{R}^3$ to rotation matrix $R \in SO(3)$. This is enabled by the Rodrigues' rotation formula, which provides a method to compute the exponential map from $\mathfrak{so}(3) \rightarrow SO(3)$ without needing to compute the full matrix exponential [75]. We see that given an axis-angle representation $\omega \in \mathbb{R}^3$, we can convert it to a skew-symmetric matrix $\Omega = [\omega]_{\times}$ through Eq. 2.9, which defines a bijection mapping $[\cdot]_{\times} : \mathbb{R}^3 \rightarrow \mathfrak{so}(3)$. In fact, the vector space \mathbb{R}^3 is a vector space isomorphism for $\mathfrak{so}(3)$ since it preserves both addition and scalar multiplication [32]. In fact, under the Lie-bracket defined as $[\omega, \eta] = \omega \times \eta$, where $\omega, \eta \in \mathbb{R}^3$, this map is a Lie-algebra isomorphism between \mathbb{R}^3 and $\mathfrak{so}(3)$.

Given the geometric interpretation of axis-angle representation, we have $\omega = \theta \mathbf{u}$, where $\mathbf{u} = \frac{\omega}{\|\omega\|}$ is the unit vector representing the axis of rotation, and $\theta = \|\omega\|$ is the angle of rotation around the axis \mathbf{u} .

Thus, via Lie-algebra isomorphism, we can establish the exponential map of,

$$\exp[\cdot]_{\times} : \mathbb{R}^3 \rightarrow SO(3) \quad (2.12)$$

where the conversion between an axis-angle $\omega = \theta \mathbf{u}$ to its associated rotation matrix R is given by the Rodrigues' formula,

$$R = I + (\sin \theta)[\mathbf{u}]_{\times} + (1 - \cos \theta)[\mathbf{u}]_{\times}^2 \quad (2.13)$$

In this work, we always refer to this mapping between the axis-angle representation $\omega \in \mathbb{R}^3$ and $R \in SO(3)$ when we talk about the exponential map and logarithmic map for $SO(3)$.

Logarithmic Map Formally, if G is a matrix Lie group with Lie algebra \mathfrak{g} , then

the logarithmic map for G is defined as the map [30],

$$\log(\cdot) : G \rightarrow \mathfrak{g} \quad (2.14)$$

Following the second definition of an exponential map from Eq. 2.12, defined as the mapping between the axis-angle representation and the rotation matrix, the logarithmic map of $SO(3)$ is defined as the mapping of,

$$\log(\cdot) : SO(3) \rightarrow \mathbb{R}^3 \quad (2.15)$$

Since the exponential map is not one-to-one and, in fact, periodic, the inverse can be defined if we restrict the angle of rotation to the range of $\theta \in (-\pi, \pi)$, as,

$$[\omega]_{\times} = \log(R) = \frac{\theta}{2\sin\theta}(R - R^{\top}) \quad (2.16)$$

And the angle of the rotation, θ is given by

$$\theta = \arccos \frac{\text{Tr}(R) - 1}{2} \quad (2.17)$$

In this work, when referring to the geodesic metric on $SO(3)$, we always assume that it is the angle metric, which is the angle between two rotations.

2.4.4 Intuitive Implications of Compactness

The 3D rotational space of $SO(3) \triangleq \{R \in \mathbb{R}^{3 \times 3} : R^{\top}R = \mathbb{I}_{3 \times 3}, \det(R) = 1\}$ is a compact matrix Lie group, which topologically is a compact manifold, which is characterized by topologically wrapping around itself. Furthermore, the $SO(3)$ manifold is compact and has no boundary, making it a closed manifold. On the other hand, an open manifold is defined as a non-compact manifold without a boundary. The Euclidean space of \mathbb{R}^n , for example, is an open manifold, which topologically does not wrap around.

Due to the compactness of the $SO(3)$ manifold, there exist configurations of pairs of points where multiple, nonunique geodesically minimal paths exist between them; for instance, there are two unique geodesically minimal paths for a pair of antipodal

points on a circle, and there are infinitely many for a pair of antipodal points on a sphere. This is not the case in an open manifold like the 3D Euclidean space of \mathbb{R}^3 , over which there exists a unique geodesically minimal path between any arbitrary pair of points. The distinction in compactness between the 3D rotational space of $SO(3)$ and the 3D Euclidean space makes optimization over $SO(3)$ more ill-conditioned than over the space of \mathbb{R}^3 . This results in the optimization over the rotational space being non-convex. Specifically, as shown by Yau [93], there exists no non-trivial continuous convex function over compact Riemannian manifolds.

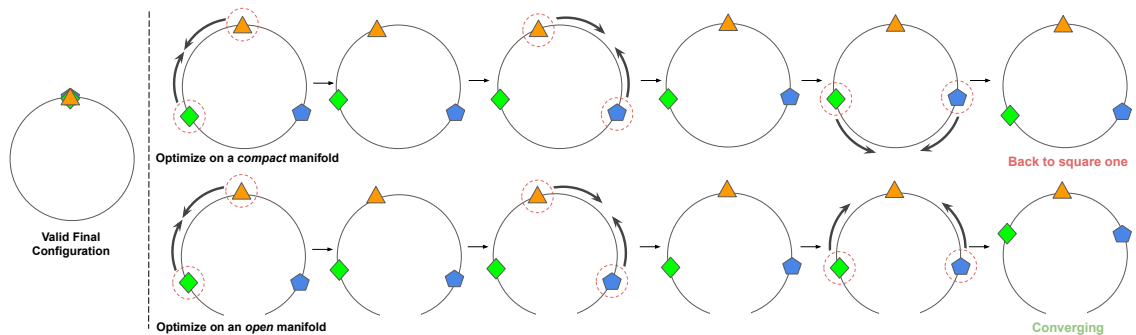


Figure 2.2: **A simple illustration on the effects of compact vs open manifold on the convergence behaviour of optimization algorithm.** Given a 2D circle, and 3 points that exist on the 2D circle where their true values should overlap, (i.e., their ground truth values represent the same point on the 2D circle), the left shows the valid final configuration. Now assign random position on the circle to the three points as initialization, assuming we have access to the underlying ground truth relative geodesic distance between all pairs of points, which in this case, are all 0, we would like to make pairwise updates to minimize the difference between their current relative geodesic distance and their ground truth relative geodesic distance (0). To execute each update, we take a pair of points, and move both points in the direction that minimize their relative geodesic distance, and repeatedly do this pairwise update for all pairs, until convergence or if maximum number of steps is reached. The first row illustrates the configurations of doing pairwise updates for all pairs of points on a compact manifold of a closed circle, and the second row shows the exact same update procedure but applied on an open manifold of a curved line. In this case, we see that optimization on the compact manifold results in failure in convergence, while optimization on an open manifold results in progress in convergence.

Figure 2.2 gives an intuitive demonstration of the problem with optimization over compact space, showing the different converging behaviours of a simple three-point averaging problem over a compact 2D circle, vs. over an open 2D curve. These properties of the $SO(3)$ manifold will affect the convergence of self-supervised orientation estimation, which we discuss below.

2.5 Baselines

While self-supervised learning for object translation, specifically in the form of object keypoints [50, 71, 72], has shown great success, in this work, we show that naively applying such an iterative self-supervised formulation to the rotational group $SO(3)$ will often fail to converge. Below we discuss two approaches to self-supervised orientation estimation in $SO(3)$.

2.5.1 Quaternion Averaging

A standard objective in rotation estimation is to minimize the geodesic distance between a predicted unit quaternion and its corresponding ground-truth orientation [32, 48], $\theta = \arccos(2\langle \hat{\mathbf{q}}_i, \mathbf{q}_{gt} \rangle^2)$ where \hat{q}_i is the predicted orientation in unit quaternion for image i and q_{gt} is the ground-truth orientation. An objective function is often defined to directly minimize this geodesic distance [48].

In our task, defined above (Section 2.3), we are given the relative rotation q_i^j between some pairs of rotations q_i and q_j . Using this relative supervision, we can use the geodesic distance between a sample estimate, \hat{q}_i , and its desired relative position, with respect to a sampled neighbor and a known relative rotation q_i^j , $\tilde{q}_i = q_i^j \otimes \hat{q}_j$, leading to the loss $\mathcal{L}_q = 1 - \langle \hat{q}_i, q_i^j \otimes \hat{q}_j \rangle^2$, where \otimes denotes the quaternion multiplication. Note that this loss is monotonically related to the geodesic distance when using unit quaternions, while avoiding the need to compute an arccos.

2.5.2 $SO(3)$ Averaging

To optimize the rotations with respect to the non-Euclidean geometry of the rotational manifold of $SO(3)$, one approach is described by Manton [49]. Each orientation is iteratively updated in the tangent space using the logmap of $SO(3)$ and projected back to $SO(3)$ using the exponential map. Specifically, we can take the gradient of the loss

$$\mathcal{L}_{SO(3)} = \left\| \log (R_i^\top R_i^j R_j) \right\|^2 \quad (2.18)$$

$$\nabla_{\hat{r}_i} \mathcal{L}_{SO(3)} = r_{\Delta} = \log (R_i^{\top} R_i^j R_j) \quad (2.19)$$

which gives the update step $\hat{R}_i \leftarrow \hat{R}_i \exp(\gamma r_{\Delta})$, where γ is the learning rate and \log is the logmap of $SO(3)$ as detailed in Sec 2.4.3 . When optimizing the full set of orientations, this algorithm can fall into local optima due to the closed nature of the space which allows any orientation to be reached by two unique straight paths, as the space wraps around on itself.

2.6 Method

We propose an alternative that projects the optimization to an open image and optimizes the distances in that space. Specifically, we use the Modified Rodriguez Projection to minimize the relative error between neighboring poses in \mathbb{R}^3 . We provide experiments in Section 2.8 that show that self-supervised orientation estimation using Modified Rodriguez Projection converges much faster than self-supervised orientation estimation in $SO(3)$, with theoretic analysis of an illustrative example demonstrated in Section 2.7.

2.6.1 Iterative Modified Rodrigues Projective Averaging

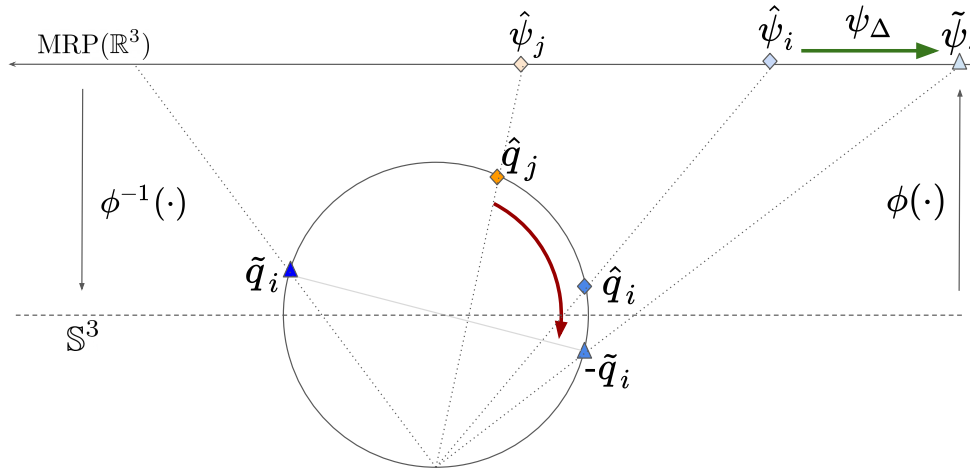


Figure 2.3: **Illustration of Iterative Modified Rodrigues Projective Averaging.** Projection of relative supervision, q_i^j , shown in red, from back-projected rotation $\hat{q}_j := \phi^{-1}(\hat{\psi}_j)$ to \hat{q}_i into the MRP space update, ϕ_Δ , shown in green. While \tilde{q}_i could have been selected as the the goal rotation, it would have induced a much larger movement in the projected space.

As mentioned previously, optimizing on a closed space, such as $SO(3)$ or \mathbb{S}^3 can be problematic, since the relative distance between two points can eventually be minimized by moving them in the exact opposite direction of the minimum path between them. To alleviate this issue, we would like to instead perform self-supervised learning in an open space, where this symmetry is broken. This can be done using Modified Rodrigues Parameters (MRP) [74, 88]. MRP is the stereographic projection of the closed manifold of the quaternion sphere \mathbb{S}^3 to \mathbb{R}^3 , and has been widely used in attitude estimation and control [19]. In combining this projection with the mapping between $SO(3)$ and \mathbb{S}^3 , this projection can be used to optimize rotations. We define a unit quaternion $q = \begin{bmatrix} \rho & \nu \end{bmatrix} \in \mathbb{S}^3 \triangleq \{x \in \mathbb{R}^4 : \|x\| = 1\}$, where $\rho \in \mathbb{R}$ defines the scalar component and $\nu \in \mathbb{R}^3$ defines the imaginary vector component of the unit quaternion. The projection operator $\phi(q) = \psi \in \mathbb{R}^3$ and its inverse $\phi^{-1}(\psi) = q \in \mathbb{S}^3$ are given by [74, 88] where $\psi = \phi \left(\begin{bmatrix} \rho & \nu \end{bmatrix} \right) = \frac{\nu}{1+\rho}$ and $\begin{bmatrix} \rho & \nu \end{bmatrix} = \phi^{-1}(\psi) = \begin{bmatrix} \frac{1-\|\psi\|^2}{1+\|\psi\|^2} & \frac{2\psi}{1+\|\psi\|^2} \end{bmatrix}$.

2. Deep Projective Rotation Estimation through Relative Supervision

Given this projective orientation space, we need to map our relative rotation R_i^j into the projective space in order to use these relative rotations for the self-supervised learning task. This projection is required, as the relative supervision is in $SO(3)$, and the direction and magnitude of this relative measurement are distorted differently in different regions of the projective MRP space. Given a pair of estimated projected rotations $\hat{\psi}_i := \phi(\hat{R}_i)$ and $\hat{\psi}_j := \phi(\hat{R}_j)$, we project $\hat{\psi}_j$ back to a unit quaternion $\phi^{-1}(\hat{\psi}_j) = \hat{q}_j \in \mathbb{S}^3$ and rotate it according to R_i^j , $\tilde{q}_i = q_i^j \otimes \hat{q}_j$, where \otimes is quaternion multiplication and q_i^j is the quaternion form of R_i^j . The resulting unit quaternion \tilde{q}_i is then projected back into the Modified Rodrigues Parameter space, $\tilde{\psi}_i$. A simplified visual analogy of this process is shown in Figure 2.3¹.

While this relative rotation could be applied and projected at either the sampled point $\hat{\psi}_i$, or the neighboring location $\hat{\psi}_j$, we select the neighboring location $\hat{\psi}_j$, as it does not require us to compute gradients through the forward or inverse projections $\phi(\cdot)$ and $\phi^{-1}(\cdot)$, respectively. This projected rotation $\tilde{\psi}_i$ represents the value $\hat{\psi}_i$ should hold, relative to the current predicted rotation $\hat{\psi}_j$. It should be noted that $\psi(q) \neq \psi(-q)$, while q and $-q$ represent the same rotation. In terms of the projective space, this means that the sign of \tilde{q}_i matters. To remove this ambiguity, we select the nearest projection to $\hat{\psi}_i$ in the projective MRP space. It should be noted that this is different from selecting the closer antipode on \mathbb{S}^3 , as the large deformations found near the south pole² can cause the nearer antipode in \mathbb{S}^3 to be further in MRP space. In contrast, if we were to select a consistent sign for the scalar component \tilde{q}_i , for example ensuring the scalar component is always positive, a small change in $\hat{\psi}_j$ can cause large changes in $\tilde{\psi}_i$. While this change is required to stabilize our optimization, it does add some ambiguity to the direction of optimization. However, the directions to each of the projected locations, $\psi(\tilde{q}_i)$ and $\psi(-\tilde{q}_i)$, are only anti-parallel (pulling in exactly opposite directions) when $\tilde{\psi}_i - \hat{\psi}_i$ intersects the origin.

The loss with respect to a given estimate, $\hat{\psi}_i$, can then be written as the l_2 distance between its current value and the projected relative location, $\tilde{\psi}_i$, relative to a given neighbor, $\hat{\psi}_j$:

¹Note that for the purpose of visualization clarity, we have lifted the projection plane from the equator of the \mathbb{S}^3 unit sphere to above the unit sphere.

²The south pole in this case is described by the quaternion $-1 + 0i + 0j + 0k$

$$\mathcal{L}_{\Psi_+} = \left\| \hat{\psi}_i - \phi(\tilde{q}_i) \right\|^2 \quad (2.20)$$

$$\mathcal{L}_{\Psi_-} = \left\| \hat{\psi}_i - \phi(-\tilde{q}_i) \right\|^2 \quad (2.21)$$

$$\mathcal{L}_{\Psi} = \min(\mathcal{L}_{\Psi_-}, \mathcal{L}_{\Psi_+}) \quad (2.22)$$

where we recall that, $\tilde{q}_i = q_i^j \otimes \hat{q}_j$, and $\hat{q}_j = \phi^{-1}(\psi_j)$.

Note that, while $\hat{\psi}_j$ is a predicted value, we do not pass gradients through it, allowing it to anchor the update to a consistent orientation. The gradient update³ is then given by:

$$\nabla_{\hat{\psi}_i} \mathcal{L}_{\Psi} = \psi_{\Delta} = \begin{cases} \hat{\psi}_i - \phi(\tilde{q}_i), & \text{if } \mathcal{L}_{\Psi_+} < \mathcal{L}_{\Psi_-} \\ \hat{\psi}_i - \phi(-\tilde{q}_i), & \text{otherwise} \end{cases} \quad (2.23)$$

Additionally, a maximum gradient step, η , in the projective space is imposed, if the gradient exceeds a defined amount, as shown in Eq. 2.24.

$$\psi_{\Delta} \leftarrow \eta \frac{\psi_{\Delta}}{\|\psi_{\Delta}\|} \quad (2.24)$$

This prevents extremely large steps from being taken, as the projective transform can distort the space.

The full method for Iterative Modified Rodrigues Projective Averaging is shown in Algorithm 1. In practice, we find $\gamma = 0.5$ and $\eta = 0.1$ to produce the best results.

2.7 Intuitive Example

We present an intuitive example to demonstrate when optimizing a set of orientations to solve the rotation averaging problem described in Equation (1) can fail. In this example, we show the benefits of the Iterative Modified Rodrigues Projective Averaging (**MRP(Ours)**) approach over the baseline approach. We show that

³We omit a constant factor for brevity, and integrate it into the learning rate, γ .

Algorithm 1 Iterative Modified Rodrigues Projective Averaging

Require: Initial estimates $\hat{\Psi} = \{\hat{\psi}_1 \dots \hat{\psi}_N\}$
Require: Local neighborhood \mathcal{N}_i for each rotations $\hat{\psi}_i$
Require: Relative rotations q_i^j for each $j \in \mathcal{N}_i$
Require: Step size γ
Require: Max gradient threshold η
Require: Max iteration steps M
Require: Global pairwise angular error $\varepsilon \leftarrow 0$
Require: Global pairwise angular error threshold ε_0

- 1: **while** $\varepsilon \geq \varepsilon_0$ or *iteration* $\leq M$ **do**
- 2: Sample a projected rotation $\hat{\psi}_i \sim \hat{\Psi}$ to update
- 3: Sample a neighbor $\hat{\psi}_j \sim \mathcal{N}_i$
- 4: Update the projected rotation ψ_Δ (Eq. 2.23)
- 5: **if** *the magnitude of the update is larger than η* **then**
- 6: Resize update to be of size η (Eq. 2.24)
- 7: **end if**
- 8: Apply update in MRP space $\hat{\psi}_i \leftarrow \hat{\psi}_i + \gamma\psi_\Delta$
- 9: Update global pairwise angular error ε
- 10: **end while**
- 11: **return** $\hat{\Psi}$

while both $SO(3)$ averaging and **MRP(Ours)** share a class of nonoptimal critical points, in the projective case, these critical points are a subset of the problematic configurations for $SO(3)$ averaging. Before diving into this, see Figure 2.2 for a simplified pictorial illustration that demonstrates the intuition of how **MRP(Ours)** reduces the dimension of critical points through stereographic projection.

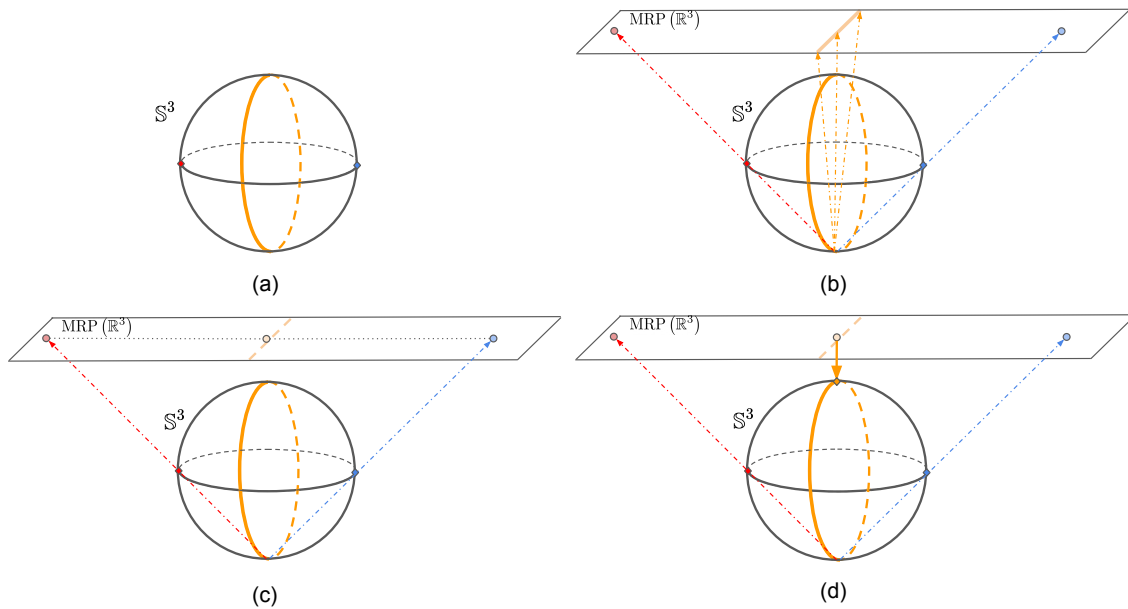


Figure 2.4: **An illustration of the intuition behind how MRP reduces the dimension of the set of problematic local optima through stereographic projection.** (a) Take two orientation quaternions in \mathbb{S}^3 , shown in red and blue. We see that all points along the orange equator are valid averages of these two quaternions in \mathbb{S}^3 ; (b) We can reduce the dimension of the set of valid averages by using MRP to project these orientations into the hyperplane of \mathbb{R}^3 ; (c) And instead, take the average of the two projected quaternion as seen in red and blue circles, which gives us the average as the orange circle in \mathbb{R}^3 ; (d) The average in the projected space can be projected back to the original orientation space when needed. Thus we have successfully *reduced* the dimension of the set of problematic local optima from an equator of points to one point on the original manifold of 3D rotations (in this specific case, the unit quaternion sphere).

2.7.1 Examples of Critical Points

In this section, we analyze a class of critical points shared by both standard $SO(3)$ averaging and Iterative Modified Rodrigues Projective Averaging. For simplicity, we will examine the $N = 3$ rotation case, where $\mathcal{R} = \{R_1, R_2, R_3\}$ with relative rotations of $R_i^j := R_i R_j^\top$. As this is an iterative algorithm, we need to initialize our predicted rotations to some values $\hat{\mathcal{R}} = \{\hat{R}_1, \hat{R}_2, \hat{R}_3\}$. In this case, we initialize each predictions to $\hat{R}_i := R_i R_0 \exp\left(\left(\theta_0 + i\frac{2\pi}{N}\right)\omega_0\right)$ where R_0 is an arbitrary but constant rotational offset, ω_0 and θ_0 define an arbitrary, but constant axis and constant rotation, about which each initial estimate \hat{R}_i is rotated an additional angle of θ_i . We find that if we use the previously described methods to update this initial configuration, under

certain values of \mathcal{R} , R_0 , θ_0 , and ω_0 , the expected update at each value \hat{R}_i is $\mathbf{0}$, forming a critical point for each algorithm.

Critical Point for $SO(3)$ Averaging

Given the initial predictions of $\hat{\mathcal{R}}$ defined above, for all values of \mathcal{R} , R_0 , θ_0 , and ω_0 , we find that the expectation of the gradient of $SO(3)$ averaging loss, $\mathbb{E}_{i,j} [\nabla_{\hat{r}_i} \mathcal{L}_{SO(3)}]$, is $\mathbf{0}$. The gradient of any sampled pair i, j is given by

$$\begin{aligned} \nabla_i \mathcal{L}_{SO(3)}^{i,j} &:= \nabla_{\hat{r}_i} \mathcal{L}_{SO(3)} \left(\hat{R}_i, \hat{R}_j, R_i^j \right) \\ &= \log \left(\hat{R}_i^\top R_i^j \hat{R}_j \right) \\ &= \log \left((R_i R_0 \exp(\theta_i \omega_0))^\top R_i^j R_j R_0 \exp(\theta_j \omega_0) \right) \\ &= \log \left(\exp((\theta_j - \theta_i) \omega_0) \right) \\ &= \text{wrap}_{[-\pi, \pi]} [(\theta_j - \theta_i) \omega_0] \\ &= \text{wrap}_{[-\pi, \pi]} \left[\frac{2\pi}{N} (j - i) \right] \omega_0 \\ &= \frac{2\pi}{N} (j - i) \omega_0. \end{aligned}$$

This lead to an expected gradient of each estimate rotation \hat{R}_i of

$$\mathbb{E}_j \left[\nabla_{\hat{r}_i} \mathcal{L}_{SO(3)} \left(\hat{R}_i, \hat{R}_j, R_i^j \right) \middle| i = 1 \right] = \frac{1}{2} \text{wrap}_{[-\pi, \pi]} \left[\sum_{j \neq i} \frac{2\pi}{N} (j - i) \right] \omega_0 = \mathbf{0}.$$

For all estimates \hat{R}_i , this sums to an integer multiple of $2\pi\omega_0$, which, due to the definition of the $SO(3)$ exponential map, wraps to $\mathbf{0}$.

Critical Point for Iterative Modified Rodrigues Projective Averaging

When optimizing using our Iterative Modified Rodrigues Projective Averaging method, we find that this configuration is only a critical point when the relative orientations between each pair of rotations are equal and opposite, i.e., $R_i^j = R_i^{k\top} \rightarrow R_i^j = \exp(\pm \frac{2\pi}{N} \omega_0)$ and the predicted orientations are initialized at identity: $R_0 = \mathbf{I}$. This

only happens when the true orientations \mathcal{R} are evenly spaced about an axis of rotations: $R_i := \exp\left(\left(\theta_0 - i\frac{2\pi}{N}\right)\omega_0\right)$, leaving only axis of rotation ω_0 and the constant angular offset θ_0 about that axis as free parameters.

As we are trying to update these rotations using a method compatible with stochastic gradient descent, we are concerned with the expectation of our update with respect to a sampled pair. In this case, the expected loss and update, defined in Equations (3c) (4) in the main text, respectively, for any projected rotation $\hat{\psi}_i$ and its neighbor $\hat{\psi}_j$ is $\mathcal{L}_{\Psi^+}^{i,j} := \left\| \hat{\psi}_i - \phi(q_i^j \otimes \phi^{-1}(\hat{\psi}_j)) \right\|^2$ where q_i^j is the quaternion associated with R_i^j . As all $\hat{\psi}_i$ are initialized to the identity, i.e., $\phi(q_I) = \mathbf{0}$ where q_I is the identity quaternion, we get

$$\begin{aligned} \mathcal{L}_{\Psi^+}^{i,j} &:= \left\| -\phi^{-1}(q_i^j) \right\|^2 & \nabla_i \mathcal{L}_{\Psi^+}^{i,j} &:= -\phi^{-1}(q_i^j) \\ \mathcal{L}_{\Psi^-}^{i,j} &:= \left\| -\phi^{-1}(-q_i^j) \right\|^2 & \nabla_i \mathcal{L}_{\Psi^-}^{i,j} &:= -\phi^{-1}(-q_i^j) \end{aligned}$$

The relative rotations in this configuration are

$$R_i^j := \exp\left(\pm \frac{2\pi}{3}\omega_0\right)$$

with relative quaternions $q_i^j := \left[\cos\left(\frac{\pi}{3}\right) \pm \sin\left(\frac{\pi}{3}\right)\omega_0 \right]$, which leads to

$$\phi(q_i^j) = \frac{\pm \sin\left(\frac{\pi}{3}\right)\omega_0}{1 + \cos\left(\frac{\pi}{3}\right)} = \frac{\pm\omega_0}{\sqrt{3}} \quad \phi(-q_i^j) = \frac{\mp \sin\left(\frac{\pi}{3}\right)\omega_0}{1 - \cos\left(\frac{\pi}{3}\right)} = \pm\sqrt{3}\omega_0.$$

This results in the potential losses for the positive and negative antipodes of

$$\mathcal{L}_{\Psi^+}^{i,j} = \|\phi(q_i^j)\| = \frac{1}{3} \quad \mathcal{L}_{\Psi^-}^{i,j} = \|\phi(-q_i^j)\| = 3$$

for all pairs of (i, j) *todo*. Selecting the minimum loss antipodes, we get gradients of

$$\nabla_i \mathcal{L}_{\Psi^+}^{i,j} = \frac{\mp 1}{\sqrt{3}}\omega_0 \quad \nabla_i \mathcal{L}_{\Psi^-}^{i,j} = \frac{\pm 1}{\sqrt{3}}\omega_0,$$

for $j = i + 1$ and $j = i - 1$, respectively. The final expectation of the gradients with respect to the neighborhood sampling is

$$\mathbb{E}_j \left[\nabla_{\hat{\psi}_i} \mathcal{L}_{SO(3)}(\hat{\psi}_i, \hat{\psi}_j, R_i^j) | i = 1 \right] = \frac{1}{2} \sum_{j \neq i} \nabla_i \mathcal{L}_{\Psi}^{i,j} = \frac{1}{2} \left(\frac{1}{\sqrt{3}}\omega_0 - \frac{1}{\sqrt{3}}\omega_0 \right) = \mathbf{0}.$$

While this demonstrates that our method is not without critical points, even in this simple example, it shows that this configuration is only problematic when the true rotations are equally spaced around an axis of rotation, ω_0 , and the estimates are

2. Deep Projective Rotation Estimation through Relative Supervision

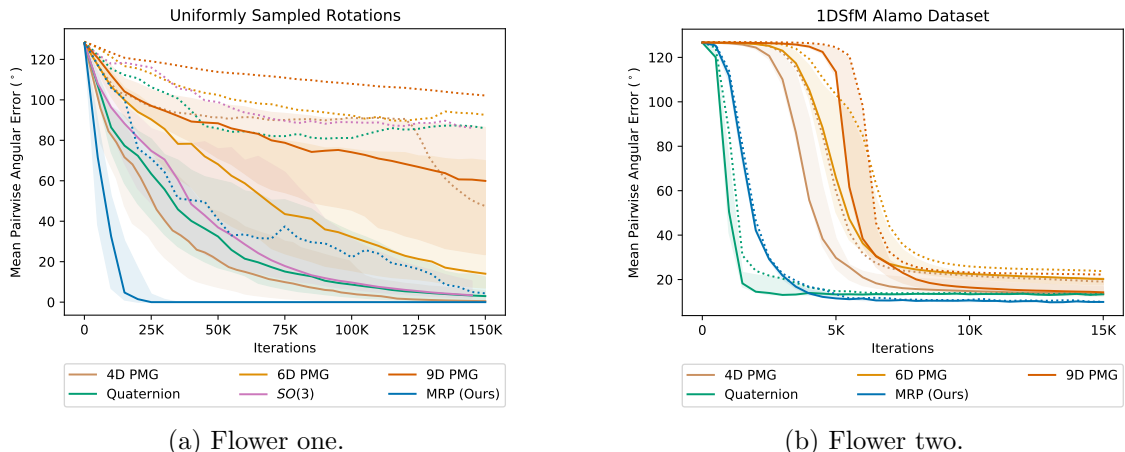


Figure 2.5: **Relative rotation consensus results for direct parameter optimization.** Relative rotation consensus with direct optimization of rotation parameters over 50 unique environments with 100 random generated orientations each (**left**) and Alamo 1DSfM [89] (**right**). Median average-pair-wise angular error ($^{\circ}$) between each estimated rotations is shown, with shaded region representing the first and third quartile for each method. The max average-pair-wise angular error for each algorithm at each iteration is shown as a dashed line.

initialized at identity. This compares very favorably to the $SO(3)$ algorithm, which can be in a critical point for any set of relative rotations, R_i^j , and with initialization that can vary with an additional arbitrary constant rotation R_0 .

2.8 Experiments

Next, we perform experiments to show that our method converges faster and more consistently than the alternative approaches. Our empirical results are grouped into two settings: (1) direct optimization of randomly generated rotations, Section 2.8.1, and (2) optimization of the parameters of a convolutional neural network using synthetically rendered images, Section 2.8.2. In both cases, relative orientations between elements in a neighborhood are provided. We show Iterative Modified Rodrigues Projective Averaging is able to converge faster and more often than alternative approaches. We further show in Section 2.8.2 that our method can easily be used to supervise convolutional neural networks, when only relative orientation information is available.

2. Deep Projective Rotation Estimation through Relative Supervision

Algorithm	<i>Avg Pairwise Angular Error < 5°</i>			<i>Normalized AUC</i>		
	Mean Steps	Max Steps	Min Steps	Mean	Max	Min
$SO(3)$	157.7K	Not Converged	85.0K	24.47	82.92	7.55
4D PMG [17]	126.1K	Not Converged	27.0K	15.67	52.40	3.06
6D PMG [97]	235.9K	Not Converged	80.0K	43.53	89.15	11.34
9D PMG [44]	284.5K	Not Converged	150.0K	62.94	101.77	17.77
Quaternion	160.3K	Not Converged	40.0K	23.55	84.85	3.47
MRP (Ours)	37.5K	160.0K	15.0K	5.08	15.56	2.18

Table 2.1: Number of iteration steps until convergence and Normalized Area Under Curve (nAUC) over 50 unique environments of 100 randomly generated orientations. 300K optimization steps are taken for each experiment.

Algorithm	<i>% Avg Pairwise Angular Error < 5°</i>					<i>Final Error(°)</i>	
	30K	70K	100K	150K	300K	Mean	Median
$SO(3)$	0%	0%	6%	57%	94%	2.056	0.10
4D PMG [17]	2%	32%	46%	72%	90%	1.969	0.14
6D PMG [97]	0%	0%	4%	20%	52%	20.096	3.20
9D PMG [44]	0%	0%	0%	2%	20%	40.125	43.02
Quaternion	0%	12%	30%	56%	82%	9.72	0.04
MRP (Ours)	66%	88%	96%	98%	100%	0.004	0.004

Table 2.2: Percentage of experiments converged and final angular errors over 50 unique environments of 100 randomly generated orientations. 300K optimization steps are taken for each experiment.

Algorithm	<i>Mean Relative Error (°)</i>		<i>Mean Absolute Error (°)</i>		<i>Mean nAUC</i>	
	E. Island	Alamo	E. Island	Alamo	E. Island	Alamo
	4D PGM [17]	11.94	15.00	7.34	9.94	25.60
6D PGM [97]	11.26	18.84	6.90	13.09	27.77	58.04
9D PGM [44]	10.22	16.32	6.32	11.43	29.31	60.14
Quaternion	11.58	13.40	7.23	8.93	16.01	22.57
MRP (Ours)	8.84	9.89	5.49	6.56	16.21	25.61

Table 2.3: Final results on 1DSfM [89] datasets after 20K iterations

2.8.1 Direct Parameter Optimization

We evaluate the convergence behaviour of our Iterative Modified Rodrigues Projective Averaging method, **MRP (Ours)**, described in Section 2.6.1, as well as the $SO(3)$ averaging method, described in Section 2.5.2. For the $SO(3)$ averaging method, we implement both the pure Riemannian optimization, **SO(3)**, as well as a method using a Projective Manifold Gradient Layer [17] to map the Riemannian gradient of the $SO(3)$ averaging loss, Equation 2.18, to a Euclidean optimization in \mathbb{R}^D , where we test $D = 4$, $D = 6$ [97], and $D = 9$ [44], **4D PMG** [17], **6D PMG** [97], **9D PMG** [44], respectively. Additionally, we evaluate direct quaternion optimization, described in Sections 2.5.1, **Quaternion**.

Uniformly Sampled Rotations

We test the performance of each algorithm when directly optimizing the rotation parameters of a set of size $N = 100$ with known relative rotations R_i^j , and local neighborhood structure. Ground truth and initial estimated rotations are both randomly sampled from a uniform distribution in $SO(3)$. Each rotation, R_i , has a neighborhood, \mathcal{N}_i , consisting of the closest $|\mathcal{N}_i| = 3$ rotations with respect to geodesic distance. The connectivity of this neighborhood graph is checked to ensure the graph contains only a single connected component. We test all algorithms over 50 sets of unique environments, each with $N = 100$ randomly generated orientations as described above. The estimated rotations are updated by each algorithm in batches of size 8, for 300K iterations.

As the goal of our algorithm is to improve the convergence properties of iterative averaging methods, we analyze each algorithm at various stages of optimization. We are particularly interested in the average number of update steps until the algorithm has converged, which we define as when the average angular error between all pairs of rotations is below 5° . As we can see in Figure 2.5, the Iterative Modified Rodrigues Projective Averaging method, **MRP (Ours)**, converges before the standard $SO(3)$ averaging method. On average, our method converged to within 5° in 37K steps. The next best method, **4D PMG** [17], which takes over three times as many iterations to converge to the same level of accuracy. Further, Table 2.1 shows that our method is the only one to converge across all environments within 300K iterations. For each

2. Deep Projective Rotation Estimation through Relative Supervision

Algorithm	Mean	Median	5° Acc (%)
	Error (°)	Error (°)	
4D PMG [17]	123.84	123.96	0
Quaternion	28.83	21.74	50
MRP (Ours)	3.71	3.73	100
Oracle	1.58	1.56	100

Table 2.4: Final results for image based rotation estimation.

method, we also compute the mean area under the pairwise error curve, with the number of steps normalized to between zero and one (nAUC), also shown in Table 2.1. We find that in the best-, average-, and worst-case scenarios, our method has the best convergence behavior. We find that at each stage of training, the Iterative Modified Rodrigues Projective Averaging, **MRP (Ours)**, training has both a lower average pairwise error, shown in Table 2.2. Our method converged far more often at each stage of training, also shown in Table 2.2.

Structure from Motion Dataset

To test our algorithms under natural noise conditions, we also evaluate our algorithm on the 1DSfM [89] structure from motions datasets. These datasets contain full transforms for each sample; however, we are only concerned with optimizing the rotations. Each environment is tested with 5 random initializations and the estimated rotations are updated by each algorithm in batches of size 64, for 20K iterations. The results of a subset of the environments are shown in Table 2.3 and the remainder can be found in Appendix A.1. The noise characteristics of relative rotations in this dataset are similar to those found when capturing relative poses, but, unlike the environments found in the previous section (Uniformly Sampled Rotations), the distribution of rotations does not fully cover the orientation space. Specifically, the distribution of rotations mostly only vary in yaw, with limited variation in pitch and roll. As a result, all methods converge relatively quickly. Our algorithm outperforms the baselines in terms of accuracy. While the **Quaternion** optimization converges slightly faster, it consistently finds a lower accuracy configuration, resulting in a low nAUC, but higher relative and absolute accuracy.

2.8.2 Neural Network Optimization

To show that the Iterative Modified Rodrigues Projective Averaging method, **MRP (Ours)**, can be used to learn object orientation from images using neural networks by optimizing the parameters of a simple CNN, specifically a ResNet18 [33], we follow the procedure as in Section 2.8.1 with some minor changes. Instead of operating directly on a set of rotation parameters, we learn a function $\hat{R}_i = f(I_i)$ from rendered images of the YCB drill [92] model, shown in Figure 2.9, rendered at each of 100 random rotations R_i . We continue to only supervise each method described in Section 2.8.1 using the relative rotations between each image. For different rotation parameterizations compared to Section 2.8.1, we modify the output of the neural net function so that the predicted orientation is represented by the rotation parameterization of which the method is compared. For example, for **MRP (Ours)**, we learn a function $\hat{\psi}_i = f(I_i)$ where $\hat{\psi}_i$ is the predicted modified Rodrigues parameters. The neural network optimization experiment setup for image-based orientation estimation is described in Figure 2.6.

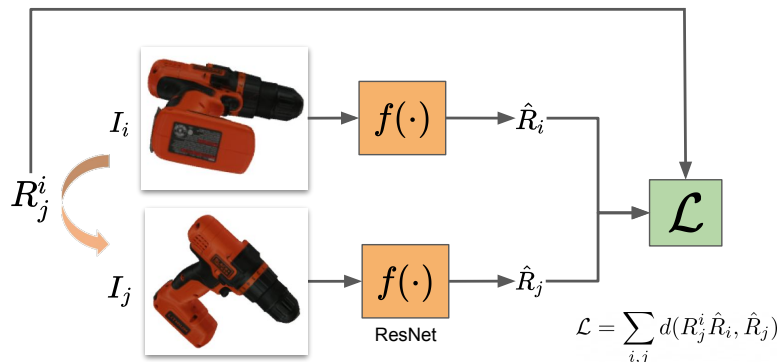


Figure 2.6: **Experiment setup for image-based orientation estimation using neural network optimization.** Given the YCB drill model, the model is randomly rendered at two orientations, producing a pair of images, (I_i, I_j) , with ground truth relative rotation between them as R_j^i . We then learn a function $f(\cdot) : \mathcal{I} \rightarrow SO(3)$ parameterized by a ResNet18 model that takes in an image of an object, I_i , and outputs an absolute orientation prediction \hat{R}_i such that the geodesic distance between the ground truth relative rotation R_j^i and the relative rotation induced from the pair of orientation prediction (\hat{R}_i, \hat{R}_j) is minimized.

We compare the best performing methods, and, as a lower bound, we also train an oracle network, **Oracle**, which is trained to output absolute rotations by regressing

with the ground truth absolute orientations, R_i and optimized with the cosine quaternion loss. We use the Adam [42] optimizer, batch size of 32 and learning rate of 1×10^{-4} for all experiments, and with a maximum training time of 10K steps, trained over 8 environments, each with 100 images associated with randomly rendered rotations. We find that **MRP (Ours)** is capable of converging to a rotational frame consistent with the relative rotations used for supervision relatively quickly, with a significantly lower average pairwise error than all other relative methods, shown in Figure 2.9 and Table 2.4.

Curriculum Training for Generalization to Test Set

At test time, we find that a curriculum is required for any relatively supervised method to be generalized to unseen orientation pair. This curriculum training involves starting with an initial base rotation. The model is rendered at this base rotation and a random rotation within $\alpha = 30^\circ$ of this base rotation. This base rotation is initially sampled with $\theta = 30^\circ$ of a constant anchor orientation, until the average training angular error of the previous epoch drops below a given threshold, in this case, 5° . Once the error falls below this threshold, the angular range θ , from which this base rotation is sampled, increases by 5° . This process is repeated, increasing the value of θ by 5° each time the training error threshold is reached until the angle of the curriculum increases to 180° and concluding the training process with curriculum, as shown in Figure 2.7

We find that **MRP (Ours)** is capable of completing the curriculum in a reasonable number of iterations, approximately 100K, achieving a median final pairwise accuracy of 5.19° in three training sessions. This test error calculated based on sampling from two random rotations across the space of $SO(3)$, different from the training error, which is calculated by sampling based on the angle range of the curriculum and is always, at most, 30° apart. The quaternion optimization method, **Quaternion**, stalls out at curriculum angle of 90° , achieving a final pairwise accuracy of 12.41° and the **4D PMG** [17] method never gets past the first level of the curriculum, with a final error of 125.09° . The full training progression of each method, over three random initialization each, can be seen in Figure 2.8

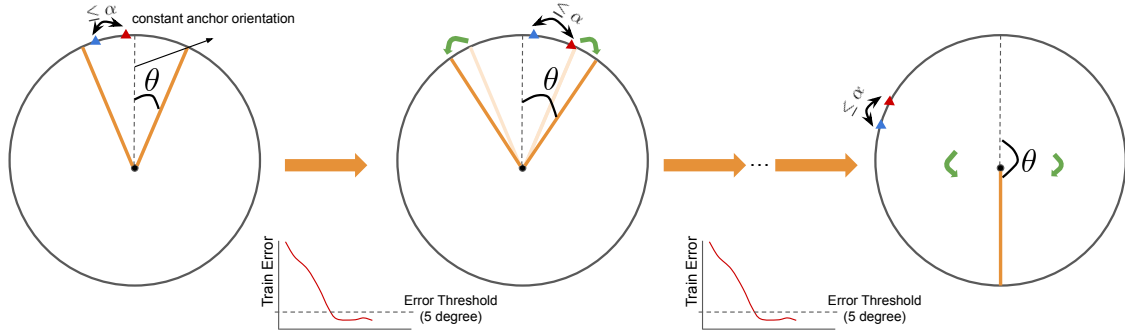


Figure 2.7: **Illustration of curriculum used for training.** With a constant anchor orientation denoted by the dashed line, the curriculum range is defined as the orientation range which is θ degree away from this constant anchor orientation. In the figure, this curriculum range is defined by the area enclosed by the orange lines. This curriculum range is increased whenever the training error drops below a given error threshold, until θ reaches 180° . The pair of orientations for training is always selected to be within $\alpha = 30^\circ$ of each other, whilst both located within the curriculum range. For example, the blue and red triangles represent some valid orientation pairs.

2.9 Limitations & Future Directions

While this parameterization of the rotational space is valuable for learning rotations using only relative supervision, it is not without limitations. One of the primary ones is the need for a curriculum for generalizability to unseen relative rotations. Without this, our experiment show that all representations fall into the local optima of outputting a constant orientation. Additionally, in generalization experiments, we are only able to achieve a final error of 5 degrees. This may not be accurate enough for many fine motor tasks, though an additional refinement network that is trained to handle rotations within a sub-region of the whole rotation space could reduce this error.

2.10 Conclusions

In this work, we show that through the use of Modified Rodrigues Parameters, we are able to open the closed manifold of $SO(3)$, improving the convergence behavior of the rotation averaging problem. While optimizing in the projective space of Modified Rodrigues Parameters is still susceptible to a similar class of local optima as the naive method, this, however, is only affected in a subset of these configurations,

2. Deep Projective Rotation Estimation through Relative Supervision

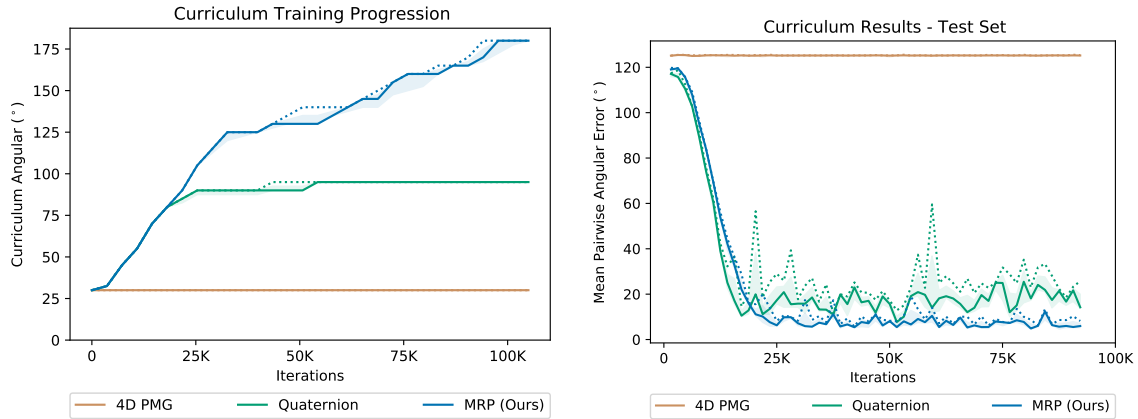
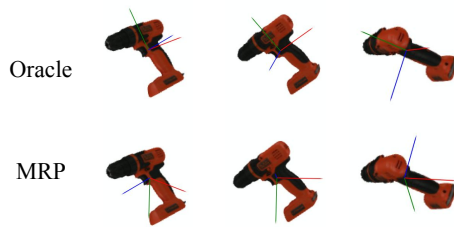


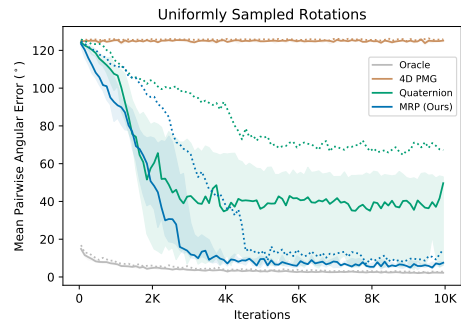
Figure 2.8: **Curriculum Angle (left) and Average Pairwise Error (right)**, sampled over the full orientation space for three training sessions with each method. Median average-pairwise angular error ($^{\circ}$) is shown with shaded areas representing the first and third quartile over all training sessions. The max average-pairwise angular error for each algorithm at each iteration is shown as a dashed line.

greatly reducing the dimensionality of the problematic configurations. We show that Iterative Modified Rodrigues Projective Averaging is able to outperform the naive application of relative-orientation supervision in both direct parameter optimization and image-based rotations estimation from neural networks. We hope our method allows more systems to convert the relative supervision of relative methods, like ICP, to consistent and accurate absolute poses.

2. Deep Projective Rotation Estimation through Relative Supervision



(a) Estimated rotation frame learned for the YCB [92] drill model using Iterative Modified Rodrigues Projective Averaging and relative rotations (x , y , z)



(b) Results for rotations estimated by neural networks given images of the YCB drill [92] rendered at each of 100 random rotations with various supervisions

Figure 2.9: **Neural net optimization results.** Estimated rotation frame learned for the YCB [92] drill model using Iterative Modified Rodrigues Projective Averaging and relative rotations (x , y , z) (left). Results for rotations estimated by neural networks given images of the YCB drill [92] rendered at each of 100 random rotations with various supervisions, (right). Median average-pairwise angular error ($^{\circ}$) is shown with shaded areas representing the first and third quartile over all training sessions. The max average-pairwise angular error for each algorithm at each iteration is shown as a dashed line.

Chapter 3

Task-Specific Cross-Pose Estimation for Robot Manipulation

3.1 Introduction

Many manipulation tasks require that a robot is able to move an object to a location relative to another object. For example, a cooking robot may need to place a lasagna in an oven, place a pot on a stove, place a plate in a microwave, place a mug onto a mug rack, or place a cup onto a shelf. Understanding and placing objects in task-specific locations is a key skill for robots operating in human environments. Further, the robot should be able to generalize to novel objects within the training categories, such as placing new lasagnas into the oven or new mugs onto the mug rack. A common approach in robot learning is to train a policy “end-to-end,” mapping from pixel observations to low-level robot actions. However, end-to-end trained policies cannot easily reason about complex pose relationships such as the ones described above, and they have difficulty generalizing to novel objects.

In contrast, we propose achieving these tasks by learning to reason about an object’s three-dimensional geometry. For the type of tasks defined above, the robot needs to reason about the relationship between key parts on one object with respect to key parts on another object. For example, to place a mug on a mug rack, the robot must reason about the relationship between the pose of the mug handle and

3. Task-Specific Cross-Pose Estimation for Robot Manipulation

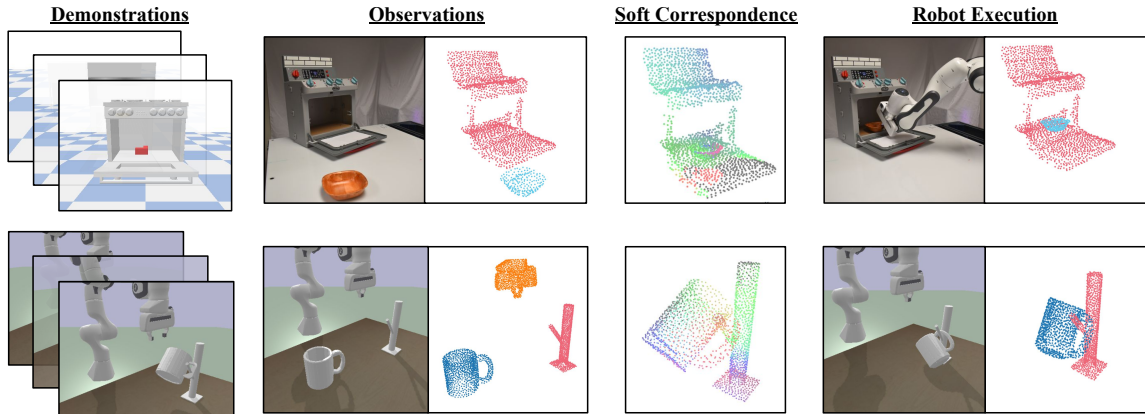


Figure 3.1: **TAX-Pose in action.** **Top:** PartNet-Mobility Placement Task. **Bottom:** Mug Hanging Task. The model is trained using demonstrations of anchor and action objects in their ground-truth cross-pose for the task. The model first observes the initial configuration of the objects, estimates correspondence between the pair of objects, and then calculates the desired pose. The desired pose is then used to guide robot motion planning.

the pose of the mug rack; if the mug rack changes its pose, then the pose of the mug must change accordingly in order to still be placed on the rack (see Figure 3.2). We name this task-specific notion of the pose relationship between a pair of objects as “cross-pose” and we formally define it mathematically. Further, we propose a vision system that can efficiently estimate the cross-pose from a small number of demonstrations of a given task, generalizing to novel objects within the training categories (see Figure 3.1). To achieve the manipulation task, we input the estimated cross-pose into a motion planning algorithm which will manipulate the objects into the desired pose relationship (e.g. placing the mug onto the rack, placing the lasagna into the oven, etc).

In this paper, we present TAX-Pose (Task-specific Cross-Pose), a deep 3D vision-based robotics method that learns to predict a task-specific pose relationship between a pair of objects based on a set of demonstrations. We use this prediction to plan a trajectory that actuates the objects to achieve the desired relative pose. Our cross-pose estimation system is translation equivariant and can generalize from a small number of demonstrations to new objects in unseen poses.

The contributions of this paper include:

1. A precise definition of “cross-pose,” which defines a task-specific pose relationship between two objects.

2. A novel method that estimates the cross-pose between two objects; this method is provably translation equivariant and can learn from a small number of demonstrations.
3. A robot system to manipulate objects into the desired cross-pose needed to achieve a given manipulation task.

We present simulated and real-world experiments to test the performance of our system in achieving a variety of cross-pose manipulation tasks, learning from a small number of demonstrations. We show the generalizability of our model through an object placement task, where the robot must accurately place objects in, on, or around novel objects. We then show the precision of the method, hanging unseen mugs onto a mug rack.

3.2 Related Work

Learning from Demonstration (LfD): LfD is a diverse field of study which focuses on enabling robots to learn skills from expert demonstrations. We refer the readers to previous survey papers [4, 9, 63] for a comprehensive review of LfD approaches. The commonly-used LfD technique is Behavior Cloning (BC) [6, 59], which involves imitating an expert agent given a set of demonstration trajectories by learning to predict the expert’s action in a given state. This simple formulation has proven successful in a variety of tasks, including autonomous driving [10], robotic manipulation [58], and many more. In this project, we use the demonstrations to estimate a “goal pose” and then use motion planning to manipulate the objects into the desired pose.

Object Pose Estimation: Object pose estimation is the task of detecting and inferring the 6DoF pose of an object, which includes its position and orientation. Traditionally, the problem of 6D object pose estimation is tackled by matching feature points between 3D models and images [47, 60]. However, these methods often require rich visual features such as textures for a successful estimation and thus would fail on texture-less objects. To tackle this, RGB-D images from depth cameras have been shown to be able to work on texture-less objects [13, 36, 65], but are sensitive to occlusions in the scene. Recent deep-learning based methods decouple pose estimation

3. Task-Specific Cross-Pose Estimation for Robot Manipulation

into semantic segmentation, translation and rotation estimation components [91], leveraging keypoints [34], or fusing color and geometric information [35].

Although our task is related to that of object pose estimation, we do not aim to estimate the pose of a single object; rather, we define the notion of “cross-pose” which describes a task-specific relationship between a pair of objects, and focus on the problem of predicting the task-specific relative $SE(3)$ transformation between interacting objects, which can be passed into a motion planner to produce a trajectory for the manipulator to follow. There exist many manipulation pipelines that rely on different object representations to tackle this problem of relative transformation prediction. Traditionally, 6D object pose estimators as surveyed before, are used to explicitly predict the 6DoF pose of manipulated objects and at test time, relative transform between objects are obtained by matching the detected object 6DoF pose to the desired object pose. However, training 6D object pose estimators requires large object pose annotations which are resource-intensive to collect; moreover, parameterizing an object pose by a fixed object pose template makes it difficult to generalize to unseen objects within the same object category. More recently, [50, 57] proposed to use 3D semantic keypoints as an alternative form of object representation for manipulation. Specifically, semantic keypoint detectors are trained using large hand-labeled task-specific object keypoints dataset, and at test time, keypoints are predicted on new intra-category object instances. Detected keypoints are then used to recover a local coordinate frame, and subsequently a relative transformation can be extracted by solving a constraint optimization problem. Another line of approach [27, 68] that is highly relevant to our work, instead approach this problem by predicting dense descriptors over the observed object image/point cloud space. At test time, given a set of query points from human task demonstrations, one can find the matching query points on unseen intra-category object instances through descriptor matching, and then extract a relative transformation via orthogonal Procrustes [80]. While this approach requires very few task demonstration to train on and also avoids the need for large hand-annotated 6D pose or semantic keypoint dataset, it is however very sensitive to the location and scale of the sampled query points. In practice, the object region of which these query points are sampled from are hand selected by humans.

Point Cloud Registration: Our method for estimating the cross-pose between

two objects builds upon previous work in point cloud registration. The typical objective in point cloud registration is to find the optimal rigid alignment between two point clouds, to minimize the sum of squared distances between two sets of points [80]. Traditionally, ICP [8] and its variants [1, 12, 29, 37, 61, 66] have been used to compute the optimal rigid alignment between two point clouds. Deep Closest Point (DCP) [85] avoids local minima common for ICP by seeking to approximate correspondence in a high-dimensional learned feature space. Our method builds upon the architecture of DCP for cross-pose estimation; however, in contrast to point cloud registration, in which the objective is to minimize the sum of squared distances between two sets of points, our objective is to estimate a task-specific pose relationship between two different objects.

3.3 Problem Definition

We first define the notion of cross-pose in the context of object placement tasks. Given two objects \mathcal{A} and \mathcal{B} , we define the “relative placement” task of placing object \mathcal{A} at a pose relative to object \mathcal{B} . For example, consider the task of placing a lasagna in an oven, placing a pot on a stove, or placing a mug on a rack, or placing a robot gripper on the rim of a mug. All of these tasks involve placing one object (which we call the “action” object \mathcal{A}) at a semantically meaningful location relative to another object (which we call the “anchor” object \mathcal{B})¹.

In this work, we make the simplifying assumption that, for a given pair of objects \mathcal{A} and \mathcal{B} , there is a single target relative pose needed to achieve a given task. Let $\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}$ be the point clouds of the objects, where $\mathbf{P}_k \in \mathbb{R}^{3 \times N_k}$, and N_k denotes the number of (x, y, z) points in the point cloud of object k . We define the task-specific “cross-pose” between objects \mathcal{A} and \mathcal{B} via the function $f(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}})$ which has the following properties: $f(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}) = \mathbf{I}$ (where \mathbf{I} is the identity) when \mathcal{A} and \mathcal{B} are each in the target pose needed to complete the task (lasagna is in the oven; mug is on the rack, etc). The task-specific cross-pose function f has these further properties:

$$f(\mathbf{T} \cdot \mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}) = \mathbf{T} \cdot f(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}), \quad f(\mathbf{P}_{\mathcal{A}}, \mathbf{T} \cdot \mathbf{P}_{\mathcal{B}}) = f(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}) \cdot \mathbf{T}^{-1} \quad (3.1)$$

¹Note that the definition of action and anchor is symmetric; either object can be treated as the action object and the other as the anchor.

3. Task-Specific Cross-Pose Estimation for Robot Manipulation

where $\mathbf{T} \in SE(3)$ is a pose transformation. We explore the desirable properties of this definition of cross-pose in Appendix A.2.

As a corollary, if $f(\mathbf{P}_A, \mathbf{P}_B) = \mathbf{I}$, then $f(\mathbf{T} \cdot \mathbf{P}_A, \mathbf{T} \cdot \mathbf{P}_B) = \mathbf{I}$. In other words, the target cross-pose is invariant to the reference frame or a global pose transformation. Suppose that our task is to place a mug on a rack in a particular configuration. If we transform (rotate and translate) the mug by transformation \mathbf{T} and we similarly transform the rack also by \mathbf{T} , then the cross-pose between the mug and the rack will be unchanged and the mug will still be “on” the rack, as seen in Figure 3.2.

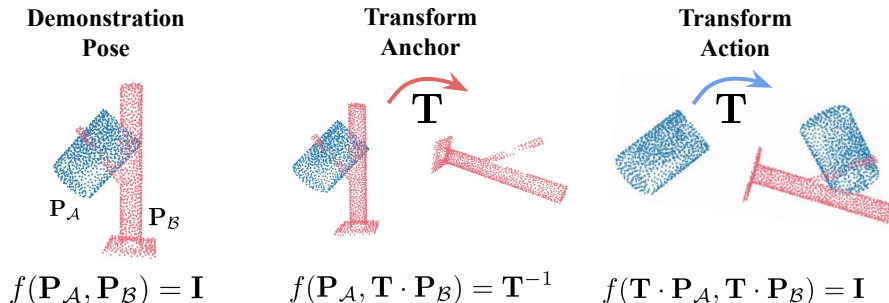


Figure 3.2: **Visualization of the properties of cross-pose.** If we transform both \mathbf{P}_A (mug) and \mathbf{P}_B (rack) objects by the same transform, then the relative pose between these objects is unchanged (the mug is still “on” the rack) so the cross-pose is unchanged.

We aim to learn a model, f_θ , that takes as input the two point clouds and predicts an $SE(3)$ rigid transformation: $f_\theta(\mathbf{P}_A, \mathbf{P}_B) = \mathbf{T}_{AB}$, where \mathbf{T}_{AB} denotes the cross-pose between object \mathcal{A} and object \mathcal{B} as defined above. We can then transform object \mathcal{A} by \mathbf{T}_α and \mathcal{B} by \mathbf{T}_β such that $\mathbf{T}_\alpha \cdot \mathbf{T}_{AB} \cdot \mathbf{T}_\beta^{-1} = \mathbf{I}$. Given the properties of “cross-pose” described above, this will shift objects \mathcal{A} and \mathcal{B} into the desired target pose for the task. In practice, we typically transform only object \mathcal{A} by $\mathbf{T}_\alpha = \mathbf{T}_{AB}^{-1}$ without moving object \mathcal{B} , although in theory either (or both) objects can be moved.

3.4 Method

We frame the task of cross-pose estimation as a correspondence-prediction task between a pair of point clouds, followed by an analytical least-squares optimization to find the optimal *cross-pose* for the predicted correspondences. At a high level, our method performs the following steps:

1. **Soft Correspondence Prediction:** For a pair of objects \mathcal{A}, \mathcal{B} , a neural network learns to predict a per-point embedding to establish a (soft) correspondence between \mathcal{A} and \mathcal{B} .
2. **Adjustment via Correspondence Residuals:** To accommodate estimation tasks where \mathcal{A} and \mathcal{B} may not be in direct contact or overlap, we apply a pointwise residual vector to displace each of the predicted corresponding points. This allows points in \mathcal{A} to correspond to points in free space near \mathcal{B} , for instance when a pot (\mathcal{A}) sits on top of a stove (\mathcal{B}).
3. **Find the optimal Cross-Pose Transform:** We use a standard SVD solution to the weighted Procrustes problem to find the optimal alignment given the corrected soft correspondences.

Because each step above is fully-differentiable, our method can learn arbitrary correspondences that solve arbitrary cross-pose estimation tasks. Our method is heavily inspired by Deep Closest Point (DCP) [85]. The key difference between our pose alignment model and DCP is that we are predicting the cross-pose between two *different* objects for a given task instead of registering two point clouds of an identical object. An overview of our method can be found in Figure 3.3. As we will discuss, this correspondence-based approach allows our method to be translation-equivariant: translating one object (\mathcal{A} or \mathcal{B}) will lead to a translated cross-pose prediction.

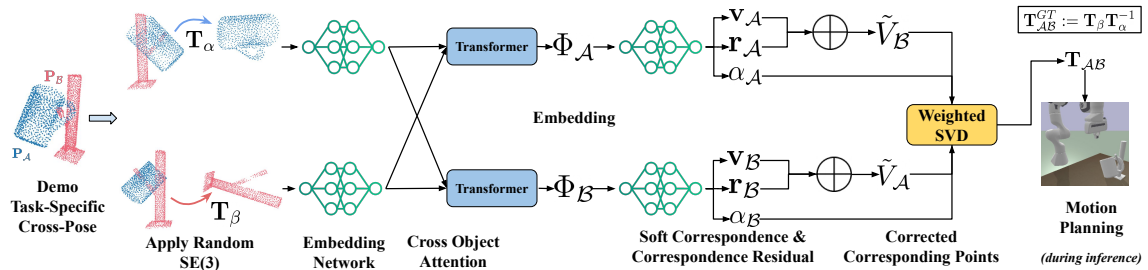


Figure 3.3: **TAX-Pose training overview.** Our method takes as input two point clouds given a specific task and outputs the cross-pose between them for the task. TAX-Pose first learns point clouds features using two DGCNN networks and two transformers. Then the learned features will be input to two point residual networks to predict per-point soft correspondence, correspondence residuals and SVD weights between the two objects. Then the desired cross-pose can be inferred analytically using weighted singular value decomposition.

We now describe our Cross-Pose estimation algorithm in detail. To recap the problem statement, given objects \mathcal{A} and \mathcal{B} with point cloud observations $\mathbf{P}_A, \mathbf{P}_B$ re-

spectively, our objective is to estimate the task-specific cross-pose $\mathbf{T}_{\mathcal{A}\mathcal{B}} = f(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}) \in SE(3)$. Note that the cross-pose between object \mathcal{A} and \mathcal{B} is defined with respect to a given task (e.g. putting a lasagna in the oven, putting a mug on the rack, etc).

3.4.1 Cross-Pose Estimation via Soft Correspondence Prediction

Soft Correspondence Prediction

The first step of the method is to compute two sets of correspondences between \mathcal{A} and \mathcal{B} , one which maps from points in \mathcal{A} to \mathcal{B} , and one which maps from points in \mathcal{B} to \mathcal{A} . These need not be a bijection, and can be asymmetric. We desire for this correspondence to be differentiable, so following DCP we define a *soft correspondence*, which assigns for every point $p_i^{\mathcal{A}} \in \mathbf{P}_{\mathcal{A}}$ a corresponding *virtual corresponding point* $v_i^{\mathcal{A} \rightarrow \mathcal{B}}$, which is a convex combination of points in $\mathbf{P}_{\mathcal{B}}$, and vice versa. Formally:

$$v_i^{\mathcal{A} \rightarrow \mathcal{B}} = \mathbf{P}_{\mathcal{B}} w_i^{\mathcal{A} \rightarrow \mathcal{B}} \quad \text{s.t.} \quad \sum_{j=1}^{N_{\mathcal{B}}} w_{ij}^{\mathcal{A} \rightarrow \mathcal{B}} = 1 \quad v_i^{\mathcal{B} \rightarrow \mathcal{A}} = \mathbf{P}_{\mathcal{A}} w_i^{\mathcal{B} \rightarrow \mathcal{A}} \quad \text{s.t.} \quad \sum_{j=1}^{N_{\mathcal{A}}} w_{ij}^{\mathcal{B} \rightarrow \mathcal{A}} = 1$$

with normalized weight vectors $w_i^{\mathcal{A} \rightarrow \mathcal{B}}$ and $w_i^{\mathcal{B} \rightarrow \mathcal{A}}$. Importantly, these virtual corresponding points are not constrained to the surfaces of \mathcal{A} or \mathcal{B} ; instead, they are constrained to the convex hulls of $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$, respectively. Thus, we can reduce the correspondence prediction problem to predicting $w_i^{\mathcal{A} \rightarrow \mathcal{B}}$ and $w_i^{\mathcal{B} \rightarrow \mathcal{A}}$ for each point in $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$, respectively.

To accomplish this, we first encode each point cloud $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$ into a latent space using a neural network encoder. This encoder head is comprised of two distinct encoders $g_{\mathcal{A}}$ and $g_{\mathcal{B}}$, each of which receives point cloud $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$, respectively, and outputs a dense, point-wise embedding for each object: $\Phi_{\mathcal{A}} = g_{\mathcal{A}}(\mathbf{P}_{\mathcal{A}})$, $\Phi_{\mathcal{B}} = g_{\mathcal{B}}(\mathbf{P}_{\mathcal{B}})$ where $\phi_i^{\mathcal{A}} \in \Phi_{\mathcal{A}}$ is the d -dimensional embedding of the i -th point in object \mathcal{A} , and likewise for object \mathcal{B} (see Figure 3.3). We zero-center each observation point cloud before passing it into its encoder, and we employ a cross-object attention module between the two embedding spaces. This enables our method to be translation-equivariant, as discussed in Appendix A.3.

Since the point-wise embeddings $\phi_i^{\mathcal{A}}$ and $\phi_i^{\mathcal{B}}$ have the same dimension d , we can select the inner product of the space as a similarity metric between two embeddings.

For any point p_i^A , we can extract the desired normalized weight vector $w_i^{\mathcal{B} \rightarrow \mathcal{A}}$ with the softmax function:

$$w_i^{\mathcal{A} \rightarrow \mathcal{B}} = \text{softmax}(\Phi_{\mathcal{B}}^\top \phi_i^{\mathcal{A}}), \quad w_i^{\mathcal{B} \rightarrow \mathcal{A}} = \text{softmax}(\Phi_{\mathcal{A}}^\top \phi_i^{\mathcal{B}}) \quad (3.3)$$

Adjustment via Correspondence Residuals

Correspondences which are constrained to the convex hull of objects are insufficient to express a large class of desired tasks. For instance, we might want a point on the handle of a teapot to correspond to some point above a stovetop, which lies outside the convex hull of the points on the stovetop. To allow for such placements, we further learn a *residual vector* that corrects each virtual corresponding point, allowing us to displace each virtual corresponding point to any arbitrary location that might be suitable for the task. Concretely, we use a point-wise neural network $g_{\mathcal{R}}$ which maps each embedding into a 3-D residual vector:

$$r_i^{\mathcal{A} \rightarrow \mathcal{B}} = g_{\mathcal{R}}(\phi_i^{\mathcal{A}}), \quad r_i^{\mathcal{B} \rightarrow \mathcal{A}} = g_{\mathcal{R}}(\phi_i^{\mathcal{B}})$$

Applying these to the virtual corresponding points, we get our *corrected virtual correspondence*:

$$\tilde{v}_i^{\mathcal{A} \rightarrow \mathcal{B}} = v_i^{\mathcal{A} \rightarrow \mathcal{B}} + r_i^{\mathcal{A} \rightarrow \mathcal{B}}, \quad \tilde{v}_i^{\mathcal{B} \rightarrow \mathcal{A}} = v_i^{\mathcal{B} \rightarrow \mathcal{A}} + r_i^{\mathcal{B} \rightarrow \mathcal{A}} \quad (3.4)$$

as shown in Figure 3.4.

Least-Squares Cross-Pose Optimization with Weighted SVD

We now have two sets of points and their associated corrected virtual correspondence: $(\mathbf{P}_{\mathcal{A}}, \tilde{\mathbf{V}}_{\mathcal{B}})$ and $(\mathbf{P}_{\mathcal{B}}, \tilde{\mathbf{V}}_{\mathcal{A}})$, where $\tilde{\mathbf{V}}_{\mathcal{B}} = [\tilde{v}_1^{\mathcal{A} \rightarrow \mathcal{B}} \dots \tilde{v}_{N_{\mathcal{A}}}^{\mathcal{A} \rightarrow \mathcal{B}}]^\top$ and similarly for $\tilde{\mathbf{V}}_{\mathcal{A}}$. We would like to compute the cross-pose transformation $\mathbf{T}_{\mathcal{AB}}$ that minimizes the weighted distance between correspondences, where the weights signify the importance of specific correspondences and are predicted as an additional channel of the encoding neural networks, denoted as $\alpha_{\mathcal{A}}, \alpha_{\mathcal{B}}$ for $\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}$ respectively. This is the well-known weighted Procrustes problem, for which there exists an analytical solution (see Appendix A.4 for details). We use a differentiable SVD operation [54], which allows

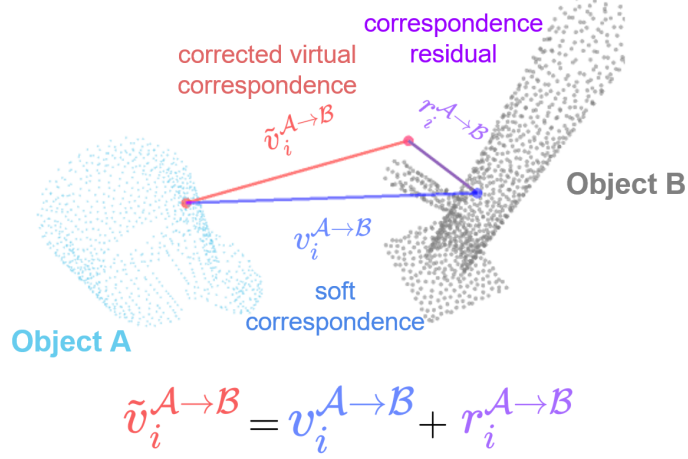


Figure 3.4: **Computation of corrected virtual correspondence.** Given a pair of objects \mathcal{A}, \mathcal{B} , a per-point soft correspondence $v_i^{A \rightarrow B}$ is first computed. Next to allow the predicted correspondence to lie beyond object’s convex hull, these soft correspondences are adjusted with correspondence residuals, $r_i^{A \rightarrow B}$, which results in the corrected virtual correspondence, $\tilde{v}_i^{A \rightarrow B}$.

us to compute a rotation R and translation t that minimizes the weighted distance between correspondences (where $\mathbf{T}_{AB} = [R, t]$). Figure 3.5 goes through a visual example of extracting cross-pose from corrected virtual correspondence through weighted SVD.

3.4.2 Supervision

To train the encoders $g_{\mathcal{A}}(\mathbf{P}_{\mathcal{A}})$, $g_{\mathcal{B}}(\mathbf{P}_{\mathcal{B}})$ as well as the residual networks $g_{\mathcal{R}}(\phi_i^{\mathcal{A}})$, $g_{\mathcal{R}}(\phi_i^{\mathcal{B}})$, we use a set of losses defined below. We assume we have access to a set of demonstrations of the task, in which the action and anchor objects are in the target relative pose such that $\mathbf{T}_{AB} = \mathbf{I}$.

Point Displacement Loss

Instead of directly supervising the rotation and translation (as is done in DCP), we supervise the predicted transformation using its effect on the points. For this loss [45, 91], we take the point clouds of the objects in the demonstration configuration, and transform each cloud by a random transform, $\hat{\mathbf{P}}_{\mathcal{A}} = \mathbf{T}_{\alpha} \mathbf{P}_{\mathcal{A}}$, and $\hat{\mathbf{P}}_{\mathcal{B}} = \mathbf{T}_{\beta} \mathbf{P}_{\mathcal{B}}$. This would give us a ground truth transform of $\mathbf{T}_{AB}^{GT} = \mathbf{T}_{\beta} \mathbf{T}_{\alpha}^{-1}$; the inverse of this

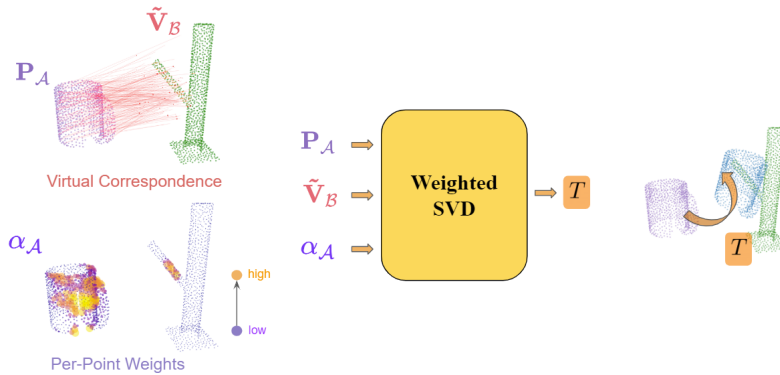


Figure 3.5: **Illustration of weighted SVD.** Given a point cloud, \mathbf{P}_A , the associated corrected virtual correspondence, $\tilde{\mathbf{V}}_B$, and the per-point weights predicted, α_A , we extract a $SE(3)$ transformation, \mathbf{T} . We can then apply to the original point cloud \mathbf{P}_A to result in a desirable cross-pose for the task.

transform would move object \mathcal{B} to the correct position relative to object \mathcal{A} . Using this ground truth transform, we compute the MSE loss between the correctly transformed points and the points transformed using our prediction.

$$\mathcal{L}_{\text{disp}} = \|\mathbf{T}_{AB}\mathbf{P}_A - \mathbf{T}_{AB}^{GT}\mathbf{P}_A\|^2 + \|\mathbf{T}_{AB}^{-1}\mathbf{P}_B - \mathbf{T}_{AB}^{GT-1}\mathbf{P}_B\|^2 \quad (3.5)$$

Direct Correspondence Loss

While the Point Displacement Loss best describes errors seen at inference time, it can lead to correspondences that are inaccurate but whose errors average to the correct pose. To improve these errors we directly supervise the learned correspondences $\tilde{\mathbf{V}}_A$ and $\tilde{\mathbf{V}}_B$:

$$\mathcal{L}_{\text{corr}} = \|\tilde{\mathbf{V}}_A - \mathbf{T}_{AB}^{GT}\mathbf{P}_A\|^2 + \|\tilde{\mathbf{V}}_B - \mathbf{T}_{AB}^{GT-1}\mathbf{P}_B\|^2. \quad (3.6)$$

Correspondence Consistency Loss

Furthermore, a consistency loss can be used. This loss penalizes correspondences that deviate from the final predicted transform. A benefit of this loss is that it can help the network learn to respect the rigidity of the object, while it is still learning to accurately place the object. Note, that this is similar to the Direct Correspondence Loss, but uses the predicted transform as opposed to the ground truth one. As such,

this loss requires no ground truth:

$$\mathcal{L}_{\text{cons}} = \left\| \tilde{\mathbf{V}}_{\mathcal{A}} - \mathbf{T}_{\mathcal{AB}} \mathbf{P}_{\mathcal{A}} \right\|^2 + \left\| \tilde{\mathbf{V}}_{\mathcal{B}} - \mathbf{T}_{\mathcal{AB}}^{-1} \mathbf{P}_{\mathcal{B}} \right\|^2. \quad (3.7)$$

3.4.3 Overall Training Procedure

We train with a combined loss \mathcal{L}_{net} as,

$$\mathcal{L}_{\text{net}} = \mathcal{L}_{\text{disp}} + \lambda_1 \mathcal{L}_{\text{corr}} + \lambda_2 \mathcal{L}_{\text{cons}} \quad (3.8)$$

where λ_1 and λ_2 are hyperparameters. We use a similar network architecture as DCP [85], which consists of DGCNN [86] and Transformer [81]. We briefly experimented with Vector Neurons [24] and found that this led to worse performance on this task. In order to quickly adapt to new tasks, we optionally pre-train the DGCNN embedding networks over a large set of individual objects using the InfoNCE loss [52] with a geometric distance weighting and random transformations, to learn $SE(3)$ invariant embeddings.

Pre-Training

We utilize pre-training for the embedding network for the mug hanging task, and describe the details below.

We pretrain embedding network for each object category (mug, rack, gripper), such that the embedding network is $SE(3)$ *invariant* with respect to the point clouds of that specific object category. Specifically, the mug-specific embedding network is pretrained on 200 ShapeNet [15] mug instances, while the rack-specific and gripper-specific embedding network is trained on the same rack and Franka gripper used at test time, respectively.

For the network to be trained to be $SE(3)$ invariant, we pre-train with InfoNCE loss [52] with a geometric distance weighting and random $SE(3)$ transformations. Specifically, given a point cloud of an object instance, $\mathbf{P}_{\mathcal{A}}$, of a specific object category \mathcal{A} , and an embedding network $g_{\mathcal{A}}$, we define the point-wise embedding for as $\Phi_{\mathcal{A}} = g_{\mathcal{A}}(\mathbf{P}_{\mathcal{A}})$, where $\phi_i^{\mathcal{A}} \in \Phi_{\mathcal{A}}$ is a d -dimensional vector for each point $p_i^{\mathcal{A}} \in \mathbf{P}_{\mathcal{A}}$. Given a random $SE(3)$ transformation, \mathbf{T} , we define $\Psi_{\mathcal{A}} = g_{\mathcal{A}}(\mathbf{T}\mathbf{P}_{\mathcal{A}})$.

The weighted contrastive loss used for pretraining, \mathcal{L}_{wc} , is defined as

$$\mathcal{L}_{wc} := - \sum_{p_i^A \in \mathbf{P}_{\mathcal{A}}} \log \left[\frac{\exp(d_{ij} \cdot (\phi_i^{A\top} \psi_j^A))}{\sum_{p_k^A \in \mathbf{P}_{\mathcal{A}}} \exp(d_{ik} \cdot (\phi_i^{A\top} \psi_k^A))} \right] \quad (3.9)$$

$$d_{ij} := \begin{cases} \mu \tanh(\lambda \|p_i^A - p_j^A\|_2), & \text{if } i \neq j \\ 1, & \text{otherwise} \end{cases} \quad (3.10)$$

$$\mu := \max(\tanh(\lambda \|p_i^A - p_j^A\|_2)) \quad (3.11)$$

For this pretraining, we use $\lambda := 10$.

3.5 Experiments

To evaluate TAX-Pose, we conduct a wide range of simulated and real-world experiments on two classes of relative placement tasks: Object Placement and Mug Hanging. The Object Placement objective is to place an action object on a flat surface on or near an anchor object. The Mug Hanging task objective is to grasp and then hang unseen mugs on a rack.

3.5.1 PartNet-Mobility Placement

Task Description

The PartNet-Mobility Placement task is defined as placing a given action object relative to an anchor object based on a semantic goal positions. We select a set of household furniture objects from the PartNet-Mobility dataset [90] as the anchor objects, and a set of small rigid objects released with the Ravens simulation environment [94] as the action objects. For each anchor object, we define a set of semantic goal positions (i.e. ‘top’, ‘left’, ‘right’, ‘in’), where action objects should be placed relative to each anchor. Each semantic goal position defines a unique task in our cross-pose prediction framework. Given a synthetic point cloud observation of both objects, the task is to predict a cross-pose that places the object at the specific semantic goal. We train a single model across anchor/action categories, one model per semantic task (for a total of 4 models). We train entirely on simulated

3. Task-Specific Cross-Pose Estimation for Robot Manipulation

data, and transfer directly to real world with no finetuning. Details can be found in Appendix A.5. We report rotation (\mathcal{E}_R) and translation (\mathcal{E}_t) error between our predicted transform and the ground truth as geodesic rotational distance [32, 39] and $L2$ distance, respectively.

Baselines

We compare our method to the following baselines:

- *E2E Behavioral Cloning*: Generate motion-planned trajectories using OMPL that take the action object from start to goal. These serve as “expert” trajectories for Behavioral Cloning (BC), where we train a neural network to output a policy that, at each time step, outputs an incremental 6-DOF transformation that imitates the expert trajectory
- *E2E DAgger*: Using the same BC dataset as above, we train a policy using DAgger [59]
- *Trajectory Flow*: Using the same BC dataset with DAgger, we train a policy to predict a dense per-point 3D flow vector at each time step instead of a single incremental 6-DOF transformation. Given this dense per-point flow, we can extract a rigid transformation using SVD yielding the next pose
- *Goal Flow*: Instead of training a multi-step policy to reach the goal, train a network to output a single dense prediction which assigns a per-point 3D flow vector that points from each action object point directly to its corresponding goal location. We extract a rigid transformation from these flow vectors using SVD, yielding the goal pose










		AVG.																	
		\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t						
Baselines	E2E BC	42.26	0.73	37.82	0.82	37.15	0.65	44.84	0.68	30.69	1.06	40.38	0.69	45.09	0.76	45.00	0.79	45.65	0.64
	E2E DAgger [59]	37.96	0.69	34.15	0.76	36.61	0.66	40.91	0.65	24.87	0.97	35.95	0.70	40.34	0.74	32.86	0.79	39.45	0.53
Ablations	Traj. Flow [26]	35.95	0.67	31.24	0.82	39.21	0.72	34.35	0.66	28.48	0.75	37.14	0.59	29.49	0.70	39.60	0.76	39.69	0.48
	Goal Flow [26]	26.64	0.17	25.88	0.15	25.05	0.15	30.62	0.15	27.61	0.10	28.01	0.18	20.96	0.24	29.02	0.23	22.13	0.20
Ours	TAX-Pose	6.64	0.16	6.85	0.16	2.05	0.10	3.87	0.12	4.04	0.08	12.71	0.31	6.87	0.37	5.89	0.13	14.93	0.18

Table 3.1: Goal Inference Rotational and Translational Error Results (\downarrow). Rotational errors (\mathcal{E}_R) are in degrees ($^\circ$) and translational errors (\mathcal{E}_t) are in meters (m). The lower the better.

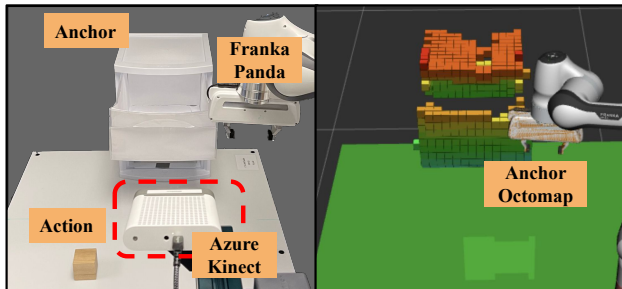


Figure 3.6: **Real-world experiments illustration.** **Left:** work-space setup for physical experiments. **Center:** Octomap visualization of the perceived anchor object.




			
Goal Flow	0.18	0.38	0.37
TAX-Pose	0.95	0.95	0.85

Table 3.2: Real-world goal placement success rate

Real-World Experiments

We design a set of real world experiments to evaluate the performance of our cross-pose prediction model on real objects. We choose several real-world furniture objects similar to those found in the simulated training categories, annotate semantic goal locations for each, and choose several analogous action objects (bowl, block) to place at the goals. For each semantic goal, the task is to predict the appropriate cross-pose directly from point clouds recorded by a depth camera and have a Franka Emika Panda robot place the action object at the cross-pose (see Fig. 3.6 for the workspace). We compare TAX-Pose and the Goal Flow baseline for pose estimation and use OMPL motion planning on top of an Octomap [38] scene reconstruction to plan placement trajectories. As ground-truth cross-pose is hard to define in the real-world, we qualitatively define a “goal region” for each anchor object. Success is defined as the inferred pose of the action object landing inside the goal region of the anchor object. Details can be found in Appendix A.5.

Results

In both our simulated experiments (Table 3.1) and our real-world experiments (Table 3.2), we find that TAX-Pose outperforms the baseline methods. In simulated experiments, while direct regression via goal-flow outperforms TAX-Pose in some rare cases of translation prediction, TAX-Pose performs substantially better than all other baselines in rotation prediction. Moreover, in real-world experiments, due to the provably translation-equivariant property of TAX-Pose, it generalizes to

novel distributions of starting poses better than the Goal Flow regression baseline, successfully placing action objects into the goal regions.

3.5.2 Mug Hanging

Task Description

The Mug Hanging task is consisted of two sub tasks: *grasp* and *place*. See Figure 3.7 for a detailed breakdown of the mug hanging task in stages.

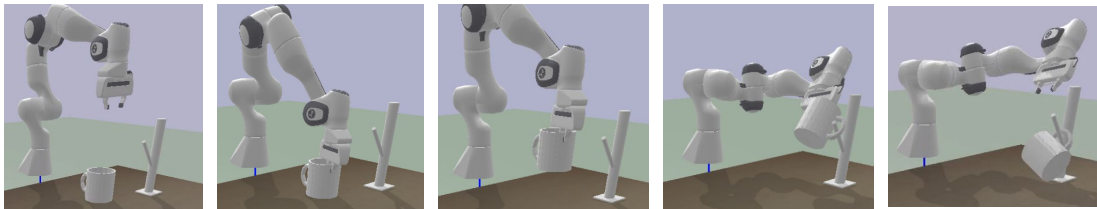


Figure 3.7: **Visualization of mug hanging task (upright pose).** Mug hanging task is consisted of two stages, given a mug that is randomly initialized on the table, the model first predicts a $SE(3)$ transform from gripper end effector to the mug rim $\mathbf{T}_{g \rightarrow m}$, then grasp it by the rim. Next, the model predicts another $SE(3)$ transform from the mug to the rack $\mathbf{T}_{m \rightarrow r}$ such that the mug handle gets hanged on the the mug rack.

To successfully execute the manipulation task of hanging a mug on a rack by the mug’s handle requires the successful inference of two sequential task-specific cross-poses: 1) predict a successful grasp pose; 2) predict the hanging pose of mug relative to the rack. In the first stage, it requires our model to reason about the cross-pose between the gripper and the mug, while the second stage requires prediction of the cross-pose between the mug and the rack.

Task Setup

To evaluate the performance of our method in simulation, we utilize Pybullet [18] and simulate a Franka Panda arm situated above a table with 4 depth cameras placed at each table corner. For training, the model is provided with 10 demonstrations in simulation, each on a different mug instances. At test time, we measure the task execution success on unseen mug instances, with randomly generated initial poses.

To evaluate robustness to different initial poses, we evaluate on two sets of initial poses:

1. *Upright Pose*: where the mug is initialized to have an upright orientation, and placed randomly on the surface of the table
2. *Arbitrary Pose*: where the mug is initialized to have arbitrary orientation and position, irrespective of the location of the table surface

We measure task success rates of the following task categories:

- *Grasping*: where success is achieved when the object is grasped stably
- *Placing*: where success is achieved when the mug is placed stably on the rack
- *Overall*: when the predicted transforms enable both grasp and place success.

Baselines

In simulation, we compare our method to the results described in [68].

- **Dense Object Nets (DON)** [27]: Using manually labeled semantic keypoints on the demonstration clouds, DON is used to compute sparse correspondences with the test objects. These correspondences are converted to a pose using SVD. A full description of usage of DON for the mug hanging task can be found in [68].
- **Neural Descriptor Field (NDF)** [68]: Using the learned descriptor field for the mug, the positions of a constellation of task specific quarry points are optimized to best match the demonstration using gradient descent.

Training Data

To be directly comparable with the baselines we compared to, we use the exact same sets of demonstration data used to train the network in NDF [68], where the data are generated via teleportation in PyBullet, collected on 10 mug instances with random pose initialization.

We compare our method to Neural Descriptor Field (NDF) [68] and Dense Object Nets (DON) [27]. Details of these methods can be found in [68].

Training and Inference

Using the pretrained embedding network for mug and gripper, we train a grasping model for the grasping task to predict a transformation $\mathbf{T}_{g \rightarrow m}$ in gripper’s frame from gripper to mug to complete the *grasp* stage of the task. Similarly, using the pretrained embedding network for rack and mug, we train a placement model for the placing task to predict a transformation $\mathbf{T}_{m \rightarrow r}$ in mug’s frame from mug to rack to complete the *place* stage of the task. Both model are trained with the same combined loss \mathcal{L}_{net} as described in the main paper. During inference, we simply use grasping model to predict the $\mathbf{T}_{g \rightarrow m}$ at test time, and placement model to predict $\mathbf{T}_{m \rightarrow r}$ at test time.

Motion Planning

After the model predicts a transformation $\mathbf{T}_{g \rightarrow m}$ and $\mathbf{T}_{m \rightarrow r}$, using the known gripper’s world frame pose, we calculate the desired gripper end effector pose at grasping and placement, and pass the end effector to IKFast to get the desired joint positions of Franka at grasping and placement. Next we pass the desired joint positions at gripper’s initial pose, and desired grasping joint positions to OpenRAVE motion planning library to solve for trajectory from gripper’s initial pose to grasp pose, and then grasp pose to placement pose for the gripper’s end effector.

	Grasp	Place	Overall	Grasp	Place	Overall
	Upright Pose			Arbitrary Pose		
DON [27]	0.91	0.50	0.45	0.35	0.45	0.17
NDF [68]	0.96	0.92	0.88	0.78	0.75	0.58
TAX-Pose (Ours)	0.99	0.97	0.96	0.75	0.84	0.63

Table 3.3: Mug on Rack Simulation Task Success Results

Results

We quantitatively evaluate our method in simulation on 100 trials on different initial configurations on unseen mug instances with randomly generated pose configurations for both **Upright** and **Arbitrary** poses and compare the performance of our method against DON [27] and NDF [68]. See Table 3.3 for full simulation results.

Model	# Demos Used		
	1	5	10
DON [27]	0.32	0.36	0.45
NDF [68]	0.46	0.70	0.88
TAX-Pose (Ours)	0.77	0.90	0.96

Table 3.4: # Demos vs. Overall Success

Ablation Analysis

Number of Demonstrations. To study the effects of number of demonstrations used on the performance of our method, we report quantitative performance of our method alongside baseline methods trained on different numbers of demonstrations (10, 5, 1) for upright pose mug hanging task as seen in Table 4. Our method outperforms the baselines for all number of demonstrations; TAX-Pose can perform well even with just 5 demonstrations.

Cross-Pose Estimation Design Choices

We analyze the effects of the different design choices made in our Cross-Pose estimation algorithm for the upright pose mug hanging task. In order to examine the effects of different design choices in the training pipeline, we conduct ablation experiments with final task-success (*grasp*, *place*, *overall*) as evaluation metrics for Mug Hanging task with upright pose initialization for the following components of our method, see Table 3.5 for full ablation results. For consistency, all ablated models are trained to 15K batch steps.

1. **Loss.** In the full pipeline reported, we use a weighted sum of the three types of losses described in **Section 4.2** of the paper. Specifically, the loss used \mathcal{L}_{net} is given by

$$\mathcal{L}_{\text{net}} = \mathcal{L}_{\text{disp}} + \lambda_1 \mathcal{L}_{\text{cons}} + \lambda_2 \mathcal{L}_{\text{corr}} \quad (3.12)$$

where we chose $\lambda_1 = 0.1$, $\lambda_2 = 1$ after hyperparameter search.

We ablate usage of all three types of losses, by reporting the final task performance in simulation for all experiments, specifically, we report task success on the following \mathcal{L}_{net} variants.

- (a) Remove the point displacement loss term, $\mathcal{L}_{\text{disp}}$, after which we are left

3. Task-Specific Cross-Pose Estimation for Robot Manipulation

with

$$\mathcal{L}'_{\text{net}} = (0.1)\mathcal{L}_{\text{cons}} + \mathcal{L}_{\text{corr}}$$

- (b) Remove the direct correspondence loss term, $\mathcal{L}_{\text{corr}}$, after which we are left with

$$\mathcal{L}'_{\text{net}} = \mathcal{L}_{\text{disp}} + (0.1)\mathcal{L}_{\text{cons}}$$

- (c) Remove the correspondence consistency loss term, $\mathcal{L}_{\text{cons}}$, after which we are left with

$$\mathcal{L}'_{\text{net}} = \mathcal{L}_{\text{disp}} + \mathcal{L}_{\text{corr}}$$

- (d) From testing loss variants above, we found that the point displacement loss is a vital contributing factor for task success, where removing this loss term results in no overall task success, as shown in Table 3.5. However, in practice, we have found that adding the correspondence consistency loss and direct correspondence loss generally help to lower the rotational error of predicted placement pose compared to the ground truth of collected demos. To further investigate the effects of the combination of these two loss terms, we used a scaled weighted combination of $\mathcal{L}_{\text{cons}}$ and $\mathcal{L}_{\text{corr}}$, such that the former weight of the displacement loss term is transferred to consistency loss term, with the new $\lambda_1 = 1.1$, and with $\lambda_2 = 1$ stays unchanged. Note that this is different from variant (a) above, as now the consistency loss given a comparable weight with dense correspondence loss term, which intuitively makes sense as the consistency loss is a function of the predicted transform \mathbf{T}_{AB} to be used, while the dense correspondence loss is instead a function of the ground truth transform, \mathbf{T}_{AB}^{GT} , which provides a less direct supervision on the predicted transforms. Thus we are left with

$$\mathcal{L}'_{\text{net}} = (1.1)\mathcal{L}_{\text{cons}} + \mathcal{L}_{\text{corr}}$$

2. **Usage of Correspondence Residuals.** After predicting a per-point soft correspondence between objects \mathcal{A} and \mathcal{B} , we adjust the location of the predicted corresponding points by further predicting a point-wise correspondence residual vector to displace each of the predicted corresponding point. This allows the

predicted corresponding point to get mapped to free space outside of the convex hulls of points in object \mathcal{A} and \mathcal{B} . This is a desirable adjustment for mug hanging task, as the desirable cross-pose usually require points on the mug handle to be placed somewhere near but not in contact with the mug rack, which can be outside of the convex hull of rack points. We ablate correspondence residuals by directly using the soft correspondence prediction to find the cross-pose transform through weighted SVD, without any correspondence adjustment via correspondence residual.

3. **Weighted SVD vs Non-weighted SVD.** We leverage weighted SVD as described in **Section 4.1** of the paper as we leverage predicted per-point weight to signify the importance of specific correspondence. We ablate the use of weighted SVD, and we use an un-weighted SVD, where instead of using the predicted weights, each correspondence is assign equal weights of $\frac{1}{N}$, where N is the number of points in the point cloud \mathbf{P} used.
4. **Pretraining.** In our full pipeline, we pretrain the point cloud embedding network such that the embedding network is $SE(3)$ invariant. Specifically, the mug-specific embedding network is pretrained on 200 ShapeNet mug objects, while the rack-specific and gripper specific embedding network is trained on the same rack and Franka gripper used at test time, respectively. We conduct ablation experiments where,
 - (a) We omit the pretraining phase of embedding network
 - (b) We do not finetune the embedding network during downstream training with task-specific demonstrations.

Note that in practice, we find that pretraining helps speed up the downstream training by about a factor of 3, while models with or without pretraining both reach a similar final performance in terms of task success after both models converge.

5. **Usage of Transformer as Cross-object Attention Module.** In the full pipeline, we use transformer as the cross-object attention module, and we ablate this design choice by replacing the transformer architecture with a simple 3-layer MLP with ReLU activation and hidden dimension of 256, and found that this

3. Task-Specific Cross-Pose Estimation for Robot Manipulation

leads to worse place and grasp success.

- 6. Dimension of Embedding.** In the full pipeline, the embedding is chosen to be of dimension 512. We conduct experiment on much lower dimension of 16, and found that with dimension =16, the place success is much lower, dropped from 0.97 to 0.59.

Ablation Experiment	Grasp	Place	Overall
No $\mathcal{L}_{\text{disp}}$	0.01	0	0
No $\mathcal{L}_{\text{corr}}$	0.89	0.91	0.84
No $\mathcal{L}_{\text{cons}}$	0.99	0.95	0.94
Scaled Combination of $\mathcal{L}_{\text{corr}}$ & $\mathcal{L}_{\text{cons}}$ ($(1.1)\mathcal{L}_{\text{cons}} + \mathcal{L}_{\text{corr}}$)	0.10	0.01	0.01
No Adjustment via Correspondence Residuals	0.97	0.96	0.93
Unweighted SVD	0.92	0.94	0.88
No Finetuning for Embedding Network	0.98	0.93	0.91
No Pretraining for Embedding Network	0.99	0.72	0.71
3-Layer MLP In Place of Transformer	0.90	0.82	0.76
Embedding Network Feature Dim = 16	0.98	0.59	0.57
TAX-Pose (Ours)	0.99	0.97	0.96

Table 3.5: Mug Hanging Ablations Results

3.5.3 Failure Cases

Some failure cases for TAX-Pose happens when the predicted gripper misses the rim of the mug by a xy-plane translation error, thus resulting in failure of grasp, as seen in Figure 3.8. And common failure mode for the mug placement subtask is characterized by erroneous transform prediction that results in the mug’s handle completely missing the rack hanger, thus resulting in placement failure, as seen in Figure 3.9.

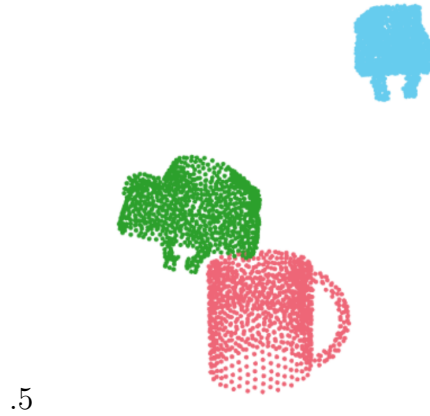


Figure 3.8: **Failure of *grasp* prediction.** Predicted TAX-Pose for the gripper misses the rim of mug.

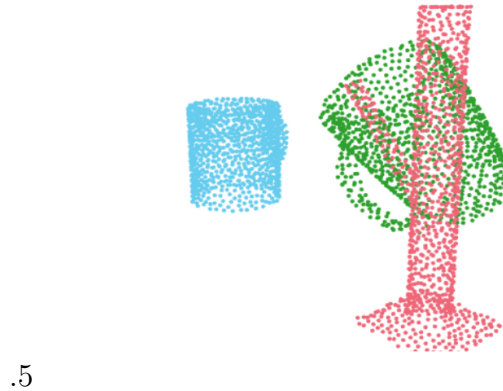


Figure 3.9: **Failure of *place* prediction.** Predicted TAX-Pose for mug results in the mug handle misses the rack hanger completely.

Figure 3.10: **An illustration of unsuccessful TAX-Pose predictions for mug hanging.** In both subfigures, red points represent the anchor object, blue points represent action object’s starting pose, and green points represent action object’s predicted pose.

3.6 Limitations & Future Directions

While our method is able to accurately predict the requisite transform to achieve a given task, it does require an accurate segmentation of the objects of importance. Additionally, while our method is tested with some occlusions, it performs better with a mostly complete cloud of the object being manipulated. This means that multiple views of that object must be captured. This can be done with multiple cameras, or by lifting the object and capturing multiple views. Additionally, as our method is a

function of correspondences, symmetries could cause potential problems. This could be caused by interacting with symmetric objects or multimodality in the task to be completed, such as objects with multiple valid placement surfaces, or racks with multiple usable hangers. These problems could be alleviated using a consensus-based method for mapping from multimodal soft correspondences to a single transform. We leave this for future work.

3.7 Conclusions

In this part of the thesis, we show that dense soft correspondence can be used to learn task specific object relationships that generalize to novel object instances. Correspondence residuals allow our method to estimate correspondences to virtual points, outside of the objects convex hull, drastically increasing the number of tasks this method can complete. We further show that this “cross-pose” can be learned for a task, using a small number of demonstrations. Finally, we show that our method far outperforms the baselines on two challenging tasks in both real and simulated experiments.

Chapter 4

Conclusions

In conclusions, we explored and presented two different approaches to training accurate, generalizable 6D pose estimator for manipulation tasks with minimal supervision needed, for image and point cloud data respectively.

In the first part, we show that through the use of Modified Rodrigues Parameters, we are able to open the closed manifold of $SO(3)$, improving the convergence behavior of the rotation averaging problem. We show that Iterative Modified Rodrigues Projective Averaging is able to outperform the naive application of relative-orientation supervision in both direct parameter optimization and image-based rotations estimation from neural networks. We hope our method allows more systems to convert the relative supervision of relative methods, like ICP, to consistent and accurate absolute poses.

In the second part, we presented a new lens to solving pose estimation for manipulation tasks, by instead of seeking to obtain accurate and robust single object absolute poses, we instead of directly train network to learn to predict the desired task-specific cross-pose between a pair of objects by learning from a few task demonstrations. We show that dense soft correspondence can be used to learn task specific object relationships that generalize to novel object instances. Correspondence residuals allow our method to estimate correspondences to virtual points, outside of the objects convex hull, drastically increasing the number of tasks this method can complete. We further show that this “cross-pose” can be learned for a task, using a small number of demonstrations. Finally, we show that our method far outperforms

4. *Conclusions*

the baselines on two challenging tasks in both real and simulated experiments.

Appendix A

Appendix

A.1 Additional Local Rotation Averaging Experiment Results on Structure from Motion Dataset

We report results on all structure from motions datasets available in the 1DSfM [89]. Each environment is tested with 5 random initializations and the estimated rotations are updated by each algorithm in batches of size 64, for 20K iterations. While Iterative Modified Rodrigues Projective Averaging, **MRP (Ours)** outperform all **PMG** [17] based methods, the direct **Quaternion** optimization regularly converges to relatively accurate local optima more quickly than ours, as shown in Table A.3 and Figure A.1. That being said, our method converges to a more accurate final configuration for most datasets, with respect to mean relative error, Table A.4, mean absolute error, Table A.1, and median absolute error, Table A.2. Our method, as well as the baselines, do not appear to perform well on the larger datasets. As a reminder, this algorithm is specifically designed for training deep learned methods, not for direct rotation optimization. When training deep learned methods, all of the weights are shared, allowing the network to use a single example to improve the accuracy of all rotations near that example. Additionally, we see poor performance on datasets with extremely large observation noise, specifically Gendarmenmarkt, whose median observation error is over 12 degrees. All dataset statistics can be found in

A. Appendix

Table A.5. These datasets do not fully cover the orientation space, and tend to largely cover only variations in yaw. For results on datasets that represent full coverage of the orientation space, see the Uniformly Sampled Rotations dataset or the Neural Network Optimization dataset.

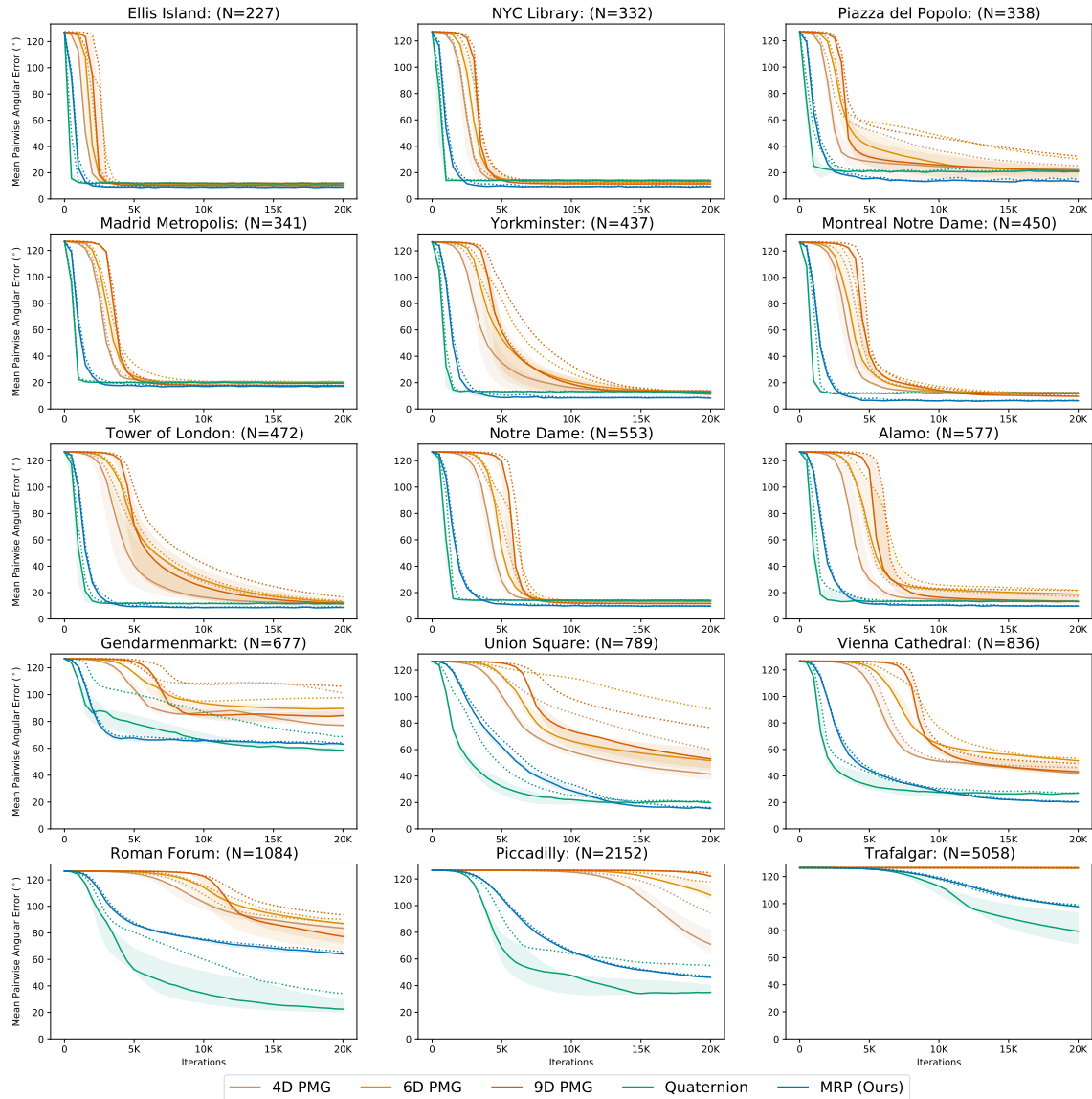


Figure A.1: **Optimization results for all 1DSfM [89] datasets, ordered by number of cameras (N).** Median average-pairwise angular error ($^{\circ}$) is shown with shaded areas representing the first and third quartile over all training sessions. The max average-pairwise angular error for each algorithm at each iteration is shown as a dashed line.

Dataset	<i>Mean Absolute Error</i>				
	PGM 4D	PGM 6D	PGM 9D	Quat	MRP (Ours)
Ellis Island	7.5	7.03	6.41	7.44	5.59
NYC Library	9.23	8.32	7.38	8.92	6.03
Piazza del Popolo	16.37	16.1	15.88	15.24	10.03
Madrid Metropolis	13.55	13.23	11.78	13	11.25
Yorkminster	9.13	8.34	7.48	8.56	5.3
Montreal Notre Dame	8.17	7.65	6.24	7.76	4.02
Tower of London	8.02	8.12	8.36	7.44	5.58
Notre Dame	8.71	7.96	7.03	8.55	5.80
Alamo	9.41	11.98	10.98	8.74	6.42
Gendarmenmarkt	66.41	73.7	68.29	46.63	48.82
Union Square	32.46	40.86	40.92	13.44	10.22
Vienna Cathedral	29.18	31.42	32.94	18.67	13.60
Roman Forum	63.23	64.85	60.51	18.11	55.65
Piccadilly	53.35	84.37	106.84	26.29	29.98
Trafalgar	121.93	124.18	125.15	69.65	91.67

Table A.1: Final Mean Absolute Error ($^{\circ}$) on all 1DSfM [89] datasets after 20K iterations

A. Appendix

Dataset	<i>Median Absolute Error</i>				
	PGM 4D	PGM 6D	PGM 9D	Quat	MRP (Ours)
Ellis Island	3.68	3.25	3.12	4.04	2.96
NYC Library	6.11	5.52	4.85	6.11	4.04
Piazza del Popolo	9.51	9.32	9.32	9.29	6.12
Madrid Metropolis	9.37	9.06	7.86	9.07	6.99
Yorkminster	6.44	5.77	4.56	6.11	3.29
Montreal Notre Dame	3.86	3.56	2.86	3.90	2.30
Tower of London	4.87	5.84	6.36	4.64	3.59
Notre Dame	4.39	3.73	3.09	4.48	2.61
Alamo	4.73	5.77	5.16	4.90	3.48
Gendarmenmarkt	64.08	71.57	62.9	43.91	45.92
Union Square	27.75	34.68	34.84	9.75	6.85
Vienna Cathedral	13.80	13.77	16.73	11.67	6.34
Roman Forum	53.78	62.46	57.71	16.56	41.95
Piccadilly	42.34	79.74	107.32	19.67	15.09
Trafalgar	126.71	129.57	130.45	65.54	89.09

Table A.2: Final Median Absolute Error ($^{\circ}$) on all 1DSfM [89] datasets after 20K iterations

Dataset	<i>Mean nAUC</i>				
	PGM 4D	PGM 6D	PGM 9D	Quat	MRP (Ours)
Ellis Island	22.56	24.07	25.02	15.05	14.58
NYC Library	28.53	31.12	32.07	18.20	16.84
Piazza del Popolo	37.36	44.18	43.98	25.13	22.21
Madrid Metropolis	35.91	38.49	39.15	24.34	24.48
Yorkminster	36.82	42.37	44.91	18.71	18.43
Montreal Notre Dame	33.97	37.54	40.37	17.69	16.19
Tower of London	39.98	45.99	49.54	18.14	18.85
Notre Dame	38.77	43.04	46.05	20.78	21.10
Alamo	39.87	49.08	50.22	20.47	22.05
Gendarmenmarkt	97.45	101.77	100.11	74.76	71.39
Union Square	77.22	87.01	89.76	34.60	46.20
Vienna Cathedral	72.25	81.07	83.48	38.74	42.94
Roman Forum	103.59	105.73	108.88	52.05	82.30
Piccadilly	115.83	123.41	126.16	62.87	78.31
Trafalgar	126.43	126.49	126.5	108.19	115.90

Table A.3: Final Mean Normalized AUC on all 1DSfM [89] datasets after 20K iterations

A. Appendix

Dataset	<i>Mean Relative Error</i>				
	PGM 4D	PGM 6D	PGM 9D	Quat	MRP (Ours)
Ellis Island	12.21	11.49	10.37	11.87	9.03
NYC Library	14.29	12.94	11.51	13.67	9.30
Piazza del Popolo	21.91	21.24	20.64	20.74	13.49
Madrid Metropolis	20.43	19.84	17.85	19.62	17.09
Yorkminster	13.73	12.64	11.58	12.97	8.35
Montreal Notre Dame	12.5	11.59	9.58	11.93	6.22
Tower of London	12.41	12.24	12.44	11.56	8.71
Notre Dame	14.15	13.1	11.65	13.86	9.66
Alamo	14.23	17.47	15.75	13.17	9.78
Gendarmenmarkt	84.21	89.61	84.77	60.25	62.98
Union Square	44.44	55.4	55.94	19.98	15.52
Vienna Cathedral	41.8	45.62	44.18	26.64	20.32
Roman Forum	79.24	77.18	78.03	25.04	64.25
Piccadilly	74.25	105.15	122.06	38.61	46.21
Trafalgar	126.18	126.42	126.49	81.28	97.53

Table A.4: Final Mean Relative Error ($^{\circ}$) on all 1DSfM [89] datasets after 20K iterations

Dataset	# Nodes	# Edges	Mean Error	Median Error
Ellis Island	227	20K	12.52	2.89
NYC Library	332	21K	14.15	4.22
Piazza del Popolo	338	25K	8.4	1.81
Madrid Metropolis	341	24K	29.31	9.34
Yorkminster	437	28K	11.17	2.68
Montreal Notre Dame	450	52K	7.54	1.67
Tower of London	472	24K	11.6	2.59
Notre Dame	553	104K	14.16	2.7
Alamo	577	97K	9.1	2.78
Gendarmenmarkt	677	48K	33.33	12.3
Union Square	789	25K	9.03	3.61
Vienna Cathedral	836	103K	11.28	2.59
Roman Forum	1084	70K	13.84	2.97
Piccadilly	2152	309K	19.1	4.93
Trafalgar	5058	679K	8.64	3.01

Table A.5: Dataset sizes and observation accuracies ($^{\circ}$) for all 1DSfM [89] datasets

A.2 Definition and Properties of Cross-Pose

Relative placement tasks: In this paper, we are specifically interested in “relative placement tasks,” which we define here. Loosely speaking, a relative placement task is a task such that only the relationship between objects \mathcal{A} and \mathcal{B} is important for task success. Specifically, suppose that $\mathbf{T}_{\mathcal{A}}^*$ and $\mathbf{T}_{\mathcal{B}}^*$ are poses for objects \mathcal{A} and \mathcal{B} respectively (in some reference frame) for which a desired task is complete (lasagna is in the oven; mug is on the rack, etc). Then for a relative placement task, if objects \mathcal{A} and \mathcal{B} are in poses $\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}^*$ and $\mathbf{T} \cdot \mathbf{T}_{\mathcal{B}}^*$ (respectively) for any transform \mathbf{T} , then the task will also be complete. In other words, if $\mathbf{T}_{\mathcal{B}}^*$ represents the pose of the oven and $\mathbf{T}_{\mathcal{A}}^*$ represents the pose of the lasagna in that oven (at task completion); then if we transform the lasagna pose by \mathbf{T} and likewise transform the oven pose by \mathbf{T} , then the lasagna will still be located inside the oven.

Definition of Cross-Pose: In this section we will investigate the properties of

our definition of “cross-pose”, and explain why these properties are desirable. Let us start by assuming that the “cross-pose” function for objects \mathcal{A} and \mathcal{B} takes the form $f(\mathbf{P}_A, \mathbf{P}_B) \in SE(3)$, where \mathbf{P}_A and \mathbf{P}_B are the point clouds associated with objects \mathcal{A} and \mathcal{B} , respectively. For convenience of the below analysis, we overload the function f to also receive as input the poses $\mathbf{T}_A, \mathbf{T}_B$ of objects \mathcal{A} and \mathcal{B} respectively (with respect to a global reference frame); in other words, we define cross-pose such that $f(\mathbf{T}_A, \mathbf{T}_B) := f(\mathbf{P}_A, \mathbf{P}_B)$.

We would like our definition of “cross-pose” to have the following properties:

1) *Goal Consistency*: Suppose that \mathbf{T}_A^* and \mathbf{T}_B^* are the poses of objects \mathcal{A} and \mathcal{B} in a desired relative configuration that achieves the relative placement task. Then if both objects are transformed by the same transform \mathbf{T} , their cross-pose should be unchanged. Specifically, we define “goal consistency” as the following property:

$$f(\mathbf{T}_A^*, \mathbf{T}_B^*) = f(\mathbf{T} \cdot \mathbf{T}_A^*, \mathbf{T} \cdot \mathbf{T}_B^*)$$

for any transform $\mathbf{T} \in SE(3)$. This definition is consistent with the notion of success for a relative placement task defined above; if objects \mathcal{A} and \mathcal{B} are in a configuration such that the relative placement task is complete, then the cross-pose will be a constant value.

Let us define the cross-pose for which the task is complete as $f(\mathbf{T}_A^*, \mathbf{T}_B^*) = \mathbf{T}_{AB}^* \in SE(3)$. In our paper, we chose $\mathbf{T}_{AB}^* = \mathbf{I}$ where \mathbf{I} is the identity. We explain below why the identity is a natural choice.

Note that we do not require that this property holds true if the objects are not in the desired relative configuration; thus, if objects \mathcal{A} and \mathcal{B} are in arbitrary poses \mathbf{T}_A and \mathbf{T}_B respectively (where $\mathbf{T}_A \neq \mathbf{T}_A^*$ and $\mathbf{T}_B \neq \mathbf{T}_B^*$), then we do not require that $f(\mathbf{T}_A, \mathbf{T}_B) = f(\mathbf{T} \cdot \mathbf{T}_A, \mathbf{T} \cdot \mathbf{T}_B)$.

2) *Usability*: The purpose of estimating the cross-pose is to determine how to transform the objects into a configuration such that the relative placement task is successful; in other words, suppose that objects \mathcal{A} and \mathcal{B} have a cross-pose of $f(\mathbf{T}_A, \mathbf{T}_B)$. Then we wish to use the “cross-pose” $f(\mathbf{T}_A, \mathbf{T}_B)$ and the desired “cross-pose” $f(\mathbf{T}_A^*, \mathbf{T}_B^*) = \mathbf{T}_{AB}^*$ to compute a transform \mathbf{T}_Δ such that we can transform object \mathcal{A} by \mathbf{T}_Δ to achieve the desired configuration for the relative placement task. In other words, we wish to compute \mathbf{T}_Δ such that $\mathbf{T}_\Delta \cdot \mathbf{T}_A$ and \mathbf{T}_B are in a configuration

that completes the task, i.e. objects \mathcal{A} and \mathcal{B} will be in poses $\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}^*$ and $\mathbf{T} \cdot \mathbf{T}_{\mathcal{B}}^*$ (respectively) for some transform \mathbf{T} ; by the definition of the relative placement task above, this configuration is considered a task success. As we will see below, our definition of cross-pose allow us to compute \mathbf{T}_{Δ} easily.

Cross Pose Definitions and Its Properties The definition of “cross-pose” used in our method is:

$$f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}) := \mathbf{T}_{\mathcal{A}} \cdot \mathbf{T}_{\mathcal{B}}^{-1}. \quad (\text{A.1})$$

Using only this definition, we obtain the following properties:

$$f(\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}) = \mathbf{T} \cdot \mathbf{T}_{\mathcal{A}} \cdot \mathbf{T}_{\mathcal{B}}^{-1}, \quad f(\mathbf{T}_{\mathcal{A}}, \mathbf{T} \cdot \mathbf{T}_{\mathcal{B}}) = \mathbf{T}_{\mathcal{A}} \cdot \mathbf{T}_{\mathcal{B}}^{-1} \cdot \mathbf{T}^{-1}.$$

Without any extra assumptions, goal consistency does not hold for this definition of cross-pose; if both objects are transformed by the same transform, the resulting transform can differ from the original cross-pose:

$$f(\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}^*, \mathbf{T} \cdot \mathbf{T}_{\mathcal{B}}^*) = \mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}^* \cdot \mathbf{T}_{\mathcal{B}}^{*-1} \cdot \mathbf{T}^{-1} \neq \mathbf{T}_{\mathcal{A}}^* \cdot \mathbf{T}_{\mathcal{B}}^{*-1} = f(\mathbf{T}_{\mathcal{A}}^*, \mathbf{T}_{\mathcal{B}}^*).$$

In order to achieve the property of goal consistency, we can add the assumption that

$$f(\mathbf{T}_{\mathcal{A}}^*, \mathbf{T}_{\mathcal{B}}^*) = \mathbf{T}_{\mathcal{A}\mathcal{B}}^* = \mathbf{I} \quad (\text{A.2})$$

in the goal configuration. This implies that

$$f(\mathbf{T}_{\mathcal{A}}^*, \mathbf{T}_{\mathcal{B}}^*) = \mathbf{T}_{\mathcal{A}}^* \cdot \mathbf{T}_{\mathcal{B}}^{*-1} = \mathbf{I}.$$

Using this assumption, we then obtain that

$$f(\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}^*, \mathbf{T} \cdot \mathbf{T}_{\mathcal{B}}^*) = \mathbf{I},$$

which satisfies the definition of goal consistency, since we then have $f(\mathbf{T}_{\mathcal{A}}^*, \mathbf{T}_{\mathcal{B}}^*) = f(\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}^*, \mathbf{T} \cdot \mathbf{T}_{\mathcal{B}}^*) = \mathbf{I}$ for any transform $\mathbf{T} \in SE(3)$.

Next we check the usability property: suppose that objects \mathcal{A} and \mathcal{B} have a desired “cross-pose” of $f(\mathbf{T}_{\mathcal{A}}^*, \mathbf{T}_{\mathcal{B}}^*) := \mathbf{T}_{\mathcal{A}\mathcal{B}}^* = \mathbf{I}$. Let us assume that objects \mathcal{A} and \mathcal{B} have a current pose of $\mathbf{T}_{\alpha} \mathbf{T}_{\mathcal{A}}^*$ and $\mathbf{T}_{\beta} \mathbf{T}_{\mathcal{B}}^*$ respectively, for arbitrary transforms \mathbf{T}_{α} and

A. Appendix

$\mathbf{T}_\beta \in SE(3)$. Then the current “cross-pose” of objects \mathcal{A} and \mathcal{B} is:

$$f(\mathbf{T}_\alpha \mathbf{T}_\mathcal{A}^*, \mathbf{T}_\beta \mathbf{T}_\mathcal{B}^*) := \mathbf{T}_{\alpha\beta} = \mathbf{T}_\alpha \mathbf{T}_{\mathcal{A}\mathcal{B}}^* \mathbf{T}_\beta^{-1} = \mathbf{T}_\alpha \cdot \mathbf{T}_\beta^{-1}. \quad (\text{A.3})$$

Then we can compute a transform

$$\mathbf{T}_\Delta := \mathbf{T}_{\mathcal{A}\mathcal{B}}^* \cdot \mathbf{T}_{\alpha\beta}^{-1} = \mathbf{T}_\beta \cdot \mathbf{T}_\alpha^{-1};$$

such that if we transform object \mathcal{A} by \mathbf{T}_Δ then object \mathcal{A} will be in the pose

$$\mathbf{T}_\Delta \cdot \mathbf{T}_\alpha \cdot \mathbf{T}_\mathcal{A}^* = \mathbf{T}_\beta \cdot \mathbf{T}_\alpha^{-1} \cdot \mathbf{T}_\alpha \cdot \mathbf{T}_\mathcal{A}^* = \mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{A}^*.$$

Since object \mathcal{A} will be in the pose $\mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{A}^*$ (after applying transformation \mathbf{T}_Δ) and object \mathcal{B} is already in the pose $\mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{B}^*$, then the objects will now be in the desired relative configuration to complete the relative placement task. Note that, because of goal consistency, $\mathbf{T}_{\mathcal{A}\mathcal{B}}^*$ is a constant, and above we have set it equal to the identity \mathbf{I} , so in this case $\mathbf{T}_\Delta = \mathbf{T}_{\alpha\beta}^{-1}$, which is the inverse of the cross-pose between objects \mathcal{A} and \mathcal{B} (see Equation A.3). Thus we have shown the usability property: we can compute \mathbf{T}_Δ simply as the inverse of the cross-pose $\mathbf{T}_{\alpha\beta}^{-1}$; by transforming object \mathcal{A} by \mathbf{T}_Δ , we move the objects into the desired relative configuration, $\mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{A}^*$ and $\mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{B}^*$ that will complete the relative placement task.

Alternative properties: The cross-pose function defined has these properties:

$$f(\mathbf{T} \cdot \mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) = \mathbf{T} \cdot f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}), \quad f(\mathbf{T}_\mathcal{A}, \mathbf{T} \cdot \mathbf{T}_\mathcal{B}) = f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) \cdot \mathbf{T}^{-1} \quad (\text{A.4})$$

Now we ask whether we could instead define a cross-pose function that has the properties of

$$f(\mathbf{T} \cdot \mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) = \mathbf{T} \cdot f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}), \quad f(\mathbf{T}_\mathcal{A}, \mathbf{T} \cdot \mathbf{T}_\mathcal{B}) = \mathbf{T}^{-1} \cdot f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}). \quad (\text{A.5})$$

As it turns out, we cannot. Suppose that objects \mathcal{A} and \mathcal{B} initially have poses of $\mathbf{T}_\mathcal{A}$ and $\mathbf{T}_\mathcal{B}$ respectively, with a cross-pose of $f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B})$. If you transform object \mathcal{A} first by \mathbf{T}_α and then transform object \mathcal{B} first by \mathbf{T}_β , then the final “cross-pose” is

computed as follows:

$$\begin{aligned} f(\mathbf{T}_\alpha \cdot \mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) &= \mathbf{T}_\alpha \cdot f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) \\ f(\mathbf{T}_\alpha \cdot \mathbf{T}_\mathcal{A}, \mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{B}) &= \mathbf{T}_\beta^{-1} \cdot \mathbf{T}_\alpha \cdot f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) \end{aligned}$$

On the other hand, if you first transform \mathcal{B} by \mathbf{T}_β and then transform object \mathcal{A} first by \mathbf{T}_α , then the cross-pose would be computed as

$$\begin{aligned} f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{B}) &= \mathbf{T}_\beta^{-1} \cdot f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) \\ f(\mathbf{T}_\alpha \cdot \mathbf{T}_\mathcal{A}, \mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{B}) &= \mathbf{T}_\alpha \cdot \mathbf{T}_\beta^{-1} \cdot f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) \end{aligned}$$

Note that we now would have two definitions of $f(\mathbf{T}_\alpha \cdot \mathbf{T}_\mathcal{A}, \mathbf{T}_\beta \cdot \mathbf{T}_\mathcal{B})$ which are not equivalent, since $\mathbf{T}_\beta^{-1} \cdot \mathbf{T}_\alpha \neq \mathbf{T}_\alpha \cdot \mathbf{T}_\beta^{-1}$. Thus, we cannot define a cross-pose function to have the properties of Equation A.5 and instead we define our cross-pose function to have the properties of Equation A.4.

A.3 Translational Equivariance

One benefit of our method is that it is translationally equivariant by construction. This means that if the object point clouds, $\mathbf{P}_\mathcal{A}$ and $\mathbf{P}_\mathcal{B}$, are translated by random translation \mathbf{t}_α and \mathbf{t}_β , respectively, i.e. $\mathbf{P}_{\mathcal{A}'} = \mathbf{P}_\mathcal{A} + \mathbf{t}_\alpha$ and $\mathbf{P}_{\mathcal{B}'} = \mathbf{P}_\mathcal{B} + \mathbf{t}_\beta$, then the resulting corrected virtual correspondences, $\tilde{\mathbf{V}}_\mathcal{B}$ and $\tilde{\mathbf{V}}_\mathcal{A}$, respectively, are transformed accordingly, i.e. $\tilde{\mathbf{V}}_\mathcal{B} + \mathbf{t}_\beta$ and $\tilde{\mathbf{V}}_\mathcal{A} + \mathbf{t}_\alpha$, respectively, as we will show below. This results in an estimated cross-pose transformation that is also equivariant to translation by construction. This is achieved because our learned features and correspondence residuals are invariant to translation, and our virtual correspondence points are equivariant to translation.

First, our point features are a function of centered point clouds. That is, given

A. Appendix

point clouds \mathbf{P}_A and \mathbf{P}_B , the mean of each point cloud is computed as

$$\bar{p}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{P}_k.$$

This mean is then subtracted from the clouds,

$$\bar{\mathbf{P}}_k = \mathbf{P}_k - \bar{p}_k,$$

which centers the cloud at the origin. The features are then computed on the centered point clouds:

$$\Phi_k = g_k(\bar{\mathbf{P}}_k).$$

Since the point clouds are centered before features are computed, the features Φ_k are invariant to an arbitrary translation $\mathbf{P}_{k'} = \mathbf{P}_k + \mathbf{t}_k$.

These translationally invariant features are then used, along with the original point clouds, to compute “corrected virtual points” as a combination of virtual correspondence points, $v_i^{k'}$ and the correspondence residuals, $r_i^{k'}$. As we will see below, the “corrected virtual points” will be translationally equivariant by construction.

The virtual correspondence points, $v_i^{k'}$, are computed using weights that are a function of only the translationally invariant features, Φ_k :

$$w_i^{A' \rightarrow B'} = \text{softmax}(\Phi_{B'}^\top \phi_i^{A'}) = \text{softmax}(\Phi_B^\top \phi_i^A) = w_i^{A \rightarrow B};$$

thus the weights are also translationally invariant. These translationally invariant weights are applied to the translated cloud

$$v_i^{A' \rightarrow B'} = \mathbf{P}_{B'} w_i^{A \rightarrow B} = (\mathbf{P}_B + \mathbf{t}_\beta) w_i^{A \rightarrow B} = \sum_j p_{j,B} w_{i,j}^{A \rightarrow B} + \mathbf{t}_\beta \sum_j w_{i,j}^{A \rightarrow B} = \mathbf{P}_B w_i^{A \rightarrow B} + \mathbf{t}_\beta,$$

since $\sum_{j=1}^{N_B} w_{i,j}^{A \rightarrow B} = 1$. Thus the virtual correspondence points $v_i^{A' \rightarrow B'}$ are equivalently translated. The same logic follows for the virtual correspondence points $v_i^{B' \rightarrow A'}$. This gives us a set of translationally equivariant virtual correspondence points $v_i^{A' \rightarrow B'}$ and $v_i^{B' \rightarrow A'}$.

The correspondence residuals, $r_i^{k'}$, are a direct function of only the translationally invariant features Φ_k ,

$$r_i^{k'} = g_{\mathcal{R}}(\phi_i^{k'}) = g_{\mathcal{R}}(\phi_i^k) = r_i^k,$$

therefore they are also translationally invariant.

Since the virtual correspondence points are translationally equivariant, $v_i^{A' \rightarrow B'} = v_i^{A \rightarrow B} + \mathbf{t}_\beta$ and the correspondence residuals are translationally invariant, $r_i^{k'} = r_i^k$, the final corrected virtual correspondence points, $\tilde{v}_i^{A' \rightarrow B'}$, are translationally equivariant, i.e. $\tilde{v}_i^{A' \rightarrow B'} = v_i^{A \rightarrow B} + r_i^k + \mathbf{t}_\beta$. This also holds for $\tilde{v}_i^{B' \rightarrow A'}$, giving us the final translationally equivariant correspondences between the translated object clouds as $(\mathbf{P}_A + \mathbf{t}_\alpha, \tilde{\mathbf{V}}_B + \mathbf{t}_\beta)$ and $(\mathbf{P}_B + \mathbf{t}_\beta, \tilde{\mathbf{V}}_A + \mathbf{t}_\alpha)$, where $\tilde{\mathbf{V}}_B = [\tilde{v}_1^{A \rightarrow B} \dots \tilde{v}_{N_A}^{A \rightarrow B}]^\top$.

As a result, the final computed transformation will be automatically adjusted accordingly. Given that we use weighted SVD to compute the optimal transform, \mathbf{T}_{AB} , with rotational component \mathbf{R}_{AB} and translational component \mathbf{t}_{AB} , the optimal rotation remains unchanged if the point cloud is translated, $\mathbf{R}_{A'B'} = \mathbf{R}_{AB}$, since the rotation is computed as a function of the centered point clouds. The optimal translation is defined as

$$\mathbf{t}_{AB} := \bar{\tilde{v}}_{A \rightarrow B} - \mathbf{R}_{AB} \cdot \bar{p}_A,$$

where $\bar{\tilde{v}}_{A \rightarrow B}$ and \bar{p}_A are the means of the corrected virtual correspondence points, $\tilde{\mathbf{V}}_B$, and the object cloud \mathbf{P}_A , respectively, for object \mathcal{A} . Therefore, the optimal translation between the translated point cloud $\mathbf{P}_{A'}$ and corrected virtual correspondence points $\tilde{\mathbf{V}}^{A' \rightarrow B'}$ is

$$\begin{aligned} \mathbf{t}_{A'B'} &= \bar{\tilde{v}}_{A' \rightarrow B'} - \mathbf{R}_{AB} \cdot \bar{p}_{A'} \\ &= \bar{\tilde{v}}_{A \rightarrow B} + \mathbf{t}_\beta - \mathbf{R}_{AB} \cdot (\bar{p}_A + \mathbf{t}_\alpha) \\ &= \bar{\tilde{v}}_{A \rightarrow B} + \mathbf{t}_\beta - \mathbf{R}_{AB} \cdot \bar{p}_A - \mathbf{R}_{AB} \cdot \mathbf{t}_\alpha \\ &= \mathbf{t}_{AB} + \mathbf{t}_\beta - \mathbf{R}_{AB} \cdot \mathbf{t}_\alpha \end{aligned}$$

To simplify the analysis, if we assume that, for a given example, $\mathbf{R}_{AB} = \mathbf{I}$, then we get $\mathbf{t}_{A'B'} = \mathbf{t}_{AB} + \mathbf{t}_\beta - \mathbf{t}_\alpha$, demonstrating that the computed transformation is translation equivariant by construction.

A.4 Weighted SVD

The objective function for computing the optimal rotation and translation given a set of correspondences, $\{p_i^k \rightarrow \tilde{v}_i^k\}_i^{N_k}$ and weights $\{\alpha_i^k\}_i^{N_k}$, is as follows:

$$\mathcal{J}(\mathbf{T}_{AB}) = \sum_{i=1}^{N_A} \alpha_i^A \|\mathbf{T}_{AB} p_i^A - \tilde{v}_i^{A \rightarrow B}\|_2^2 + \sum_{i=1}^{N_B} \alpha_i^B \|\mathbf{T}_{AB}^{-1} p_i^B - \tilde{v}_i^{B \rightarrow A}\|_2^2$$

First we center (denoted with $*$) the point clouds and virtual points independently, and stack them into frame-specific matrices (along with weights) retaining their relative position and correspondence:

$$\mathbf{A} = \begin{bmatrix} \mathbf{P}_A^* & \tilde{\mathbf{V}}_A^* \end{bmatrix}^\top, \quad \mathbf{B} = \begin{bmatrix} \tilde{\mathbf{V}}_B^* & \mathbf{P}_B^* \end{bmatrix}^\top, \quad \mathbf{\Gamma} = \text{diag} \left(\begin{bmatrix} \alpha_A & \alpha_B \end{bmatrix} \right)$$

Then the minimizing rotation \mathbf{R}_{AB} is given by:

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \text{svd}(\mathbf{A}\mathbf{\Gamma}\mathbf{B}) \quad (\text{A.6a}) \quad \mathbf{R}_{AB} = \mathbf{U}\mathbf{\Sigma}_*\mathbf{V}^\top \quad (\text{A.6b})$$

where $\mathbf{\Sigma}_* = \text{diag}([1, 1, \dots, \det(\mathbf{U}\mathbf{V}^\top)])$ and svd is a differentiable SVD operation [54].

The optimal translation can be computed as:

$$\mathbf{t}_A = \bar{\tilde{\mathbf{v}}}_B - \mathbf{R}_{AB}\bar{\mathbf{p}}_A \quad \mathbf{t}_B = \bar{\mathbf{p}}_B - \mathbf{R}_{AB}\bar{\tilde{\mathbf{v}}}_A \quad \mathbf{t} = \frac{N_A}{N}\mathbf{t}_A + \frac{N_B}{N}\mathbf{t}_B \quad (\text{A.7a})$$

with $N = N_A + N_B$. In the special translation-only case, the optimal translation can be computed by setting \mathbf{R}_{AB} to identity in above equations. The final transform can be assembled:

$$\mathbf{T}_{AB} = \begin{bmatrix} \mathbf{R}_{AB} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \quad (\text{A.8})$$

A.5 PartNet-Mobility Objects Placement Task Details

In this section, we describe the PartNet-Mobility Objects Placement experiments in details.

A.5.1 Dataset Preparation

Simulation Setup. We leverage the PartNet-Mobility dataset [90] to find common household objects as the anchor object for TAX-Pose prediction. The selected subset of the dataset contains 8 categories of objects. We split the objects into 54 seen and 14 unseen instances. During training, for a specific task of each of the seen objects, we generate an action-anchor objects pair by randomly sampling transformations from $SE(3)$ as cross-poses. The action object is chosen from the Ravens simulator’s rigid body objects dataset [95]. We define a subset of four tasks (“In”, “On”, “Left” and “Right”) for each selected anchor object. Thus, there exists a ground-truth cross-pose (defined by human manually) associated with each defined specific task. We then use the ground-truth TAX-Poses to supervise each task’s TAX-Pose prediction model. For each observation action-anchor objects pair, we sample 100 times using the aforementioned procedure for the training and testing datasets.

Real-World Setup. In real-world, we select a set of anchor objects: Drawer, Fridge, and Oven and a set of action objects: Block and Bowl. We test 3 (“In”, “On”, and “Left”) TAX-Pose models in real-world without retraining or finetuning. The point here is to show the method capability of generalizing to unseen real-world objects.

A.5.2 Metrics

Simulation Metrics. In simulation, with access to the object’s ground-truth pose, we are able to quantitatively calculate translational and rotation error of the TAX-Pose prediction models. Thus, we report the following metrics on a held-out set of anchor objects in simulation:

Translational Error: The L2 distance between the inferred cross-pose translation ($\mathbf{t}_{AB}^{\text{pred}}$) and the ground-truth pose translation ($\mathbf{t}_{AB}^{\text{GT}}$).

$$\mathcal{E}_{\mathbf{t}} = \|\mathbf{t}_{AB}^{\text{pred}} - \mathbf{t}_{AB}^{\text{GT}}\|_2$$

Rotational Error: The geodesic $SO(3)$ distance [32, 39] between the predicted cross-pose rotation ($\mathbf{R}_{AB}^{\text{pred}}$) and the ground-truth rotation ($\mathbf{R}_{AB}^{\text{GT}}$).

$$\mathcal{E}_{\mathbf{R}} = \frac{1}{2} \arccos \left(\frac{\text{tr}(\mathbf{R}_{AB}^{\text{pred}\top} \mathbf{R}_{AB}^{\text{GT}}) - 1}{2} \right)$$

Real-World Metrics. In real-world, due to the difficulty of defining ground-truth TAX-Pose, we instead manually, qualitatively define goal “regions” for each of the anchor-action pairs. The goal-region should have the following properties:

[noitemsep]The predicted TAX-Pose of the action object should appear visually correct. For example, if the specified task is “In”, then the action object should be indeed contained within the anchor object after being transformed by predicted TAX-Pose. The predicted TAX-Pose of the action object should not violate physical constraints of the workspace and of the relation between the action and anchor objects. Specifically, the action object should not interfere with/collide with the anchor object after being transformed by the predicted TAX-Pose. See Fig. A.4 for an illustration of TAX-Pose predictions that fail to meet this criterion.

A.5.3 Motion Planning

In both simulated and real-world experiments, we use off-the-shelf motion-planning tools to find a path between the starting pose and goal pose of the action object.

Simulation. To actuate the action object from its starting pose \mathbf{T}_0 to its goal pose transformed by the predicted TAX-Pose $\hat{\mathbf{T}}_{AB}\mathbf{T}_0$, we plan a path free of collision. Learning-based methods such as [21] deal with collision checking with point clouds by training a collision classifier. A more data-efficient method is by leveraging computer graphics techniques, transforming the point clouds into marching cubes [46], which can then be used to efficiently reconstruct meshes. Once the triangular meshes are reconstructed, we can deploy off-the-shelf collision checking methods such as FCL [53] to detect collisions in the planned path. Thus, in our case, we use position

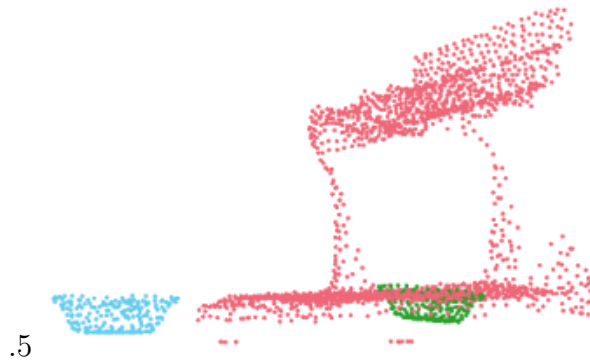


Figure A.2: **Failure of "In" prediction.** Predicted TAX-Pose violates the physical constraints by penetrating the oven base too much.

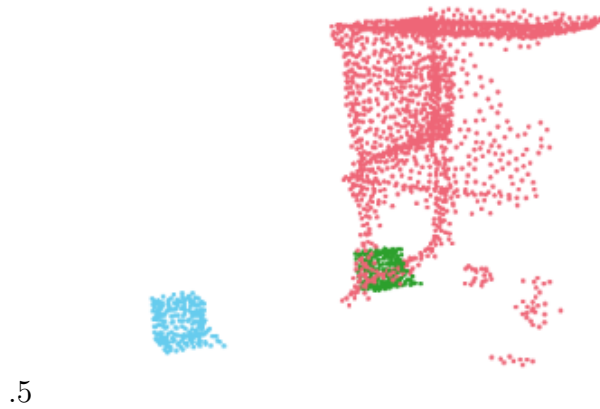


Figure A.3: **Failure of "Left" prediction.** Predicted TAX-Pose violates the physical constraints by being in collision with the leg of the drawer.

Figure A.4: **An illustration of unsuccessful real-world TAX-Pose predictions.** In both subfigures, red points represent the anchor object, blue points represent action object's starting pose, and green points represent action object's predicted pose.

control to plan a trajectory of the action object \mathcal{A} to move it from its starting pose to the predicted goal pose. We use OMPL [70] as the motion planning tool and the constraint function passed into the motion planner is from the output of FCL after converting the point clouds to meshes via marching cubes.

Real World. In real-world experiments, we need to resolve several practical issues to make TAX-Pose prediction model viable. First, we do not have access to a mask that labels action and anchor objects. Thus, we manually define a mask by using a threshold value of y -coordinate to automatically detect discontinuity in y -coordinates, representing the gap of spacing between action and anchor objects upon placement. Next, grasping action objects is a non-trivial task. Since, we are only using 2 action objects (a cube and a bowl), we manually define a grasping primitive for each action object. This is done by hand-picking an offset from the centroid of the action object before grasping, and an approach direction after the robot reaches the pre-grasp pose to make contacts with the object of interest. The offsets are chosen via kinesthetic teaching on the robot when the action object is under identity rotation (canonical pose). Finally, we need to make an estimation of the action’s starting pose for motion planning. This is done by first statistically cleaning the point cloud [26] of the action object, and then calculating the centroid of the action object point cloud as the starting position. For starting rotation, we make sure the range of the rotation is not too large for the pre-defined grasping primitive to handle. Another implementation choice here is to use ICP [8] calculate a transformation between the current point cloud to a pre-scanned point cloud in canonical (identity) pose. We use the estimated starting pose to guide the pre-defined grasp primitive. Once a successful grasp is made, the robot end-effector is rigidly attached to the action object, and we can then use the same predicted TAX-Pose to calculate the end pose of the robot end effector, and thus feed the two poses into MoveIt! to get a full trajectory in joint space. Note here that the collision function in motion planning is comprised of two parts: workspace and anchor object. That is, we first reconstruct the workspace using boxes to avoid collision with the table top and camera mount, and we then reconstruct the anchor object in RViz using Octomap [38] using the cleaned anchor object point cloud. In this way, the robot is able to avoid collision with the anchor object as well.

A.5.4 Baselines Description

In simulation, we compare our method to a variety of baseline methods.

E2E Behavioral Cloning: Generate motion-planned trajectories using OMPL that take the action object from start to goal. These serve as “expert” trajectories for Behavioral Cloning (BC). We then use a PointNet++ network to output a sparse policy that, at each time step, takes as input the point cloud observation of the action and anchor objects and outputs an incremental 6-DOF transformation that imitates the expert trajectory. The 6-DoF transformation is expressed using Euclidean xyz translation and rotation quaternion. The “prediction” is the final achieved pose of the action object at the terminal state.

E2E DAgger: Using the same BC dataset and the same PointNet++ architecture as above, we train a sparse policy that outputs the same transformation representation as in BC using DAgger [59]. The “prediction” is the final achieved pose of the action object at the terminal state.

Trajectory Flow: Using the same BC dataset with DAgger, we train a dense policy using PointNet++ to predict a dense per-point 3D flow vector at each time step instead of a single incremental 6-DOF transformation. Given this dense per-point flow, we add the per-point flow to each point of the current time-step’s point cloud, and we are able to extract a rigid transformation between the current point cloud and the point cloud transformed by adding per-point flow vectors using SVD, yielding the next pose. The “prediction” is the final achieved pose of the action object at the terminal state.

Goal Flow: Instead of training a multi-step sparse/dense policy to reach the goal, train a PointNet++ network to output a single dense flow prediction which assigns a per-point 3D flow vector that points from each action object point from its starting pose directly to its corresponding goal location. Given this dense per-point flow, we add the per-point flow to each point of the start point cloud, and we are able to extract a rigid transformation between the start point cloud and the point cloud transformed by adding per-point flow vectors using SVD, yielding goal pose. We pass the start and goal pose into a motion planner (OMPL) and execute the planned trajectory. The “prediction” is thus given by the SVD output.

A. Appendix

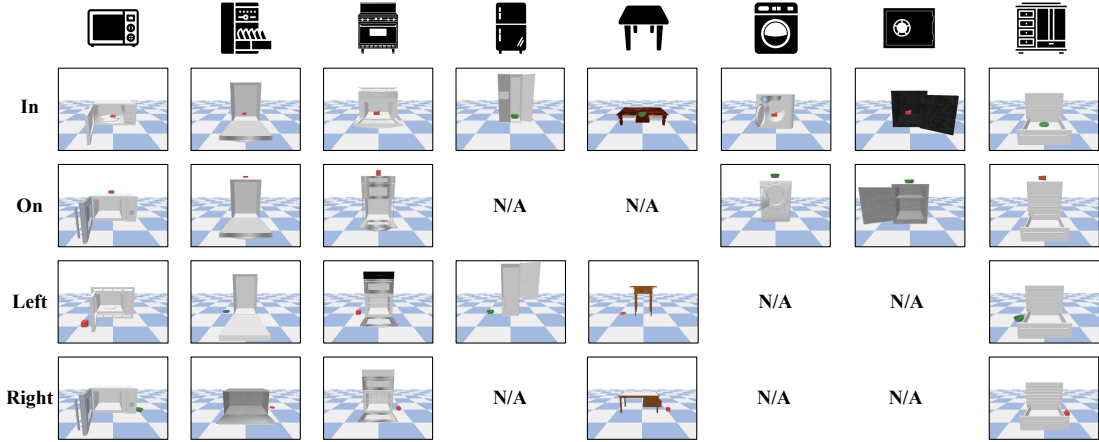


Figure A.5: A visualization of all categories of anchor objects and associated semantic tasks, with action objects in ground-truth TAX-Poses used in simulation training.

		AVG.																	
		\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t								
Baselines	E2E BC	42.37	0.69	40.49	0.80	50.79	0.59	48.02	0.61	30.69	1.09	36.59	0.81	48.48	0.42	41.42	0.84	42.49	0.37
	E2E DAgger [59]	36.06	0.67	38.57	0.68	43.99	0.63	42.34	0.57	24.87	0.96	30.87	0.90	42.96	0.46	29.79	0.83	35.08	0.33
Ablations	Traj. Flow [26]	34.48	0.65	35.39	0.85	43.42	0.63	35.51	0.60	28.26	0.80	27.67	0.68	25.91	0.44	43.59	0.82	36.05	0.36
	Goal Flow [26]	27.49	0.21	25.41	0.08	31.07	0.13	27.05	0.27	27.80	0.11	29.02	0.38	19.22	0.36	31.56	0.18	28.81	0.19
Ours	TAX-Pose	11.74	0.23	5.81	0.11	1.82	0.08	5.92	0.11	3.67	0.07	19.54	0.41	7.96	0.63	5.96	0.12	43.27	0.33

Table A.6: Goal Inference Rotational and Translational Error Results (\downarrow) for the “In” Goal. Rotational errors (\mathcal{E}_R) are in degrees ($^\circ$) and translational errors (\mathcal{E}_t) are in meters (m). The lower the better.

		AVG.													
		\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t						
Baselines	E2E BC	42.69	0.74	41.94	0.74	36.70	0.52	38.23	0.73	41.69	1.10	48.57	0.75	48.98	0.63
	E2E DAgger [59]	37.68	0.70	39.24	0.69	31.63	0.54	41.06	0.68	37.72	1.03	35.94	0.75	40.47	0.51
Ablations	Traj. Flow [26]	35.13	0.76	34.78	0.70	39.14	0.59	31.10	0.69	33.07	0.97	35.61	0.71	37.09	0.87
	Goal Flow [26]	22.10	0.20	27.82	0.26	20.43	0.09	34.66	0.10	22.71	0.12	26.48	0.27	0.48	0.32
Ours	TAX-Pose	4.45	0.12	4.21	0.12	2.29	0.10	2.73	0.09	5.77	0.10	5.81	0.13	5.89	0.19

Table A.7: Goal Inference Rotational and Translational Error Results (\downarrow) for the “On” Goal. Rotational errors (\mathcal{E}_R) are in degrees ($^\circ$) and translational errors (\mathcal{E}_t) are in meters (m). The lower the better.







		AVG.													
		\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t			\mathcal{E}_R	\mathcal{E}_t		
Baselines	E2E BC	44.87	0.74	30.95	0.89	36.86	0.72	56.86	0.52	34.35	1.03	31.69	0.77	46.86	0.78
	E2E DAgger [59]	41.32	0.68	31.40	0.84	38.49	0.73	47.64	0.51	36.47	0.99	27.72	0.73	39.83	0.51
Ablations	Traj. Flow [26]	38.85	0.58	31.87	1.07	39.48	0.44	39.48	0.44	28.71	0.69	41.06	0.73	40.70	0.31
	Goal Flow [26]	29.64	0.10	28.51	0.10	26.33	0.08	32.96	0.07	27.42	0.10	22.04	0.09	27.42	0.15
Ours	TAX-Pose	6.02	0.17	12.73	0.28	1.59	0.11	2.91	0.12	4.41	0.08	12.12	0.34	6.38	0.12

Table A.8: Goal Inference Rotational and Translational Error Results (\downarrow) for the “Left” Goal. Rotational errors (\mathcal{E}_R) are in degrees ($^\circ$) and translational errors (\mathcal{E}_t) are in meters (m). The lower the better.







		AVG.											
		\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t	\mathcal{E}_R	\mathcal{E}_t			\mathcal{E}_R	\mathcal{E}_t
Baselines	E2E BC	39.11	0.76	37.89	0.86	24.26	0.77	36.27	0.88	52.86	0.48	44.26	0.78
	E2E DAgger [59]	36.80	0.73	27.40	0.84	32.31	0.74	32.61	0.82	49.27	0.46	42.40	0.78
Ablations	Traj. Flow [26]	35.33	0.71	22.93	0.66	34.78	1.22	31.29	0.92	42.71	0.37	44.93	0.36
	Goal Flow [26]	27.34	0.16	21.79	0.15	22.37	0.28	27.79	0.15	32.96	0.07	31.79	0.15
Ours	TAX-Pose	4.33	0.13	4.64	0.14	2.48	0.11	3.91	0.15	6.47	0.17	4.17	0.08

Table A.9: Goal Inference Rotational and Translational Error Results (\downarrow) for the “Right” Goal. Rotational errors (\mathcal{E}_R) are in degrees ($^\circ$) and translational errors (\mathcal{E}_t) are in meters (m). The lower the better.

A.5.5 Per-Task Results

In the main body of the paper, we have shown the meta-results of the performance of each method by averaging the quantitative metrics for each sub-task (“In”, “On”, “Left”, and “Right” in simulation and “In”, “On” and “Left” in real-world). Here we show each sub-task’s results in Table A.6¹, Table A.7, Table A.8, and Table A.9 respectively.

As mentioned above, not all anchor objects have all 4 tasks due to practical reasons. For example, the doors of safes might occlude the action object completely

¹Categories from left to right: microwave, dishwasher, oven, fridge, table, washing machine, safe, and drawer.

A. Appendix

and it is impossible to show the action object in the captured image under “Left” and “Right” tasks (due to handedness of the door); a table’s height might be too tall for the camera to see the action object under the “Top” task. Under this circumstance, for sake of simplicity and consistency, we define a subset of the 4 goals for each object such that the anchor objects of the same category have consistent tasks definitions. We show a collection of visualizations of each task defined for each category in Fig. A.5.

Moreover, we also show per-task success rate for real-world experiments in Table A.10.










	In			On			Left		
									
Goal Flow	0.00	0.10	0.30	0.05	N/A	0.20	0.50	0.65	0.60
TAX-Pose	1.00	1.00	0.85	1.00	N/A	1.00	0.85	0.90	0.70

Table A.10: Combined per-task results for real-world goal placement success rate.

Bibliography

- [1] Gabriel Agamennoni, Simone Fontana, Roland Y Siegwart, and Domenico G Sorrenti. Point clouds registration with probabilistic data association. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4092–4098. IEEE, 2016. [3.2](#)
- [2] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019. [2.2.1](#)
- [3] Simon L Altmann. Hamilton, rodrigues, and the quaternion scandal. *Mathematics Magazine*, 62(5):291–308, 1989. [2.4.1](#)
- [4] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. [3.2](#)
- [5] Raman Arora. On learning rotations. *Advances in neural information processing systems*, 22:55–63, 2009. [2.2.2](#)
- [6] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995. [3.2](#)
- [7] Amir Beck and Shoham Sabach. Weiszfeld’s method: Old and new results. *Journal of Optimization Theory and Applications*, 164(1):1–40, 2015. [2.2.1](#)
- [8] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992. [3.2](#), [A.5.3](#)
- [9] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. Technical report, Springer, 2008. [3.2](#)
- [10] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Praseon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. [3.2](#)
- [11] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE*

- Transactions on Automatic Control*, 58(9):2217–2229, 2013. [2.2.2](#)
- [12] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. In *Computer graphics forum*, volume 32, pages 113–123. Wiley Online Library, 2013. [3.2](#)
- [13] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014. [3.2](#)
- [14] Romain Brégier. Deep regression on manifolds: a 3d rotation case study. *arXiv preprint arXiv:2103.16317*, 2021. [2.2.2](#)
- [15] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. [3.4.3](#)
- [16] Samprit Chatterjee and Eugene Seneta. Towards consensus: Some convergence theorems on repeated averaging. *Journal of Applied Probability*, 14(1):89–97, 1977. [2.2.1](#)
- [17] Jiayi Chen, Yingda Yin, Tolga Birdal, Baoquan Chen, Leonidas Guibas, and He Wang. Projective manifold gradient layer for deep rotation regression. *arXiv preprint arXiv:2110.11657*, 2021. [2.2.2](#), [??](#), [??](#), [??](#), [2.8.1](#), [2.8.1](#), [??](#), [2.8.2](#), [A.1](#)
- [18] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016. [3.5.2](#)
- [19] John Crassidis and F Markley. Attitude estimation using modified rodrigues parameters. In *Flight Mechanics/Estimation Theory Symposium*, pages 71–86. NASA, 1996. [2.2.2](#), [2.6.1](#)
- [20] John L Crassidis and F Landis Markley. Sliding mode control using modified rodrigues parameters. *Journal of Guidance, Control, and Dynamics*, 19(6):1381–1383, 1996. [2.4.1](#)
- [21] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. Object rearrangement using learned implicit collision functions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6010–6017. IEEE, 2021. [A.5.3](#)
- [22] Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974. [2.2.1](#)
- [23] Frank Dellaert, David M Rosen, Jing Wu, Robert Mahony, and Luca Carlone. Shonan rotation averaging: Global optimality by surfing $so(p)^n$. In *European Conference on Computer Vision*, pages 292–308. Springer, 2020. [2.1](#), [2.2.1](#)

- [24] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulencard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021. [3.4.3](#)
- [25] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv preprint arXiv:1802.10367*, 2018. [2.2.2](#)
- [26] Ben Eisner*, Harry Zhang*, and David Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. In *Robotics: Science and Systems (RSS)*, 2022. [??](#), [??](#), [A.5.3](#), [??](#), [??](#), [??](#), [??](#), [??](#), [??](#), [??](#)
- [27] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018. [3.2](#), [3.5.2](#), [3.5.2](#), [??](#), [3.5.2](#), [??](#)
- [28] Jared Glover and Leslie Pack Kaelbling. Tracking 3-d rotations with the quaternion bingham filter. 2013. [2.4.1](#)
- [29] Dirk Hähnel and Wolfram Burgard. Probabilistic matching for 3d scan registration. In *Proc. of the VDI-Conference Robotik*, volume 2002. Citeseer, 2002. [3.2](#)
- [30] Brian C Hall. Lie groups, lie algebras, and representations. In *Quantum Theory for Mathematicians*, pages 333–366. Springer, 2013. [2.4.3](#), [2.4.3](#)
- [31] Richard Hartley, Khuram Aftab, and Jochen Trumpf. L1 rotation averaging using the weiszfeld algorithm. In *CVPR 2011*, pages 3041–3048. IEEE, 2011. [2.2.1](#)
- [32] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International journal of computer vision*, 103(3):267–305, 2013. [2.1](#), [2.2.1](#), [2.4.2](#), [2.4.3](#), [2.5.1](#), [3.5.1](#), [A.5.2](#)
- [33] 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. [2.8.2](#)
- [34] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020. [3.2](#)
- [35] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3003–3013, 2021. [3.2](#)

- [36] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012. [3.2](#)
- [37] Timo Hinzmann, Thomas Stastny, Gianpaolo Conte, Patrick Doherty, Piotr Rudol, Marius Wzorek, Enric Galceran, Roland Siegwart, and Igor Gilitschenski. Collaborative 3d reconstruction using heterogeneous uavs: System and experiments. In *International Symposium on Experimental Robotics*, pages 43–56. Springer, 2016. [3.2](#)
- [38] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013. [3.5.1](#), [A.5.3](#)
- [39] Du Q Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009. [3.5.1](#), [A.5.2](#)
- [40] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5974–5983, 2017. [2.2.2](#)
- [41] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. [2.2.2](#)
- [42] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. [2.8.2](#)
- [43] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020. [2.1](#), [2.2.2](#)
- [44] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *arXiv preprint arXiv:2006.14616*, 2020. [2.2.2](#), [??](#), [??](#), [??](#), [2.8.1](#)
- [45] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018. [2.2.2](#), [3.4.2](#)
- [46] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. [A.5.3](#)
- [47] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999. [3.2](#)

- [48] Siddharth Mahendran, Haider Ali, and René Vidal. 3d pose regression using convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2174–2182, 2017. [2.5.1](#)
- [49] Jonathan H Manton. A globally convergent numerical algorithm for computing the centre of mass on compact lie groups. In *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.*, volume 3, pages 2211–2216. IEEE, 2004. [2.2.1](#), [2.5.2](#)
- [50] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpm: Keypoint affordances for category-level robotic manipulation. *International Symposium on Robotics Research (ISRR) 2019*, 2019. [2.1](#), [2.5](#), [3.2](#)
- [51] Alex Olshevsky and John N Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM journal on control and optimization*, 48(1):33–55, 2009. [2.2.1](#)
- [52] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. [3.4.3](#), [3.4.3](#)
- [53] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866. IEEE, 2012. [A.5.3](#)
- [54] Théodore Papadopoulos and Manolis IA Lourakis. Estimating the jacobian of the singular value decomposition: Theory and applications. In *European Conference on Computer Vision*, pages 554–570. Springer, 2000. [3.4.1](#), [A.4](#)
- [55] Valentin Peretroukhin, Matthew Giamou, David M Rosen, W Nicholas Greene, Nicholas Roy, and Jonathan Kelly. A smooth representation of belief over so(3) for deep rotation learning with uncertainty. In *Proceedings of Robotics: Science and Systems (RSS'20)*, 2020. [2.2.2](#)
- [56] G Piova, I Shames, B Fidan, F Bullo, and B Anderson. On frame and orientation localization for relative sensing networks. In *IEEE Conference on Decision and Control*, 2008. [2.2.1](#)
- [57] Zengyi Qin, Kuan Fang, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Keto: Learning keypoint representations for tool manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7278–7285. IEEE, 2020. [3.2](#)
- [58] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018. [3.2](#)
- [59] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation

- learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. [3.2](#), [3.5.1](#), [??](#), [A.5.4](#), [??](#), [??](#), [??](#), [??](#)
- [60] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International journal of computer vision*, 66(3):231–259, 2006. [3.2](#)
- [61] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. [3.2](#)
- [62] Wm Joshua Russell, Daniel J Klein, and Joao P Hespanha. Optimal estimation on the graph cycle space. *IEEE Transactions on Signal Processing*, 59(6):2834–2846, 2011. [2.2.1](#)
- [63] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999. [3.2](#)
- [64] Hanspeter Schaub, John L Junkins, et al. Stereographic orientation parameters for attitude dynamics: A generalization of the rodrigues parameters. *Journal of the Astronautical Sciences*, 44(1):1–19, 1996. [2.4.1](#)
- [65] Max Schwarz, Hannes Schulz, and Sven Behnke. Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 1329–1335. IEEE, 2015. [3.2](#)
- [66] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009. [3.2](#)
- [67] Malcolm D Shuster et al. A survey of attitude representations. *Navigation*, 8(9):439–517, 1993. [2.4.1](#)
- [68] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. *arXiv preprint arXiv:2112.05124*, 2021. [3.2](#), [3.5.2](#), [3.5.2](#), [??](#), [3.5.2](#), [??](#)
- [69] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015. [2.2.2](#)
- [70] Ioan A Sucas, Mark Moll, and Lydia E Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012. [A.5.3](#)

- [71] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *ECCV*, 2018. [2.1](#), [2.5](#)
- [72] Supasorn Suwajanakorn, Noah Snavely, Jonathan Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *NeurIPS*, 2018. [2.1](#), [2.5](#)
- [73] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10338–10347, 2021. [2.2.2](#)
- [74] George Terzakis, Manolis Lourakis, and Djamel Ait-Boudaoud. Modified rodrigues parameters: an efficient representation of orientation in 3d vision and graphics. *Journal of Mathematical Imaging and Vision*, 60(3):422–442, 2018. [2.2.2](#), [2.4.1](#), [2.6.1](#)
- [75] Carlo Tomasi. Vector representation of rotations. *Computer Science*, 527, 2013. [2.4.3](#)
- [76] Roberto Tron, René Vidal, and Andreas Terzis. Distributed pose averaging in camera networks via consensus on $se(3)$. In *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–10. IEEE, 2008. [2.2.1](#)
- [77] Roberto Tron, Bijan Afsari, and René Vidal. Average consensus on riemannian manifolds with bounded curvature. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 7855–7862. IEEE, 2011. [2.2.1](#)
- [78] Roberto Tron, Bijan Afsari, and René Vidal. Intrinsic consensus on $so(3)$ with almost-global convergence. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 2052–2058. IEEE, 2012. [2.2.1](#)
- [79] Panagiotis Tsiotras and James M Longuski. A new parameterization of the attitude kinematics. *Journal of the Astronautical Sciences*, 43(3):243–262, 1995. [2.4.1](#)
- [80] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991. [3.2](#)
- [81] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. [3.4.3](#)
- [82] Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *arXiv preprint arXiv:1912.02175*, 2019. [2.2.1](#)

- [83] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densfusion: 6d object pose estimation by iterative dense fusion. 2019. [2.1](#)
- [84] Lanhui Wang and Amit Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference: A Journal of the IMA*, 2(2):145–193, 2013. [2.1](#), [2.2.1](#)
- [85] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019. [3.2](#), [3.4](#), [3.4.3](#)
- [86] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. [3.4.3](#)
- [87] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E Bekris. se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10367–10373. IEEE, 2020. [2.2.2](#)
- [88] Thomas F Wiener. *Theoretical analysis of gimballed inertial reference equipment using delta-modulated instruments*. PhD thesis, Massachusetts Institute of Technology, 1962. [2.6.1](#)
- [89] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. ([document](#)), [2.5](#), [2.3](#), [2.8.1](#), [A.1](#), [A.1](#), [A.1](#), [A.2](#), [A.3](#), [A.4](#), [A.5](#)
- [90] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, and Others. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020. [3.5.1](#), [A.5.1](#)
- [91] 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. [3.2](#), [3.4.2](#)
- [92] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018. ([document](#)), [2.1](#), [2.2.2](#), [2.8.2](#), [2.9a](#), [2.9b](#), [2.9](#)
- [93] Shing-Tung Yau. Non-existence of continuous convex functions on certain riemannian manifolds. *Mathematische Annalen*, 207(4):269–270, 1974. [2.4.4](#)
- [94] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani,

- et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020. [3.5.1](#)
- [95] Vicky Zeng, Timothy E Lee, Jacky Liang, and Oliver Kroemer. Visual identification of articulated object parts. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2443–2450. IEEE, 2020. [A.5.1](#)
- [96] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European conference on computer vision*, pages 766–782. Springer, 2016. [2.1](#)
- [97] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. [2.2.2](#), [??](#), [??](#), [??](#), [2.8.1](#)