Learning Articulated Motions From Visual Demonstration

by

Sudeep Pillai

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

Author .....
Department of
Electrical Engineering and Computer Science
Apr 30, 2014

Certified by .....
Seth Teller
Professor
Thesis Supervisor

Accepted by .....
Leslie Kolodziejski
Chairman, Department Committee on Graduate Students

To my parents

Learning Articulated Motions

From Visual Demonstration

by

Sudeep Pillai

Submitted to the Department of
Electrical Engineering and Computer Science
on Apr 30, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Robots operating autonomously in household environments must be capable of interacting with articulated objects on a daily basis. They should be able to infer each object's underlying kinematic linkages purely by observing its motion during manipulation. This work proposes a framework that enables robots to learn the articulation in objects from user-provided demonstrations, using RGB-D sensors. We introduce algorithms that combine concepts in sparse feature tracking, motion segmentation, object pose estimation, and articulation learning, to develop our proposed framework. Additionally, our methods can predict the motion of previously seen articulated objects in future encounters. We present experiments that demonstrate the ability of our method, given RGB-D data, to identify, analyze and predict the articulation of a number of everyday objects within a human-occupied environment.

Thesis Supervisor: Seth Teller
Title: Professor

Acknowledgments

I would like to thank my advisor, Seth Teller, for the countless sessions of advice and ideas we have had. His perspectives on tackling more challenging, and real-world problems have constantly pushed my limits to dream bigger. I would like to especially thank Matt Walter and Sachi Hemachandra for their invaluable feedback, and collaboration on several aspects of my research. Many thanks to Mike Fleder, Jon Brookshire, David Hayden, Ross Finman, Nick Wang, and William Li for their thoughts and ideas over several discussions.

I would like to especially thank my parents for the unconditional love and endless support they have provided throughout my life. I thank my brothers, and their wives for being a constant source of inspiration, and support. To my two nephews, and soon-to-be-born niece, you bring a huge smile to my face every time I think of you; I couldn't ask for more.

Finally, I would like to thank my friends; being surrounded by their compassion and laughter has made my experience here at MIT all the more enjoyable.

Contents

1 Introduction **18**

1.1 Thesis Overview 20

1.1.1 Learning from Visual Demonstration 20

1.1.2 Implementation 21

1.1.3 Experiments and Analysis 21

1.2 Contributions 21

2 Related Work **24**

2.1 Background 24

3 Articulation Learning from Visual Demonstration **30**

3.1 Spatio-Temporal Feature Tracking 32

3.1.1 Trajectories via Interest-Point Tracking and Learning 33

3.2 Motion Segmentation 42

3.2.1 Trajectory Matching 42

3.2.2 Trajectory Clustering 45

3.3 Multi-Rigid-Body Pose Optimization 50

3.3.1 Notation 50

3.3.2 Pose Estimation 51

3.3.3 Pose Optimization 52

3.4 Articulation Learning 54

3.4.1 Notation 54

3.4.2 Problem Definition 54

3.4.3 Candidate Models and Model Fitting 55
3.4.4 Structure Selection 56
3.5 Learning to Predict Articulation 58
3.5.1 Object Instance Recognition and Prediction 58
3.5.2 Qualitative Results 61

4 Experiments and Analysis **65**
4.1 Data and Experimental Setup 65
4.2 Algorithm Evaluation 65
4.2.1 Learning with Simulated Data 66
4.2.2 Learning with RGB-D sensors 76
4.2.3 Articulation Prediction 86

5 Conclusion **90**

A Implementation **92**

List of Figures

1-1 The proposed framework reliably learns the underlying kinematic model of articulated objects from user-provided visual demonstrations, and subsequently predicts their motions at future encounters, from novel vantage points. 19

3-1 The architecture of our system for articulation learning from visual demonstration. 31

3-2 The training phase. 32

3-3 The prediction phase. 32

3-4 Spatio-Temporal Feature Tracking steps combine traditional dense optical flow methods with feature detection and matching techniques to construct long range trajectories with little to no drift. 33

3-5 Trajectories constructed using KLT [29] (**left**), Dense Trajectories [34] (**middle**), and our algorithm (**right**). While all 3 algorithms show reasonably good trajectory construction, our algorithm exhibits negligible drift while achieving longer trajectories. 34

3-6 Illustration of feature detection and prediction for a checkerboard pattern undergoing motion. (a) Feature detections (via GFTT) and feature predictions (via Dense Optical Flow) are indicated by blue points and blue edges respectively. (b) Gray points indicate the feature detections in the subsequent frame. Gray circles indicate the regions where feature predictions continue to be valid. 36

3-7 Illustration of feature matching and addition. (a) Features are successfully matched by comparing the SURF descriptions of putative matches. Successful matches are colored in green, while those not matched are grayed out (b) Successfully matched features are added to the feature tracks (in green). Subsequently, new features are added in regions that lack matches (in yellow). 37

3-8 The proposed feature tracking algorithm shows negligible drift, compared to KLT (**left**), and Dense Trajectories (**middle**), while achieving longer trajectories. 39

3-9 Extensions to 3-D: Feature Detections are extended to 3-D with the readily available point cloud representation, and the surface normal is estimated via integral depth images. **Top Left:** A typical RGB Image, **Top Right:** Point cloud with co-registered RGB image. **Bottom Left:** Surface normals computed via integral depth images. **Bottom Right:** The resulting interest-point features extracted to 3-D with surface normals. 40

3-10 **Left:** A typical pose-pair feature, described by its spatial position and surface normal attributed to each of the individual features. **Right:** Under rigid motions, the relative difference in position and surface normal orientation between the feature pair should remain the same. 43

3-11 Color-coded edges overlaid on the articulated object indicate the motion profile similarity of trajectory pairs. Red edges indicate high motion profile similarity (rigidly moving, or rigidly stationary), while blue indicate low motion profile similarity (non-rigid motion). The resulting similarity matrix clearly indicates two rigid clusters of feature trajectories - one consisting of 0, 1, 3; the other of 2. 44

3-12 Histogram of observed distances between a pair of trajectories accumulated over one visual lesson. On the ***left*** in *green* is an example of a **rigid** pair of trajectories. We notice that the distribution of observed distances is centered at $\mu = 0.029\ m, \sigma = 0.001\ m$, implying that the relative distance between the two trajectories does not change. On the ***right*** in *blue*, we notice a much larger $\sigma = 0.018\ m$, indicating that the trajectories diverge from each other implying **non-rigid** motion. . 45

3-13 Motion segmentation for a drawer manipulation sequence. The top drawer (Cluster 2) is opened and closed at $t = 10s$. This is followed by the opening and closing of the second drawer (Cluster 1) at $t = 20s$. Color-coded edges overlaid on the articulated object indicate the motion profile similarity of trajectory pairs. 47

3-14 The resulting clustering accurately estimates 3 separate feature trajectory clusters, colored blue, green and red. 48

3-15 Segmentation of rigid-body articulated motions correctly identified by the trajectory clustering algorithm. In both cases, the number of rigid body motions involved during a demonstration, as suggested by the similarity matrices constructed, can be clearly identified. 49

3-16 Visualization of pose estimates obtained after 6-DOF pose optimization using iSAM. 51

3-17 Examples of correctly estimated kinematic structure from 6-DOF pose estimates of feature trajectories. 57

3-18 Figure illustrating the motion manifold of the articulated object, extracted via MSERs. 58

3-19 **Top:** Visualization of re-localization of the refrigerator at a future encounter. **Bottom:** The query instance of the fridge is matched with a previously learned demonstration. 60

3-20 Given the co-registered viewpoints, the kinematic model of the refrigerator is recovered and projected in the query reference frame. The motion manifold predicted lies closely on the expected motion manifold of the object indicating a qualitatively good prediction. 60
3-21 Printer instance re-localization via matched SURF features 61
3-22 Prediction visualization shown in both states suggests a rotational motion of the printer on top (for scanner use) 61
3-23 Figure showing the articulation learning and prediction capabilities of our proposed framework. 62

4-1 Visualizations of the intermediate steps in the evaluation of articulation of a simulated **Prismatic-Rotational** kinematic chain 68
4-2 Visualizations of the intermediate steps in the evaluation of articulation of a simulated **Rotational-Rotational-Rotational** kinematic chain 68
4-3 Comparison of kinematic graph estimated for a **Prismatic-Rotational** articulation model with simulated ground truth 70
4-4 Comparison of kinematic graph estimated for a **Rotational-Rotational-Rotational** articulation model with simulated ground truth 71
4-5 Kinematic structure (**Prismatic** and **Prismatic-Prismatic**) and model parameters estimated by the proposed articulation learning framework from simulated data, compared against known ground truth. 73
4-6 Kinematic structure (**Rotational** and **Prismatic-Rotational**) and model parameters estimated by the proposed articulation learning framework from simulated data, compared against known ground truth. . . 74
4-7 Kinematic structure (**Rotational-Rotational** and **Rotational-Rotational-Rotational**) and model parameters estimated by the proposed articulation learning framework from simulated data, compared against known ground truth. 75

4-8 An example of the in-painted mask (in red) obtained via AprilTag detection, and the corresponding features extracted in the scene using the mask. 77

4-9 Average feature trajectory lengths constructed using KLT, Dense Trajectories, and our proposed feature tracker on the **Drawer**, **Refrigerator**, and **Chair** data sets. 79

4-10 Total number of feature trajectories constructed using KLT, Dense Trajectories, and our proposed feature tracker on the **Drawer**, **Refrigerator**, and **Chair** data sets. 80

4-11 Figure illustrating the evaluation of pose estimation accuracy of our proposed framework with traditional marker-based tracking solutions. 81

4-12 **Demonstration 1**: Comparison of $SE(3)$ pose estimates, while operating a refrigerator, determined via fiducial markers (Tag) and our framework (Ours). The gray shaded region indicates the maximum observed deviation of $SE(3)$ pose obtained via our framework from that observed via fiducial markers. 82

4-13 **Demonstration 2**: Comparison of $SE(3)$ pose estimates, while manipulating a drawer, determined via fiducial markers (Tag) and our framework (Ours). The gray shaded region indicates the maximum observed deviation of $SE(3)$ pose obtained via our framework from that observed via fiducial markers. 83

4-14 **Demonstration 3**: Comparison of $SE(3)$ pose estimates, while manipulating a chair, determined via fiducial markers (Tag) and our framework (Ours). The gray shaded region indicates the maximum observed deviation of $SE(3)$ pose obtained via our framework from that observed via fiducial markers. 84

4-15 Comparison of spatial, and rotational accuracy of the $SE(3)$ pose **tracked** via AprilTags (Abs-Tag) against the **predicted** pose of the tracked object (Abs-Est). The gray shaded region indicates the maximum observed deviation of $SE(3)$ pose predicted via our framework from that observed via fiducial markers. 87

List of Tables

3.1 A summary of parameters used in the feature tracking implementation. 38

3.2 A summary of parameters used in the motion segmentation implementation. 46

3.3 Candidate Models used for articulation modeling 55

4.1 Evaluation of position and orientation error associated with learning the underlying articulation in a simulated object, as compared to ground truth 69

4.2 Comparison of accuracies in kinematic model estimation determined using pose observations from AprilTags tracking against those estimated using the proposed framework. 85

4.3 Kinematic model parameters determined using either methods show comparable results. With minimal supervision, our proposed framework can accurately identify the correct number of object parts, and estimate the true object pose to provide sufficiently accurate observations for articulation learning. 86

Chapter 1

Introduction

As robots have become increasingly able to operate in the midst of uncertainty, there is growing demand for robots that can assist people in our home as well as health care, manufacturing, and logistics settings. A long-standing challenge is to develop machines that can interact effectively with the diverse, complex objects built by humans for their own use. Existing approaches to manipulation often assume either a limited set of objects for which the interactions can be pre-defined or, provide a means of learning available interactions for a small number of simple objects, typically after extensive training. These methods are not well-suited to the articulated objects commonly found in man-made environments (e.g., drawers, doors, refrigerators, chairs, etc.), whose models can be complex.

Articulation learning refers to the understanding of the underlying kinematic model that describes an object's motion. In order to learn opportunistically from unstructured environments, robots need the ability to simultaneously track and learn the objects in the scene robustly. Existing methods require the robot to operate in structured environments with prior models of the objects being manipulated. Furthermore, fiducial tags are often used to provide rich noise-free data as observations to the articulation modeling pipeline.

In this work, we develop a framework that enables a robot to learn from demonstrations provided by a human teacher, using RGB-D data as its only input. The robot is capable of learning from multiple visual lessons, and can reason over the

underlying kinematic structure of the articulated object being manipulated. We argue that manipulation tasks can be further enabled with specific knowledge of the underlying kinematic model of the object being manipulated. Additionally, learning a model from multiple visual user-provided demonstrations provides sufficient knowledge to the robot to plan for future manipulation tasks involving the same instance of the object. Thus, we focus on the particular problem of observing demonstrations of tasks in daily household environments involving manipulation of articulated objects.

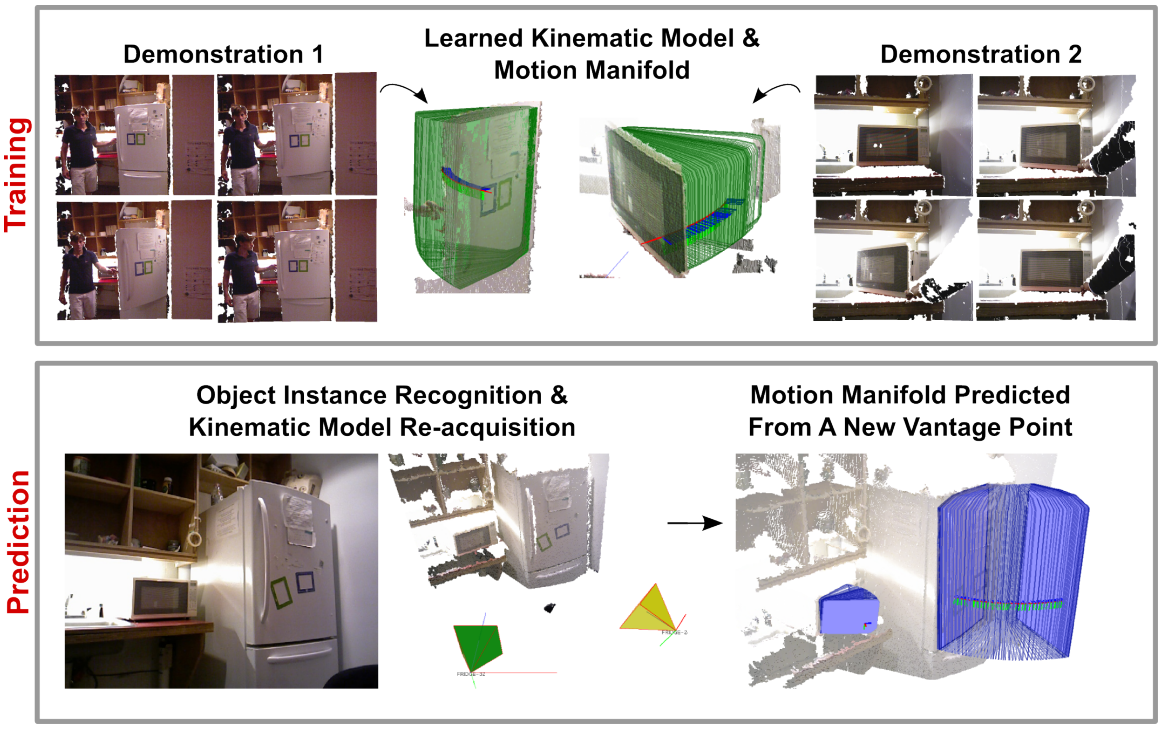


Figure 1-1: The proposed framework reliably learns the underlying kinematic model of articulated objects from user-provided visual demonstrations, and subsequently predicts their motions at future encounters, from novel vantage points.

We combine techniques from sparse feature tracking, motion segmentation, object pose estimation and articulation learning to learn the different types of manipulation involved in a demonstration, and the underlying kinematic relations of the articulated object being manipulated. We acknowledge the importance of uncertainty when estimating such models, and hence take a probabilistic approach that incorporate sensor uncertainties in a sound manner. Finally, our method enables the robot to predict the motion of articulated objects it has learned previously, providing valuable object

manipulation information to the robot in future encounters. Figure 1-1 illustrates a scenario where our framework learns the kinematic model of a refrigerator, and microwave, in separate user-provided demonstrations. Subsequently, at a future encounter, it can predict the motion of each learned articulated object.

1.1 Thesis Overview

This section summarizes the components used to enable robots to learn from user-provided visual demonstration.

1.1.1 Learning from Visual Demonstration

Spatio-Temporal Feature Tracker: The first step in learning from visual demonstration involves visually observing and tracking object features during manipulation tasks. We focus on unstructured environments where rich fiducial markers are not available. We develop a feature-based tracking system that constructs feature trajectories of object parts from their motion during manipulation. Our algorithm constructs long trajectories of object motion, while reducing feature drift.

Motion Segmentation: The feature trajectories constructed from the object motion exhibit significant information about the segmentation of the object into its parts. We develop a novel "pose-pair" representation that provides sufficient statistics on the segmentation of trajectories observed by the robot. We show that our system is capable of correctly identifying the number of objects parts involved in the manipulation, in an unsupervised manner.

Pose estimation: Given the segmentation of feature trajectories, we reconstruct the $SE(3)$ pose of the articulated object during its manipulation. Here, we employ a pose graph optimizer to successfully recover and refine the $SE(3)$ pose estimates of each object part. Additionally, we incorporate techniques to deal with outliers and high sensor uncertainty, providing added robustness.

Articulation Learning: In order to learn the underlying kinematic structure of the articulated object observed, we utilize an existing tool as described in Sturm

et al. [28]. Given the estimated $SE(3)$ pose estimates of each object part during its manipulation, our system successfully estimates the kinematic structure of several household objects.

Articulation Prediction: Motivating the need for robots that are capable of life-long learning, we enable robots to learn and reason from user-provided manipulation lessons of articulated objects. Here, we introduce capabilities to allow a robot to predict the motion of articulated objects it has learned from experience, providing valuable information on object manipulation in future encounters.

1.1.2 Implementation

In this section, we elaborate on the tools and techniques that were employed in the design and implementation of our articulation learning from visual demonstration framework.

1.1.3 Experiments and Analysis

Section 4 evaluates the performance of the proposed articulation learning from visual demonstration framework. We analyze each component of the proposed framework, including feature tracking, motion segmentation, pose estimation, articulation learning and articulation prediction. We also evaluate the articulation framework by validating it with simulated and real-world ground truth data.

1.2 Contributions

Our contributions include a visual inference framework that enables a robot to learn from user demonstrations of manipulation tasks in unstructured environments. Unlike existing frameworks that employ fiducial tags to assist object detection and tracking, our system works in unstructured environments without the need for visual markers. Combining techniques from sparse feature tracking, motion segmentation, pose estimation, and articulation learning, our framework can learn from

user-provided demonstrations of manipulation tasks, and reason over the underlying kinematic model of the articulated object being manipulated. Additionally, our framework predicts the motion of objects in future encounters, providing the robot with valuable object manipulation information for planning purposes. We envision such learning frameworks as a step toward more general-purpose robots that can learn from humans in an opportunistic manner.

Chapter 2

Related Work

Articulation learning refers to the understanding of the underlying kinematic model that describes an articulated object's motion. For a robot to learn opportunistically from unstructured environments, it must simultaneously track, and learn the articulation of, objects in the scene. This requires the robot to have capabilities such as object tracking, motion segmentation, pose estimation, and articulation learning to recover the kinematic model of the object involved during the demonstration. Our framework ties several of the aforementioned concepts into a unified articulation learning pipeline that allows robots to learn the underlying kinematic structure of an articulated object given multiple visual lessons, and use the learned structure to predict object motion during future encounters.

2.1 Background

While there has been significant progress in object tracking, motion segmentation, pose estimation, and articulation learning, little effort has been made to develop a complete framework that combines these capabilities to enable robots that can learn from user-provided visual lessons. That said, there have been recent developments in this domain, especially by Katz et al. [19]. Earlier attempts by Katz et al. [17], [16], [18] focus on the ability to extract relevant segmentation and kinematic models from interactive manipulation of an articulated object. However, they handled motions

that were primarily planar in nature, and required careful consideration of sensor capabilities. Their implementation requires sufficient texture in the scene in order to perform feature tracking reliably. The authors also note the need for tuning several parameters, making the system sensitive to changes in the parameters and thereby reducing the reproducibility of the overall architecture. Recently in Katz et al. [19], the author shows an improved variant of their interactive manipulation-based articulation learning algorithm that has equally good performance with reduced algorithmic complexity.

Existing algorithms that learn the articulation in objects from visual demonstration make strong assumptions about the scene in which the demonstration is conducted. Most existing implementations require the robot to operate in structured environments with prior models of the objects being manipulated. Visual markers or fiducial tags have been used extensively to avoid the need for additional machinery for visual perception. In their work, Sturm et al. [28] argue that fiducial tags can provide rich reduced-noise data as observations to the articulation modeling pipeline. They also make assumptions on the availability of marker pose observations at every time step, which will likely fail to hold in an unstructured setting. Furthermore, they also assume that the number of unique object parts is known a priori. We expect robots to work alongside humans in a natural manner such that they can learn persistently. This necessitates the ability to deal with unstructured environments which humans constantly modify.

As previously mentioned, our framework builds on top on different aspects of object understanding, including tracking, segmentation, pose estimation, and articulation learning. In the following sections, we provide a short background of each of these areas of active research and how they influence our overall articulation learning pipeline.

Feature tracking

Feature tracking is an essential component required in order to trace the trajectory of an object being manipulated. Traditional feature tracking algorithms make

strong assumptions about the texture of the object being tracked. Trackers such as KLT [3], or feature trajectories based on SIFT [20], strongly depend on objects being heavily and uniquely textured. Variants of these algorithms enable learning the object features in an on-line fashion. However, such capabilities give rise to drift in tracking of these features, that eventually lead to significant errors in the overall object trajectory observed. KLT is also limited in its capabilities as it does not handle object occlusions nor does it provide any feature reacquisition. SIFT-based trajectories on the other hand are capable of handling occlusions, however, they are computationally expensive. Dense trajectories [34] employ a different strategy involving dense sampling, followed by a dense optical flow for their feature extraction and propagation steps respectively. Here, the features are sampled via the Shi and Tomasi criterion as noted in [26]. For the feature propagation step, the authors employ a pyramidal implementation of dense optical flow by Farnebäck [11]. To avoid drift in the feature trajectories extracted, the trajectories are pruned after they have achieved a fixed length of 15, after which they are eliminated and new trajectories are sampled. The authors trade accuracy for speed, allowing it to run at approximately 8Hz. On the other hand, large-displacement optical flow methods [5] or particle video [25] tend to be more accurate for the construction of long range trajectories, however, they take a substantially longer time to process.

We draw concepts from existing work to introduce a robust multi-scale feature tracker that can learn new feature representations over time, but also ensures little drift by incorporating a detection-learning-update cycle.

Motion Segmentation

Motion segmentation refers to the partitioning of a sequence of observations into multiple spatio-temporal regions with similar motion profiles. Existing algorithms in motion segmentation use feature based trackers to construct spatio-temporal trajectories from sensor data, and cluster these trajectories based on rigid-body motion constraints. Recent work by Brox and Malik [4] in segmenting feature trajectories has shown promise in analyzing and labeling motion profiles of objects in video sequences

in an unsupervised manner. The authors of [13], [31], and [33] have consistently introduced newer algorithms that segment regions in video sequences based on rigid body motion. Elhamifar and Vidal [9] have shown effective methods in labeling object points purely based on motion observed in a video sequence taken by a standard camera. Katz et al. [19] also use similar motion trajectory similarities as suggested in [4] to cluster feature trajectories in 3-D.

We draw inspiration from these works and transform our motion segmentation problem into a kernel-based clustering problem by analyzing the relative motion profiles of feature trajectories.

Pose Estimation

Traditional object tracking algorithms work mostly on the level of single rigid objects. These methods extract salient features from the object and subsequently match them across frames to estimate the relative object motion in $SE(3)$. Techniques often apply RANSAC [12] to features in successive frames to remove outliers before estimating the relative object pose in $SE(3)$. More recent techniques [8] simultaneously learn new features on the object while tracking and estimating its orientation by posing it as a structure-from-motion problem. However, these algorithms focus only on pose estimation for a single object in the scene. Since we are particularly interested in the motion of articulated objects that consist of multiple rigid-object parts, we perform pose estimation and optimization for each object segment independently.

Articulation Learning

Articulation learning refers to understanding of the underlying kinematic model of an articulated object, given a sequence of observations exhibiting the motion of its object parts. In our setting, we want to enable robots that can learn the underlying kinematic linkages involved in an articulated object purely by observing user-provided visual demonstrations. Initial attempts by Yan and Pollefeys [35] employ structure from motion techniques to learn the segmentation of the object parts, and to estimate rotational degrees of freedom between the object parts. However, these techniques are

computationally expensive, and do not model their motion estimates in a probabilistic manner. Katz et al. [19] also provide a framework to segment and infer the kinematic linkages involved in an articulated object by constructing and observing feature trajectories in a video sequence. However, the authors do not explicitly incorporate the kinematic model complexity leading to over fitting of the kinematic models. Sturm et al. [28] make a valuable contribution to this end, by introducing a probabilistic framework that reasons over the likelihood of the observations while also considering its trade-off with kinematic model complexity. However, the authors utilize fiducial markers to provide rich reduced-noise data as observations to their proposed articulation modeling pipeline. They also assume that the number of unique object parts is known a priori.

Chapter 3

Articulation Learning from Visual Demonstration

In this section, we introduce the algorithms used in our framework that enable a robot to learn the articulation in objects from user-provided visual demonstrations. Figure 3-1 illustrates the steps involved in the articulation learning from visual demonstration pipeline discussed in this work.

Our proposed approach consists of a training phase and a prediction phase. The training phase is broken down as follows: (i) Given RGB-D data, the spatio-temporal feature tracker constructs long-range feature trajectories in 3-D. (ii) Using a unique relative motion-profile similarity, clusters of rigidly moving feature trajectories are determined. (iii) The 6-DOF motion of each cluster is then estimated using a 3-D pose optimizer. (iv) The most likely kinematic structure, and model parameters of the articulated object is determined, given SE(3) pose estimates of all the identified clusters. Figure 3-2 illustrates the steps involved in the training phase with appropriate input and outputs for each of the components.

Once the kinematic model of an articulated object is learned, our system can predict the motion trajectory of the object at future encounters. In the prediction phase: (i) Given RGB-D data, the descriptions of the objects in the scene, $\mathbf{D}_{query}$, are extracted using SURF [2] descriptors. (ii) All the objects and their corresponding kinematic models, $\hat{G}, \hat{M}_{ij}, (ij) \in \hat{G}$, that match $\mathbf{D}_{query}$ are re-acquired (iii) Given

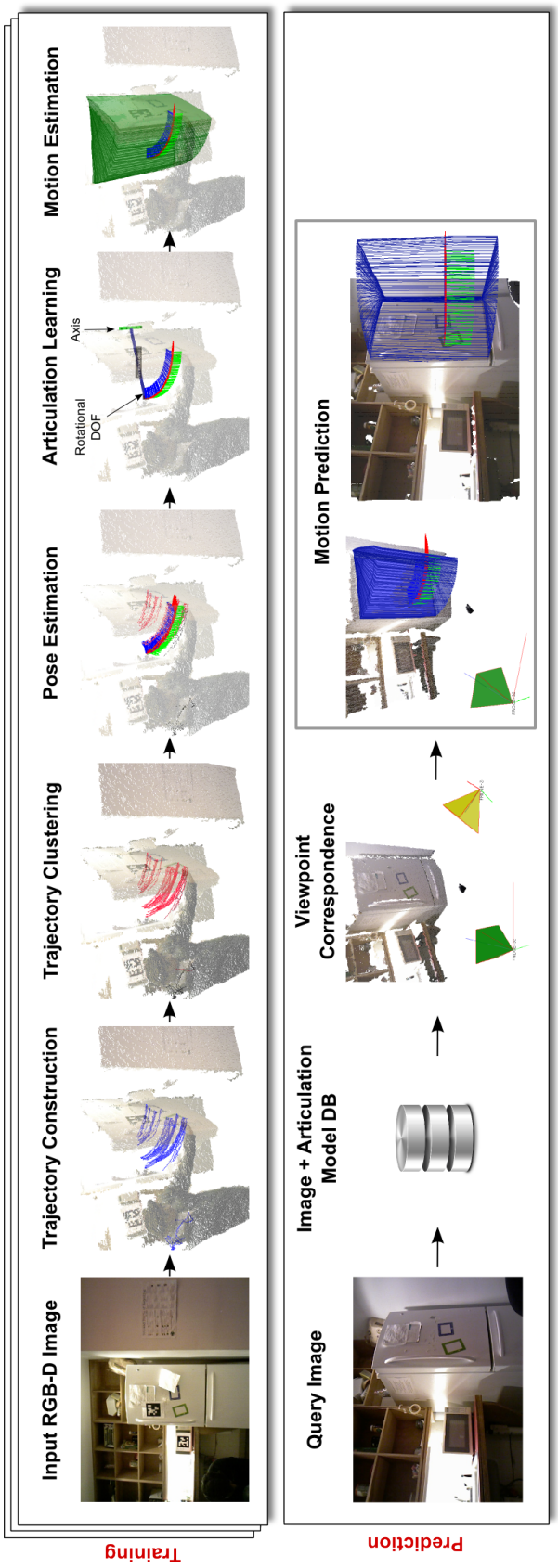


Figure 3-1: The architecture of our system for articulation learning from visual demonstration.

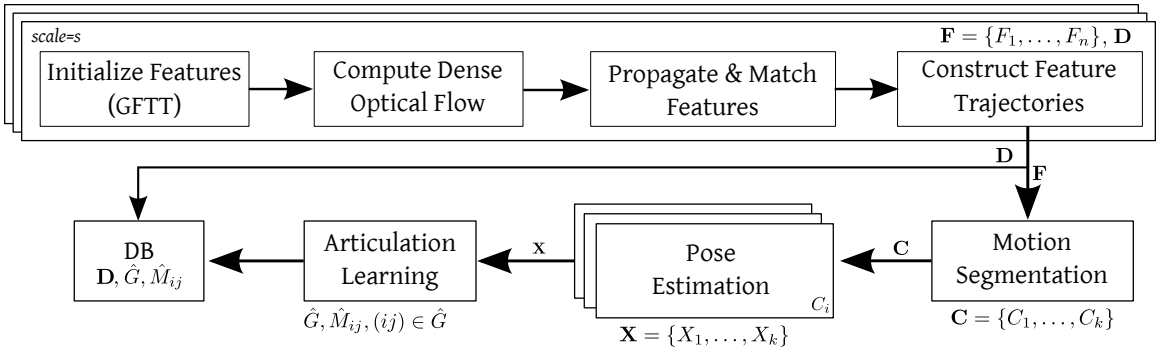


Figure 3-2: The training phase.

the correspondences, and kinematic model parameters of the articulated object, the articulated motion of the object is predicted. Figure 3-3 illustrates the steps involved in the prediction phase.

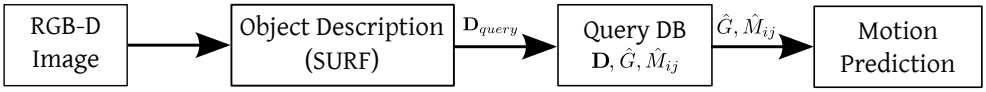


Figure 3-3: The prediction phase.

3.1 Spatio-Temporal Feature Tracking

In order to understand the interaction between the user and the articulated object being manipulated, it is crucial to identify and trace the 3-D path that the object and its parts take while being manipulated. High-fidelity trajectory observations enable inference over the underlying kinematic model of the object. We explore the use of image-based interest-point features and extend them to 3-D to build trajectories of the object parts from observations of the parts in motion.

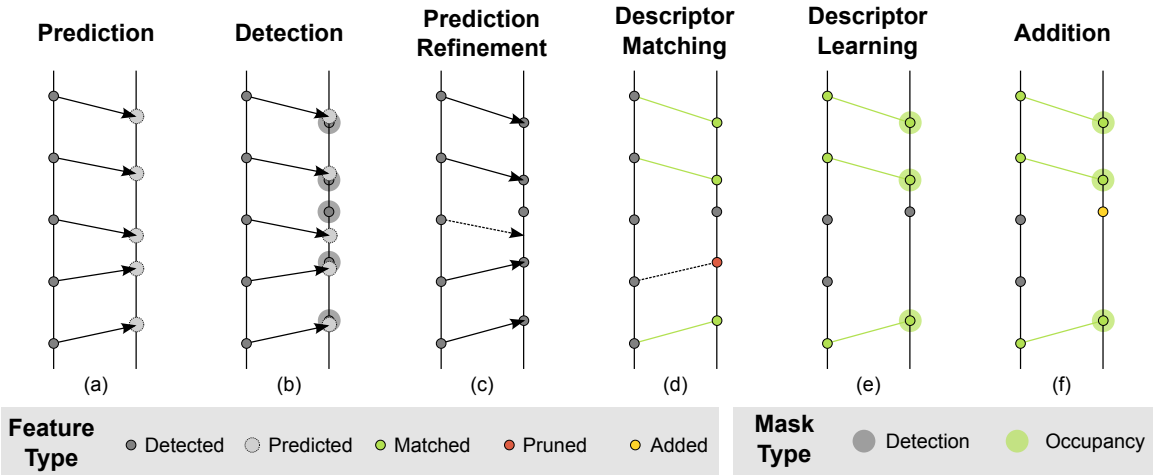


Figure 3-4: Spatio-Temporal Feature Tracking steps combine traditional dense optical flow methods with feature detection and matching techniques to construct long range trajectories with little to no drift.

3.1.1 Trajectories via Interest-Point Tracking and Learning

Since the manipulated object may be viewed from various aspects, and may be occluded, the tracking pipeline must be able to adapt to these variations. To this end, we employ feature-based techniques to track and detect interest points on the articulated object. Our framework is modular in the sense that it accommodates a variety of algorithms for feature detection, extraction, or prediction. A literature review in feature tracking or interest point detectors [32] will lead one to notice several variants of feature detectors such as Good Features To Track, FAST features, and Harris Corners. They also include scale-invariant feature detection algorithms such as the ones described in SIFT [20], and SURF [2]. The same applies in the case of feature descriptors where SIFT, SURF, ORB [24], BRIEF [7], and FREAK[1] are interchangeably used to accommodate accuracy-versus-speed trade offs.

We develop an algorithm that combines interest-point detectors and feature descriptors with traditional optical flow methods. Additionally, with a tracking-learning-update cycle, the feature trajectories constructed tend to be longer in length, compared to existing feature tracking methods, and sufficiently robust to drift. We refer the reader to Section 4.2.2 for further justification of this claim. In the sections below, we explain the individual components of our feature tracking algorithm. For a

brief overview of the feature tracking algorithm, please refer to Alg. 1 (p. 41) and Figure 3-4. Figure 3-5 illustrates the qualitative results we achieve with the proposed feature tracking pipeline.

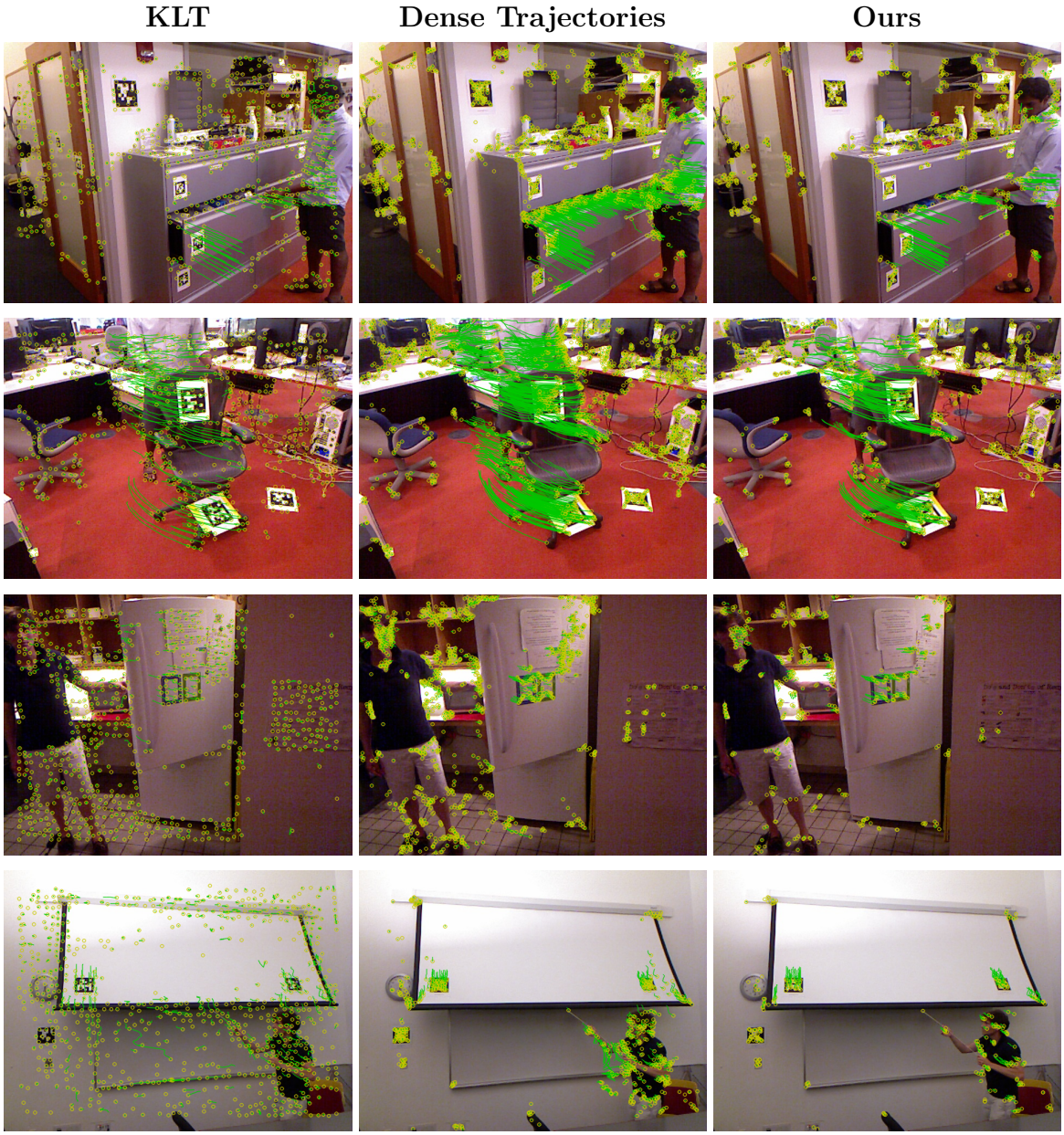


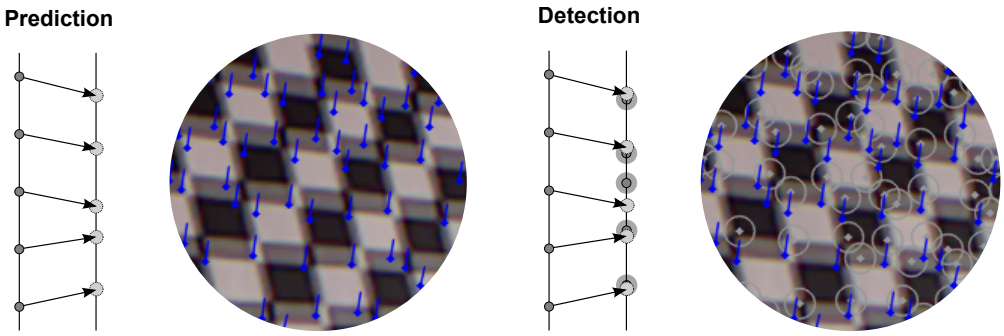
Figure 3-5: Trajectories constructed using KLT [29] (**left**), Dense Trajectories [34] (**middle**), and our algorithm (**right**). While all 3 algorithms show reasonably good trajectory construction, our algorithm exhibits negligible drift while achieving longer trajectories.

Feature Detector

Feature trajectories require an initialization step, where salient scene features are extracted based on certain criterion threshold. We employ Good Features To Track (GFTT) [26] to initialize up to 1500 salient features whose quality level is greater that 0.04. Due to the unstructured nature of indoor scenes, the feature detector is evaluated at 5 different scales, each down-sampled by a factor of $\sqrt{2}$. This provides additional scale-space robustness to the feature detector. This allows us to fix the feature block size parameter, a feature detection parameter, to 7 pixels. Once the features are detected, we populate a mask image that keeps track of regions where interest points are detected at each of the pyramid scales. The detection step is visualized in Figure 3-6(b).

Feature Prediction

Once the features are initialized, the they must be propagated in a manner that conforms to the motion of the object locally at that point. We draw techniques from previously established work on dense optical flow by Farnebäck [11] to provide predictions for the initialized features at the next time step. Our specific implementation also adds a median filtering step as suggested in [34] that reduces false positives in the typical dense optical flow implementation. In order to improve on the robustness of the optical flow estimation, we employ scale-space methods to predict the local flow of features at each scale. The combination of these implementation details provide a model sufficiently accurate for the initialized features for their prediction in future frames. The prediction step is visualized in Figure 3-6(a).

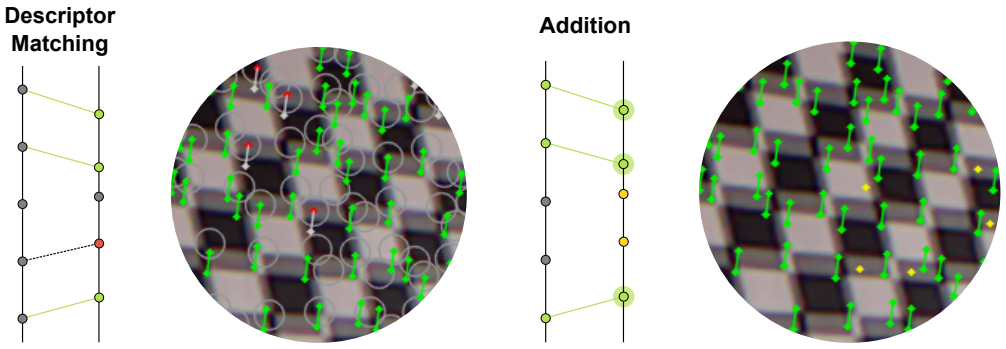


(a) Feature Prediction (b) Feature Detection

Figure 3-6: Illustration of feature detection and prediction for a checkerboard pattern undergoing motion. (a) Feature detections (via GFTT) and feature predictions (via Dense Optical Flow) are indicated by blue points and blue edges respectively. (b) Gray points indicate the feature detections in the subsequent frame. Gray circles indicate the regions where feature predictions continue to be valid.

Bootstrapped Learning and Detection

Optical flow methods typically exhibit drift over long time periods, and require correction steps to prevent this drift. We bootstrap the detection and tracking steps with a feature description step that extracts and learns the description of the feature trajectory. The expectation that feature trajectories should arise from a single point on the object amounts to strong assumption that the underlying description of the feature itself should remain unchanged. We use this assumption to bootstrap feature description at every subsequent frame to avoid long-range drift. At each image scale, we compute the SURF descriptor [2] over feature patches that were predicted from the previous step, and compare them with the description of the detected feature that is within a threshold distance of the predicted feature. Subsequently, features that successfully meet a desired match score are added to the feature trajectory, while the rest are pruned. The feature matching step is visualized in Figure 3-7(a).



(a) Feature Matching (b) Feature Addition

Figure 3-7: Illustration of feature matching and addition. (a) Features are successfully matched by comparing the SURF descriptions of putative matches. Successful matches are colored in green, while those not matched are grayed out (b) Successfully matched features are added to the feature tracks (in green). Subsequently, new features are added in regions that lack matches (in yellow).

Trajectory Correction

In addition to using feature descriptors to avoid drift in the trajectories, we correct the feature predictions with an additional feature detection step. Vanilla implementations of the feature detection-learning-update concept are prone to drift in earlier stages of the feature trajectory. We employ the previously computed detection mask as a guide to reinforce feature predictions with feature detections, and subsequently construct valid feature trajectories with negligible drift. This ensures that features continue to lie on salient regions in the scene and reduces erroneous optical flow vectors determined. This step also provides robustness to sensor noise that tend to generate spurious features in the image. Figure 3-8 shows the superior performance of our algorithm in tackling drift issues.

Name	Value	Description
`num_feats`	1500	Maximum number of features
`quality_level`	0.04	Minimum feature quality level
`block_size`	7	Feature block size
`num_scales`	5	Number of pyramid levels
`scale_factor`	$\sqrt{2}$	Pyramid down-sampling factor
`min_dist`	5 px	Minimum distance between prediction and detection

Table 3.1: A summary of parameters used in the feature tracking implementation.

Trajectory Management

For typical video sequences, some features are continuously tracked, while other features are lost due to occlusion or lack of salience in the source image. To enable rich trajectory information, we continuously add features to the scene, when necessary, using the occupancy mask described earlier. We maintain a constant number of feature trajectories tracked, by adding newly detected features in regions that are not yet occupied as suggested by the occupancy mask. This step is shown in Figure 3-7(b). As features are continually added for tracking, we incorporate track management capabilities that also provide relevant statistics about the feature trajectories and their validity. Some of the capabilities include statistics on feature descriptions for a particular trajectory, unique trajectory identification, sub-pixel refinement, and depth uncertainty estimation from the RGB-D image.

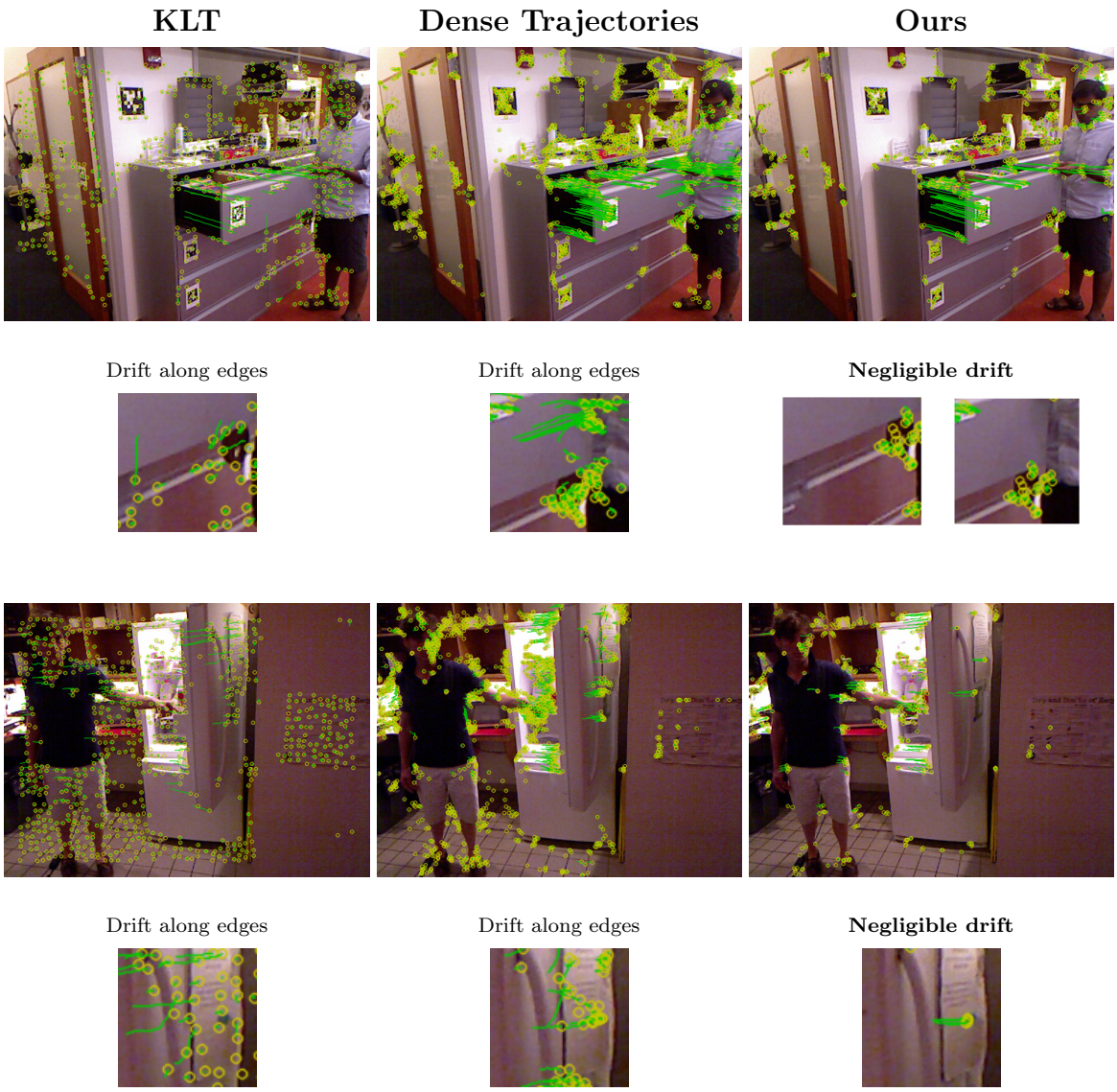


Figure 3-8: The proposed feature tracking algorithm shows negligible drift, compared to KLT (**left**), and Dense Trajectories (**middle**), while achieving longer trajectories.

Extensions to 3-D

By taking advantage of the known image-to-depth correspondence in RGB-D sensors, most existing image-based feature extraction algorithms can be easily extended to 3-D. Additionally, with the dense organized point cloud representation that RGB-D sensors provide, object surfaces and surface normals are easily extracted with the use of integral depth images [14]. Taking advantage of these techniques, the 3-D

position and surface normal of the tracked features can be deduced from existing feature key-points. As a result, each feature key-point is represented by its normalized image coordinates (u, v), position $\vec{p} \in \mathbb{R}^3$ and surface normal $\vec{n}$, represented as $(\vec{p}, \vec{n}) \in \mathbb{R}^3 \times SO(2)$. We compute the surface normals by averaging the gradients over a patch size that is a function of the depth at which the points are registered. The resulting feature trajectory is denoted as $f^t = (\vec{p}, \vec{n})^t \in \mathbb{R}^3 \times SO(2), \forall t \in T$ representing the trajectory taken by an object feature. Figure 3-9 illustrates the extension of image-based features to 3-D.

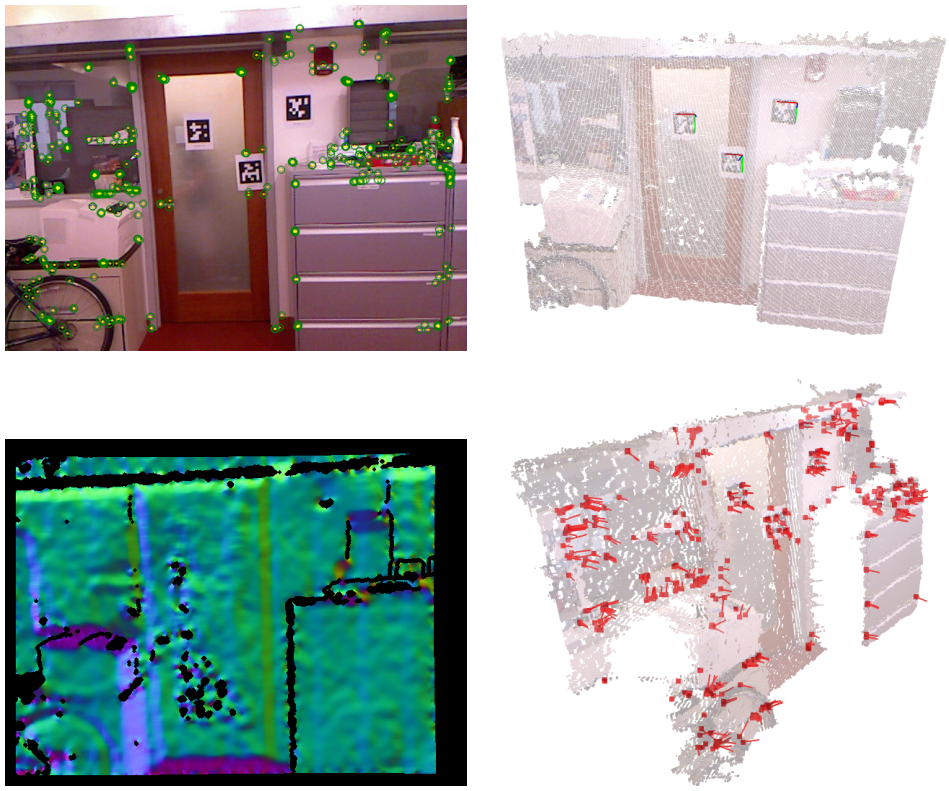


Figure 3-9: Extensions to 3-D: Feature Detections are extended to 3-D with the readily available point cloud representation, and the surface normal is estimated via integral depth images. **Top Left:** A typical RGB Image, **Top Right:** Point cloud with co-registered RGB image. **Bottom Left:** Surface normals computed via integral depth images. **Bottom Right:** The resulting interest-point features extracted to 3-D with surface normals.

Algorithm 1 Drift-free Bootstrapped Feature Tracker

1: **Inputs:**
Time-indexed RGB-D data: $\mathcal{I}^{1,\ldots,t}$
Parameters: `min_distance`, `num_scales` from Table 3.1
2: **Notation:**
$\mathcal{I}^t_{DetMask}$: Mask identifying regions where features are detected
$\mathcal{I}^t_{OccMask}$: Mask identifying regions where features are successfully predicted
3: **Initialize:**
$\mathbf{F} = \{F_1, \ldots, F_n\}$: Sets of uniquely identified feature trajectories
$F_i = \{f_i^1, \ \ldots, f_i^t\}$: Trajectory feature locations at time t
4: **Outputs:**
Sets of uniquely identified feature trajectories
$\mathbf{F} = \{F_1, \ldots, F_n\}$, where $F_i = \{(f_i^1), \ldots, (f_i^t)\}$: Set of uniquely identified trajectories
$\mathbf{D} = \{D_1, \ldots, D_n\}$, where $D_i = \{(d_i^1), \ldots, (d_i^t)\}$: Corresponding SURF descriptions for each of the identified trajectories
5: **Procedure:**

- Build image pyramids $\mathcal{I}^t_s$ for each scale $s \in \{1, \ldots, \texttt{num\_scales}\}$
- For each pyramid scale s, detect sparse features f^t using Good Features To Track feature detector
- Add detected features f^t at each scale to the detection mask image $\mathcal{I}^t_{DetMask}$ with a feature radius mask of `min_distance`
- Compute the Farneback polynomial expansion term $\mathcal{P}^t_s$ from the image pyramids $\mathcal{I}^t_s$
- Compute Farneback Dense Optical Flow $\Delta\, \mathcal{I}^t_s$ using the previous and current polynomial expansion terms $(\mathcal{P}^{t-1}_s, \mathcal{P}^t_s)$
- Predict features in the current frame $\hat{f}^t$ using features from the previous time step f^{t-1}, and the flow vector $\Delta\mathcal{I}^t_s$
- Prune predicted feature tracks $\hat{F}^t$ that do not fall within the detection mask $\mathcal{I}^t_{DetMask}$ to ensure that the predicted features also exist in the current frame
- Extract SURF descriptions d^t for each of the pruned features $\hat{f}^t$, and match against descriptors from the previous time step d^{t-1}. Prune feature tracks $\hat{F}^t$ that do not meet the match threshold, $d_{match} \leq 2 * min(d^{1,\ldots,t-1})$). Add the successfully matched features to the occupancy mask image $\mathcal{I}^t_{OccMask}$ with a feature radius mask of `min_distance` to ensure that newly added features are not too close to existing features tracked.
- Add detected features f^t to the feature tracks that do not fall in regions set in the occupancy mask $\mathcal{I}^t_{OccMask}$. Extract SURF descriptors d^t for each of the newly added points in the feature track set.

3.2 Motion Segmentation

To identify the underlying kinematic relationships within object parts in an articulated object, it is important to correctly identify its different object parts by analyzing and distinguishing the trajectories that these object parts take. We approach the problem from a manifold learning perspective, where the higher-dimensional motion manifold of the feature trajectories is assumed to lie in a lower-dimensional manifold. With this representation, we can analyze an entire manipulation sequence and parametrize the high-dimensional motion of the object and its parts to a lower-dimensional embedding while maintaining low reconstruction error. In particular, we analyze the relative pose transformations of the object parts with respect to each other over time, and infer whether the object parts are rigidly associated or not. To reason over candidate segmentation, we formulate a clustering problem to identify the different motion subspaces in which the object parts lie. See Alg. 2 (p. 50) for pseudo-code of our motion segmentation algorithm.

3.2.1 Trajectory Matching

As explained earlier, we exploit the relative motion profiles of feature trajectories to infer whether they are rigidly connected. This implies that upon clustering, similarly assigned labels are rigidly connected, while dissimilar labels are evidence of non-rigid relative motion. The feature trajectories computed as described in the previous section are used directly as an input to this step to evaluate sets of trajectories that may be rigidly connected to one another, indicating that they lie on the same rigid object part.

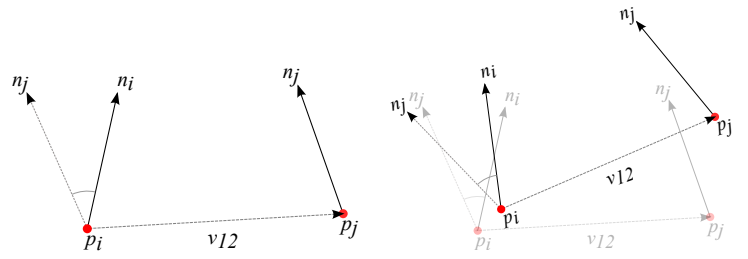


Figure 3-10: **Left:** A typical pose-pair feature, described by its spatial position and surface normal attributed to each of the individual features. **Right:** Under rigid motions, the relative difference in position and surface normal orientation between the feature pair should remain the same.

For features that belong to the same rigid part, their relative displacements must be consistent over the entire span of the feature trajectory. The same constraint applies in the space of surface normals as well, where the relative angular difference between pairs of surface normals should remain consistent if the features belong to the same rigid body. Figure 3-10 shows the rigid motion of a typical pair of features considered with position and surface normals. As expected, this constraint applies only up to some sensor noise that we account for in our observation model. Modeling the noise as zero-mean Gaussian, the distribution over the relative change in displacement vectors and surface normal vectors becomes $\mathcal{N}(\mu, \Sigma) = (0, \Sigma)$, where Σ is the noise covariance that we determine empirically for rigidly connected feature pairs. Subsequently, we use this model to define the similarity in Eqn. (3.1) that two feature trajectories belong to the same rigid-body object. In particular, we define the average likelihood that the two features belong to the same rigid object as:

$$L(i,j) = \frac{1}{T} \sum_{t \in t_i \cap t_j} \exp \left\{ -\gamma \left(d(x_i^t, x_j^t) - \mu_{d_{ij}} \right)^2 \right\} \tag{3.1}$$

where t_i and t_j are the observed time instances of the feature trajectories i, and j respectively, and $T = |t_i \cap t_j|$. γ denotes the bandwidth parameter that explains the variation in relative motions of the two trajectories. In the context of spatial motion, $d(\vec{p_i}^t, \vec{p_j}^t)$ is the L_2-distance between a pair of points, $\vec{p_i}^t$ and $\vec{p_j}^t$, in a trajectory. In the case of surface normals, $d(\vec{n_i}^t, \vec{n_j}^t)$ is 1 minus the dot product of a pair of surface

normals, $\vec{n_i}^t$ and $\vec{n_j}^t$. For a pair of 3-D key-point features $\vec{p_i}$, and $\vec{p_j}$, we estimate the mean relative displacement between a pair of points moving rigidly together as

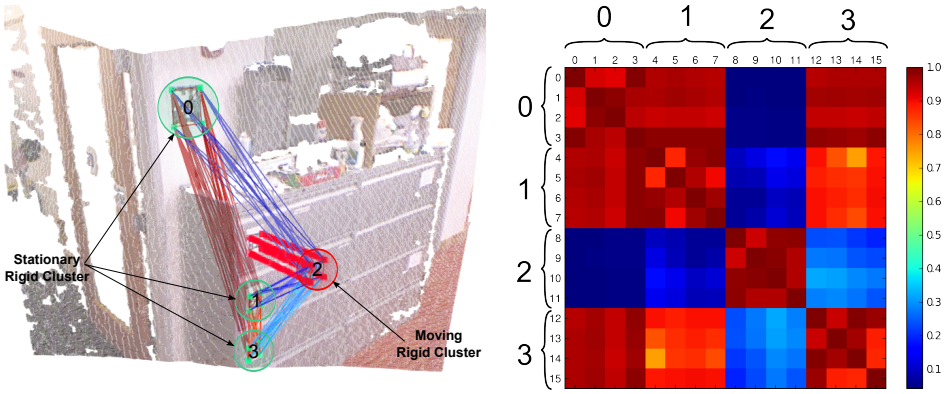
$$\mu_{d_{ij}} = \frac{1}{T} \sum_{t \in t_i \cap t_j} d(\vec{p_i}^t, \vec{p_j}^t) \tag{3.2}$$

where $d(\vec{p_i}, \vec{p_j}) = \|\vec{p_i} - \vec{p_j}\|$. For 3-D key-points, we use $\gamma_p = \frac{1}{0.02}$m in Eqn. 3.1.

For a pair of surface normals $\vec{n_i}$ and $\vec{n_j}$, we define the mean distance as

$$\mu_{d_{ij}} = \frac{1}{T} \sum_{t \in t_i \cap t_j} d(\vec{n_i}^t, \vec{n_j}^t), \tag{3.3}$$

where $d(\vec{n_i}, \vec{n_j}) = 1 - \vec{n_i} \cdot \vec{n_j}$. In this case, we use $\gamma_n = \frac{1}{\cos(15\,^\circ)}$ in Eqn. 3.1. Figure 3-11 illustrates the construction of a motion similarity matrix with the use of fiducial markers.



(a) Rigid-Body Trajectory clusters (b) Trajectory Similarity Matrix

Figure 3-11: Color-coded edges overlaid on the articulated object indicate the motion profile similarity of trajectory pairs. Red edges indicate high motion profile similarity (rigidly moving, or rigidly stationary), while blue indicate low motion profile similarity (non-rigid motion). The resulting similarity matrix clearly indicates two rigid clusters of feature trajectories - one consisting of 0, 1, 3; the other of 2.

Figure 3-12 shows two separate cases of the pose-pair distribution constructed from the relative motion profiles of a pair of trajectories. In both cases, the variance in relative pose is significantly higher for non-rigid motions, providing a sufficient statistic to distinguish rigid-motions from non-rigid ones.

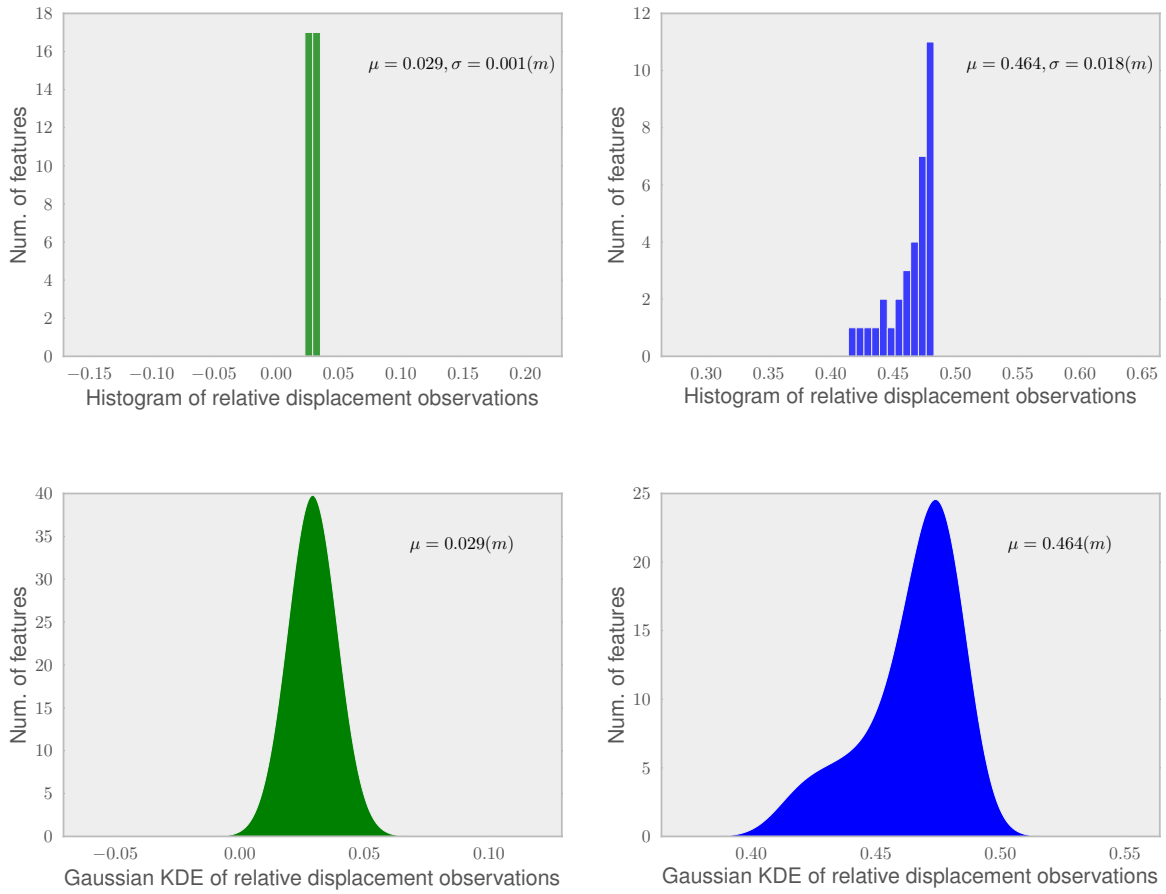


Figure 3-12: Histogram of observed distances between a pair of trajectories accumulated over one visual lesson. On the ***left*** in *green* is an example of a **rigid** pair of trajectories. We notice that the distribution of observed distances is centered at $\mu = 0.029\ m, \sigma = 0.001\ m$, implying that the relative distance between the two trajectories does not change. On the ***right*** in *blue*, we notice a much larger $\sigma = 0.018\ m$, indicating that the trajectories diverge from each other implying **non-rigid** motion.

3.2.2 Trajectory Clustering

One can also view the above likelihood function in Eqn. 3.1 as constructed via a Radial Basis Function kernel

$$K(x, x') = e^{(-\gamma\ \|x - x'\|)} \tag{3.4}$$

where x and x' denote the individual trajectories, and γ denotes the kernel bandwidth parameter described earlier. Since the bandwidth parameter γ for a pair of feature trajectories can be predicted from the expected variance in relative motions

of trajectories, we employ DBSCAN [10], a density-based clustering algorithm, to find rigidly associated feature trajectories. We expect the average likelihood determined by Eqn. 3.1 to be within three standard deviations of the mean, thus providing a nominal value for γ. An additional parameter s, that is used for clustering is the number of samples expected within each cluster. We set this value to three, with the constraint that at least three points are necessary to rigidly construct a local reference frame for an object in $SE(3)$. Higher values of s can be set, however, this will require the detection of larger number of features that fall within each cluster. Table 3.2 summarizes the parameters and their values used in our motion segmentation implementation.

Name	**Value**	**Description**
γ_p	$\frac{1}{0.02\ m}$	Bandwidth parameter for spatial variation
γ_n	$\frac{1}{0.966\ rad}$	Bandwidth parameter for surface normal variation
s	3	The number of samples in a neighborhood for a point to be considered as a core point
ϵ	0.1	The maximum distance between two samples for them to be considered as in the same neighborhood

Table 3.2: A summary of parameters used in the motion segmentation implementation.

The resulting cluster assignments are denoted as $\mathbf{C} = \{C_1, \ldots, C_k\}$, where cluster C_i consists of a set of rigidly-moving feature trajectories. Figures 3-13 and 3-14 illustrate a simplified example of our motion segmentation approach with the use of fiducial markers.

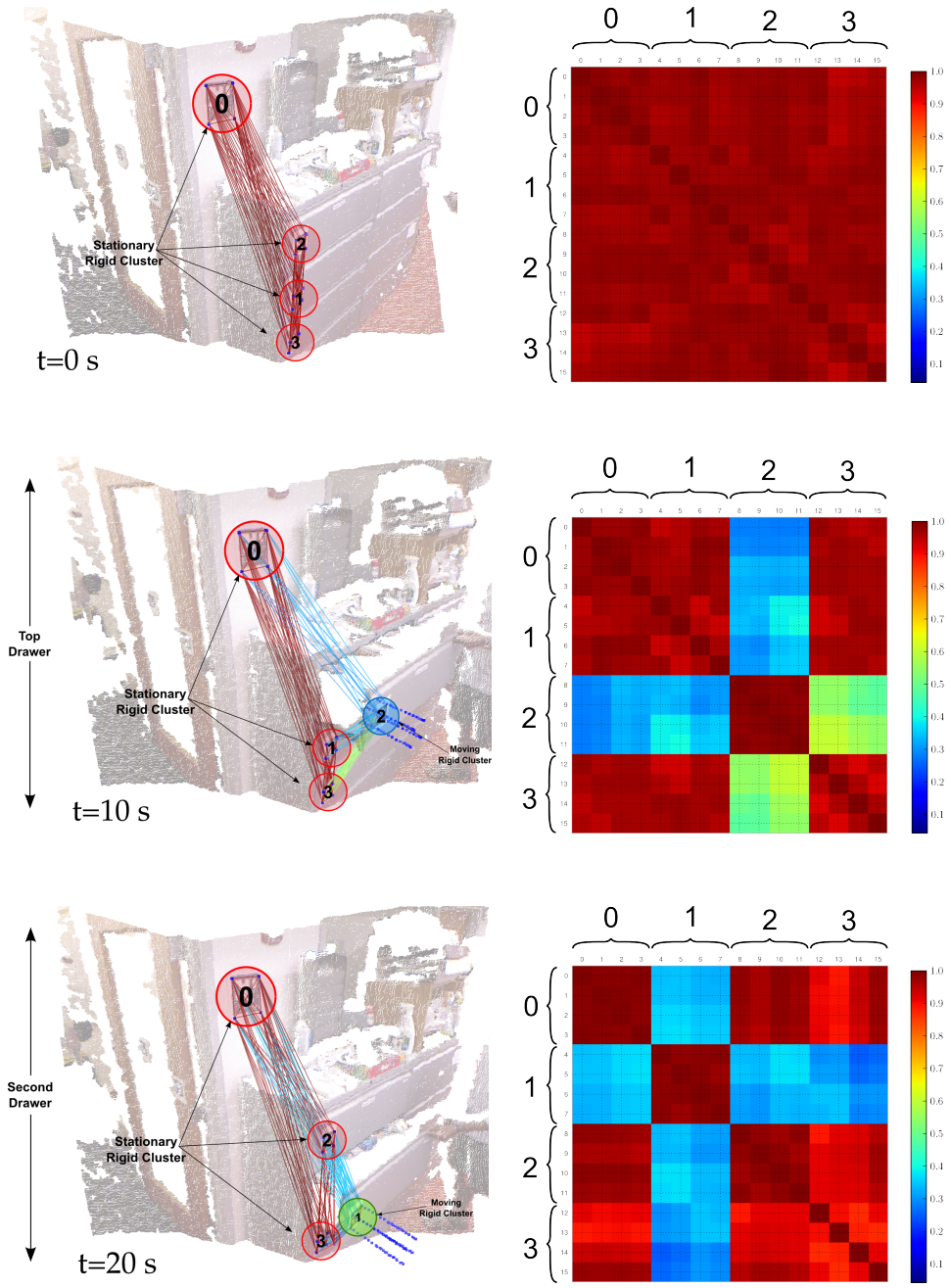


Figure 3-13: Motion segmentation for a drawer manipulation sequence. The top drawer (Cluster 2) is opened and closed at $t = 10s$. This is followed by the opening and closing of the second drawer (Cluster 1) at $t = 20s$. Color-coded edges overlaid on the articulated object indicate the motion profile similarity of trajectory pairs.

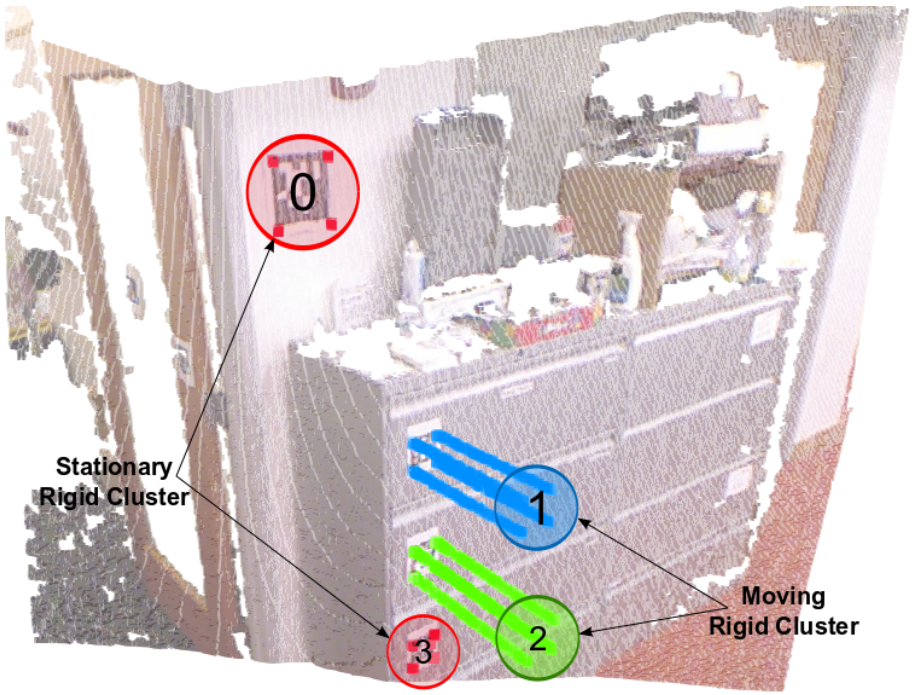
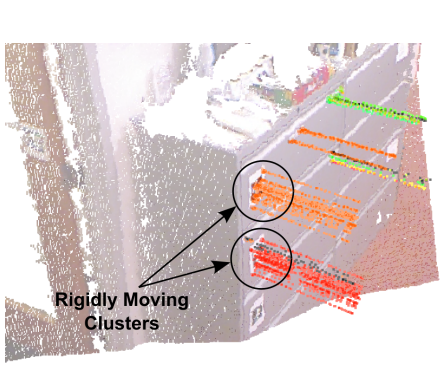
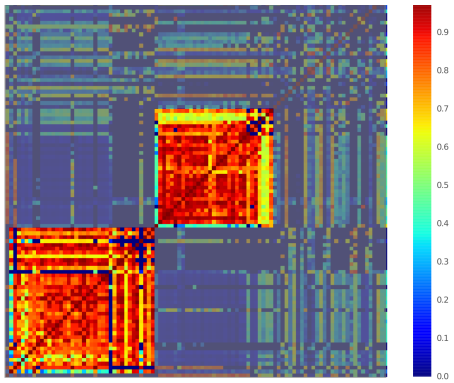


Figure 3-14: The resulting clustering accurately estimates 3 separate feature trajectory clusters, colored blue, green and red.

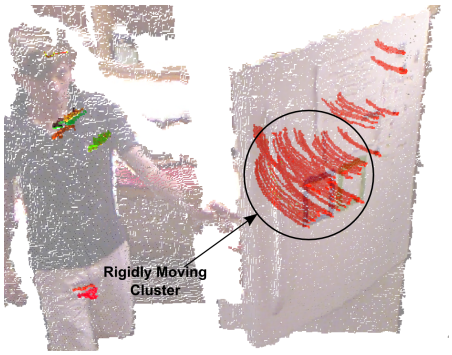
An advantage of the proposed motion segmentation is its use of feature-level relative motion estimates. This allows the algorithm in segmenting feature trajectories that initially move rigidly with respect to each other, but later exhibit non-rigid motion. One such example arises with articulated objects, where two joints move consistently with each other until one of the joints starts to move relative to the first joint. Since the pose tracking is performed on a fine-grained level, we are able to distinguish such relative motions, and estimate the new segmentation of the articulated link accordingly. Figure 3-15 illustrates the qualitative performance of the proposed motion segmentation algorithm on real-world RGB-D data.



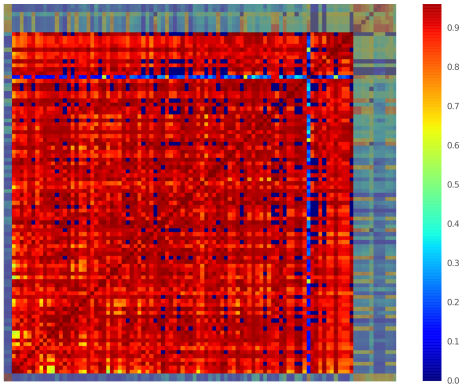
(a) Motion Segmentation



(b) Similarity Matrix



(c) Motion Segmentation



(d) Similarity Matrix

Figure 3-15: Segmentation of rigid-body articulated motions correctly identified by the trajectory clustering algorithm. In both cases, the number of rigid body motions involved during a demonstration, as suggested by the similarity matrices constructed, can be clearly identified.

Algorithm 2 Rigid Body Motion Segmentation via Pose-Pair Features

1: **Inputs:**
$\mathbf{F} = \{F_1, \ldots, F_n\}$: Sets of feature trajectories uniquely identified
Parameters: ϵ, γ_p, γ_n, s from Table 3.2
2: **Outputs:**
$\mathbf{C} : \{C_1, \ldots, C_k\}$, where $C_i = \{F_i, \ldots F_k\}$: Clusters of trajectories that move rigidly with respect to each other
3: **Procedure:**

- For each pair of trajectories, extract features that are synchronized by their associated feature timestamps.
- On the synchronized features, construct the pose pair distribution using Eqn. 3.1, 3.2, 3.3 for pairs of trajectories, and compute their spatial (W_p) and surface normal (W_n) similarity scores using γ_p and γ_n respectively.
- Construct the overall trajectory similarity matrix for all trajectory pairs using $W = min(W_p, W_n)$
- Pick the top k clusters using DBSCAN [10], with parameters ϵ and s

3.3 Multi-Rigid-Body Pose Optimization

Given the cluster label assignments for each of the feature trajectories, we subsequently determine the relative 6-DOF motion of each cluster. Since the trajectories on a finer-grained level are prone to sensor noise, we consider the optimization of the pose of a cluster as a whole. Additionally, we treat the 6-DOF motion of the cluster as a smooth continuous motion by adding a constant velocity-model assumption to improve the noisy 6-DOF pose estimates retrieved from the raw sensor observations. For an overview of the overall pose optimization algorithm, refer to Alg. 3 (p. 53).

3.3.1 Notation

Given the cluster label assignments for each of the feature trajectories, we subsequently determine the 6-DOF motion of each cluster. We define Z_i^t as the set of features belonging to cluster C_i at time t. Additionally, we define $\mathbf{X} = \{X_1, \ldots, X_k\}$ as the set of $SE(3)$ poses estimated for each of k clusters considered, and $x_i^t \in X_i$ as the $SE(3)$ pose estimated for the i^{th} cluster at time t.

3.3.2 Pose Estimation

For each cluster C_i, we consider the synchronized sensor observations of position and surface normals for each of its trajectories, and initialize an arbitrary pose x_i^1 to represent the reference frame in which the remaining pose estimates of the i^{th} cluster are defined. Subsequently, we compute the relative transformation $\Delta_i^{t-1,t}$ between successive time steps $t-1$ and t for the i^{th} cluster using the known correspondences between Z_i^{t-1} and Z_i^t. Our specific implementation employs a correspondence rejection step, that eliminates outliers falling outside the inlier distance threshold of 0.01 m, similar to the standard RANSAC [12] approach. This provides added robustness to the pose estimation routine in dealing with noisy sensor observations.

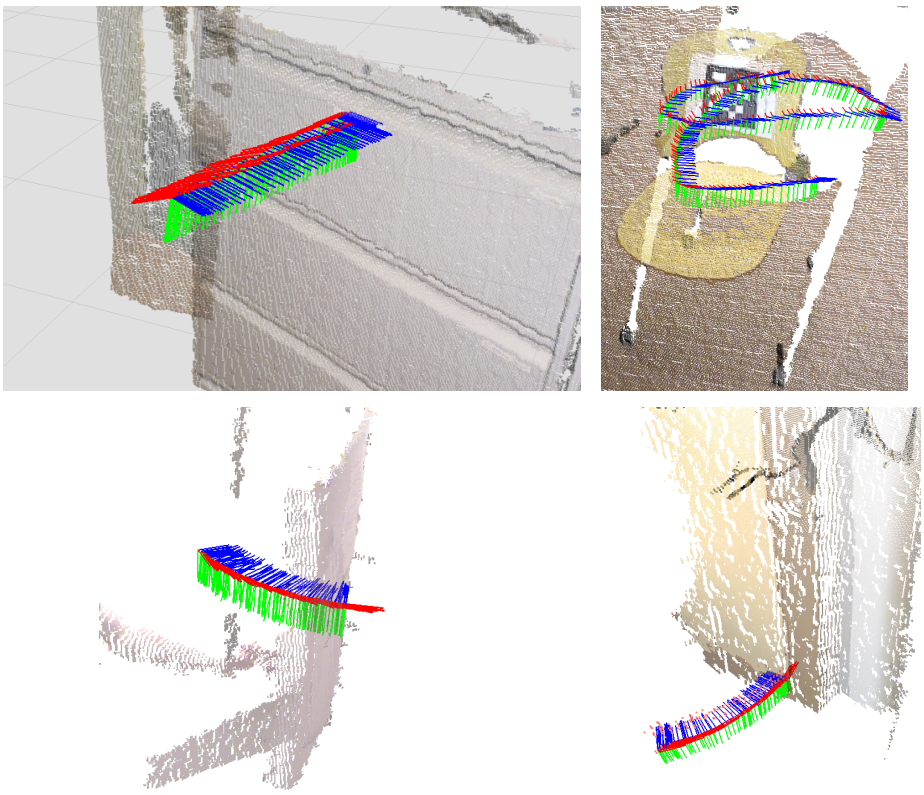


Figure 3-16: Visualization of pose estimates obtained after 6-DOF pose optimization using iSAM.

3.3.3 Pose Optimization

We augment the existing estimation step with an optimization phase to provide smooth and continuous pose estimates for each cluster by incorporating a motion model to the poses estimated. We use the 3-D pose optimizer iSAM [15], that uses a factor graph representation to define the relative pose constraints involved in the pose optimization. The node factors for the iSAM pose graph are added directly from the pose estimates computed earlier in the estimation step. A constant-velocity edge factor term is also added to provide continuity in the articulated motion. The pose optimization is then performed in order to retrieve the final pose estimates for each cluster. Figure 3-16 demonstrates a few examples of the pose optimization routine given the labeled feature trajectories.

Algorithm 3 Pose Optimization Algorithm

1: **Inputs:**
Set of rigidly moving clusters $\mathbf{C} = \{C_1, \ldots, C_k\}$
Parameters: `inlier_threshold` $= 0.02\ m$
2: **Notation:**
$\mathbf{X} = \{X_1, \ldots, X_k\}$, where $X_i = \{x_i^1, \ \ldots, x_i^t\}$
X_i: Set of poses for the i^{th} cluster or object part involved in the motion of the articulated object
x_i^t: Pose for object part i at time t
Z_i^t: Set of all points that belong to feature trajectories in C_i at time t.
N_i^t: Set of all surface normals that belong to feature trajectories in C_i at time t.
3: **Outputs:**
$\mathbf{X} = \{X_1, \ldots, X_k\}$: Optimized $SE(3)$ pose of the cluster with timestamps
4: **Procedure:** For each cluster C_i:

- Initialize the first pose x_i^1 with identity orientation and translation equivalent to $\frac{1}{\|Z_i^1\|} \sum Z_i^1$, the mean location of all the trajectories in C_i at time $t = 1$.
- For every subsequent time step t, compute the least-squares solution to the relative pose transformation $\Delta_i^{t-1,t}$ between Z_i^{t-1} and Z_i^t, given the known correspondences from the tracks. This step involves a **Correspondence Rejection Sa**mpling and **C**onsensus technique that removes outliers in Z_i^t that are not within `inlier_threshold` of the points in Z_i^{t-1}, while refining the final pose solution.
- Compute the pose ${x_i}^t = \Delta_i^{t-1,t} \oplus {x_i}^{t-1}$, and add ${x_i}^t$ as a node factor to the Slam3D pose graph optimization.
- In order to provide smooth and continuous pose estimates, a constant-velocity-model edge factor is added between each of the added node factors.
- Non-linear batch optimization is performed with a DogLeg trust-region method for added pose optimization robustness.

3.4 Articulation Learning

Once the 6-DOF pose estimates of the individual object parts are computed, the the underlying kinematic model of the full articulated object is determined using existing work from Sturm et al. [27]. Given a sequence of pose observations $\mathcal{D}_y$ of the articulated object, the most likely kinematic graph $\hat{G}$ is estimated, indicating the underlying articulation of the object. We evaluate several candidate articulation models for an articulated object that best explains the degrees of motion exercised during manipulation. In the following section, we formalize this as a probabilistic model estimation problem.

3.4.1 Notation

In our framework, we consider set of pose estimates of the individual object parts as the input to the articulation learning. Since the number of object parts k are unknown beforehand, we denote the true pose of an object part $i \in 1, \ldots, k$ by a vector $\mathbf{x}_i \in \mathbb{R}^3 \times SO(3)$ representing the 3D pose of the part in $SE(3)$. Thus, the full object pose is defined by $\mathbf{x}_{1:k} = (x_1, \ldots, x_k)^T$. We employ similar notation to that of Sturm et al. [27], and describe the relative transformation between two object parts i and j as $\Delta_{ij} = x_i \ominus x_j$. In our case, the motion composition operators would be $\oplus$ for compounded transformation and $\ominus$ for the inverse transformation. The kinematic model between part i and j is then defined as M_{ij}, with $\theta_{ij} \in \mathbb{R}^{p_{ij}}$, where p_{ij} are the number of parameters associated with the description of the link. We construct a graph $G = (V_G, E_G)$ consisting of a set of vertices $V_G = 1, \ldots, k$ that denote the object parts involved in the articulated object, and a set of undirected edges $E_G \subset V_G \times V_G$ describing the kinematic linkage between two object parts.

3.4.2 Problem Definition

Given multiple 6-DOF pose observations of object parts, the problem is to estimate the most likely kinematic configuration for the articulated object. In other words, we determine the kinematic graph $\hat{G}$ that best explains the observed poses $\mathcal{D}_y$ of the

articulated object. Formally, we estimate the kinematic graph configuration $\hat{G}$ that maximizes the posterior probability given by:

$$\hat{G} = \arg\max_{G} p(G|\mathcal{D}_y) \tag{3.5}$$

Equation 3.5 however, is difficult to evaluate due to the non-convex nature of the function, and due to the exponential parameter space that it needs to be computed over. As noted in Sturm et al. [27], we simplify the problem to that of recognizing kinematic trees of high posterior probability, and can reformulate it. Thus:

$$\hat{G} = \arg\max_{G} p(G|\mathcal{D}_y) \tag{3.6}$$

$$= \arg\max_{G} p(\{(\mathcal{M}_{ij}, \theta_{ij}) \mid (ij) \in E_G\} | \mathcal{D}_z) \tag{3.7}$$

$$= \arg\max_{G} \prod_{(ij)\in E_G} p(\mathcal{M}_{ij}, \theta_{ij}|\mathcal{D}_z) \tag{3.8}$$

3.4.3 Candidate Models and Model Fitting

Since we are particularly interested in articulated objects in daily household environments, we focus on a subset of kinematic models commonly seen such as rigid, prismatic, and rotational linkages. As mentioned earlier, we can also reproduce motions of a variety of household objects such as doors, drawers, refrigerators etc, with the combination of aforementioned kinematic model primitives. We summarize the model candidates that we use in our work in Table 3.3, and their associated degrees of freedom and model parameters.

Candidate Model M	**DOFs** d	**Parameters** p
Rigid	0	6
Prismatic	1	9
Rotational	1	12

Table 3.3: Candidate Models used for articulation modeling

We estimate the parameters $\theta \in \mathbb{R}^p$ that maximize the data likelihood of the

object pose observations given the kinematic model between the object parts, i.e.,

$$\hat{\theta} = \arg\max_{\theta} p(\mathcal{D}_z|\mathcal{M}, \theta) \tag{3.9}$$

Due to the high sensitivity of the least-squares-based parameter estimation to noise and outliers, we employ MLESAC [30] to provide the initial guesses for the kinematic parameters by picking a set of randomly drawn samples from the observation sequence which is then later refined via BFGS (Broyden-Fletcher-Goldfarb-Shanno) [6].

3.4.4 Structure Selection

Once we fit all the kinematic model candidates to the given observation sequence, we next select the kinematic model that best describes the data. More specifically, we compute the posterior probability of the kinematic models given the data as follows:

$$p(\mathcal{M}|\mathcal{D}_z) = \int \frac{p(\mathcal{D}_z|\mathcal{M}, \theta)\ p(\theta|\mathcal{M})\ p(\mathcal{M})}{p(\mathcal{D}_z)} d\theta \tag{3.10}$$

Due to the evaluation complexity of this posterior probability, we compute the BIC score instead as an approximation, with p as the number of parameters involved in the kinematic model, and n as the number of observations in the data set.

$$BIC(\mathcal{M}) = -2 \log\ p(\mathcal{D}_z|\mathcal{M}, \hat{\theta}) + p \log n \tag{3.11}$$

where $\hat{\theta}$ is the maximum likelihood parameter vector. This implies that the model that best explains the observations would correspond to that with the least BIC score, i.e.

$$\hat{\mathcal{M}} = \arg\min_{\mathcal{M}} BIC(\mathcal{M}) \tag{3.12}$$

In our work, we consider a fully-connected graph with $|V_G|$ vertices, and $|E_G| = \frac{|V_G|(|V_G|-1)}{2}$ edges. Each edge may be attributed to one of the 3 candidate kinematic models as listed in Table 3.3. Thus, the set of all possible kinematic trees that the

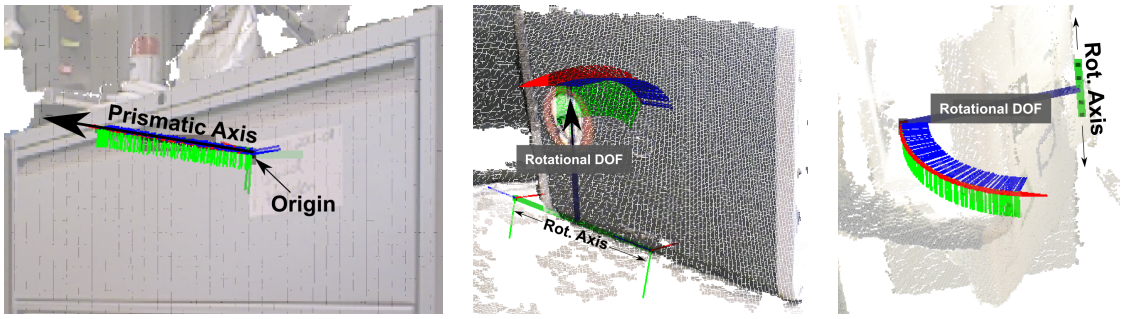
articulated object can take is given by the set of all spanning trees in the graph constructed. This subsequently leads us to finding the kinematic structure $\hat{E_G}$ that maximizes the posterior probability as follows:

$$\hat{E_G} = \arg\max_{\mathcal{E}_\mathcal{G}} p(E_G|\mathcal{D}_z) \tag{3.13}$$

$$= \arg\max_{E_G} p(\{(\hat{\mathcal{M}}_{ij}, \hat{\theta_{ij}}) \mid (ij) \in E_G\}|\mathcal{D}_z) \tag{3.14}$$

$$= \arg\max_{E_G} \prod_{(ij)\in E_G} p(\hat{\mathcal{M}}_{ij}, \hat{\theta_{ij}}|\mathcal{D}_z) \tag{3.15}$$

$$= \arg\max_{E_G} \sum_{(ij)\in E_G} \log\ p(\hat{\mathcal{M}}_{ij}, \hat{\theta_{ij}}|\mathcal{D}_z) \tag{3.16}$$



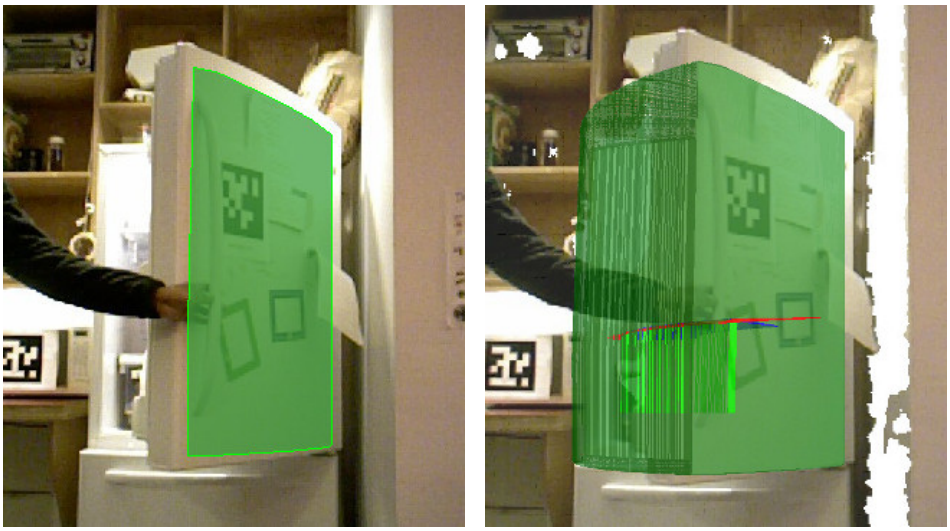
(a) Drawer (b) Laptop (c) Refrigerator Door

Figure 3-17: Examples of correctly estimated kinematic structure from 6-DOF pose estimates of feature trajectories.

We then reduce the kinematic structure selection problem into computing the minimum spanning tree of the graph (via Prim's or Kruskal's algorithm) with edges defined by $cost_{ij} = -\log\ p(\mathcal{M}_{ij}, \theta_{ij}|\mathcal{D}_{z_{ij}})$. The resulting minimum spanning kinematic tree weighted by BIC scores is the most likely kinematic model for the articulated object given the set of pose observations provided. Figure 3-17 shows a few examples of kinematic structures extracted given pose estimates as described in the previous section. We show good performance in reliably estimating the kinematic structure of the articulated object, and provide careful evaluation of several kinematic configurations that our system is capable of estimating in an unsupervised manner.

3.5 Learning to Predict Articulation

Our daily environment is filled with articulated objects, most of which we interact with on multiple occasions. One advantage of being in such environments is that it allows the robot to localize the instances of articulated objects that it may have encountered in the past. Given previous demonstrations of a particular articulated object instance being manipulated, our framework is able to localize the learned instance of the object, and subsequently predict its motion when manipulated.



(a) Extracted MSER (b) Estimated Motion Manifold

Figure 3-18: Figure illustrating the motion manifold of the articulated object, extracted via MSERs.

3.5.1 Object Instance Recognition and Prediction

Once the kinematic model of the articulated object is learned, the kinematic structure $\hat{G}$, its model parameters $\hat{\mathcal{M}}_{ij}, (ij) \in \hat{G}$ are stored in a database, along with its appearance model. The feature descriptors extracted (described in Section 3.1) for each cluster C_i of the articulated object are also retained for object recognition in future object encounters. Demonstrations involving the same instance of the articulated object are represented in a single arbitrarily selected reference frame, which is kept consistent across multiple demonstrations by registering newer demonstrations

in the original demonstration frame. Each of these attributes is stored in the database for convenient querying in the future.

Thus, on encountering the same object instance in the future, the robot can match the descriptors extracted from the current scene with those extracted from object instances it learned in the past. It then recovers the original demonstration reference frame along with the relevant kinematic structure of the articulated object for prediction purposes. Additionally, we identify the surface of the manipulated object by extracting the Maximally Stable Extremal Regions (MSER) [21], as shown in Figure 3-18 from each of the object parts undergoing motion. We use this surface to visualize the motion manifold of the articulated object. We restrict our recognition framework to object instances, and show that our framework is successfully able to recover the reference frame of the articulated object, and subsequently predict the motion that the articulated object takes when manipulated. These capabilities can be valuable especially in motion planning and object manipulation in future encounters.

Given several demonstrations of manipulating articulated objects such as doors, drawers, refrigerators, etc. our system is reliably able to recognize the objects in future encounters, and is also capable of providing a hallucination of its motion when manipulated. Figures 3-19, 3-20, 3-21, and 3-22 illustrate the recovery of different articulated object instances at independent future encounters.

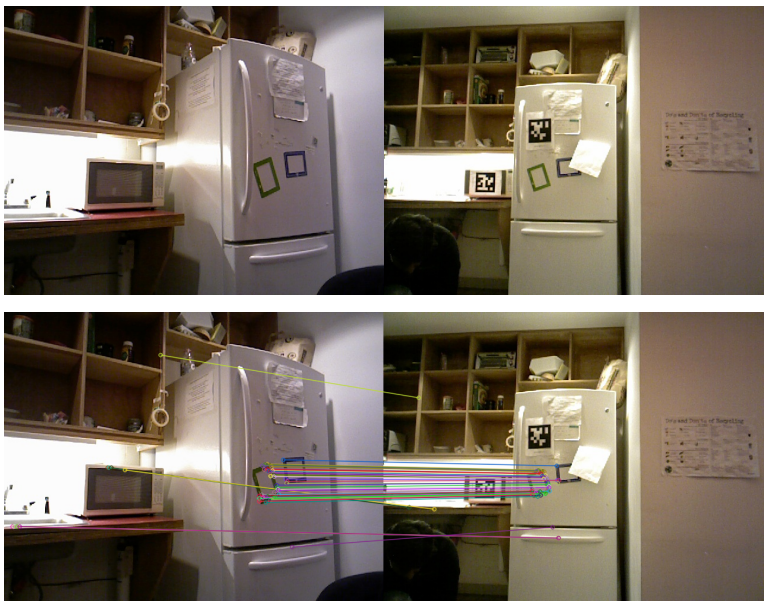
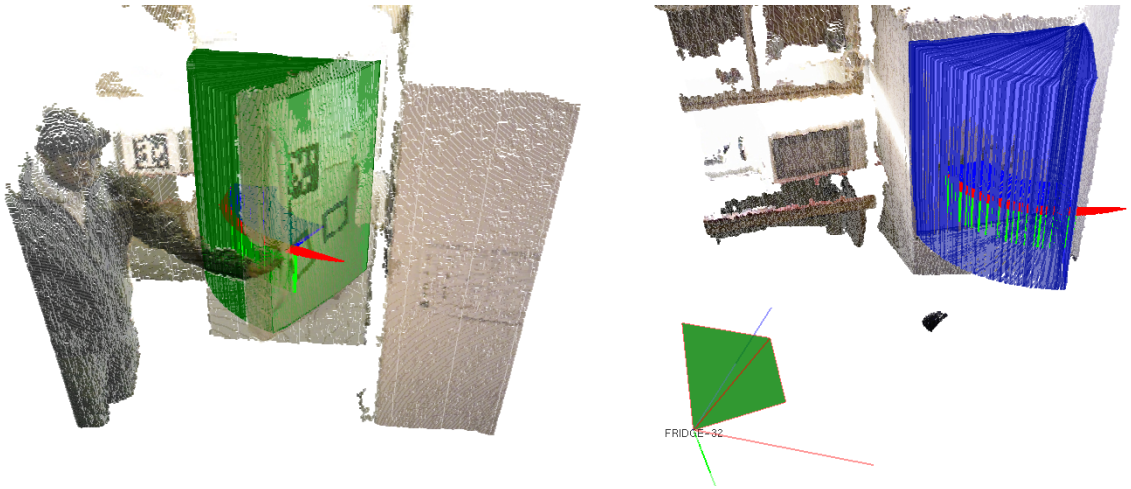


Figure 3-19: **Top:** Visualization of re-localization of the refrigerator at a future encounter. **Bottom:** The query instance of the fridge is matched with a previously learned demonstration.



(a) Motion manifold **learned**

(b) Motion manifold **predicted**

Figure 3-20: Given the co-registered viewpoints, the kinematic model of the refrigerator is recovered and projected in the query reference frame. The motion manifold predicted lies closely on the expected motion manifold of the object indicating a qualitatively good prediction.

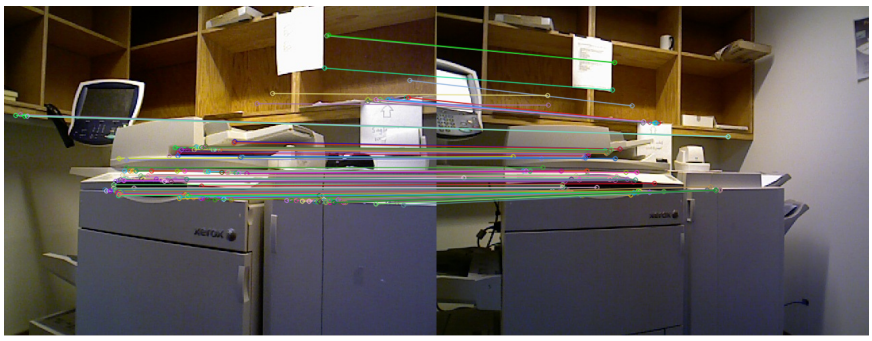


Figure 3-21: Printer instance re-localization via matched SURF features

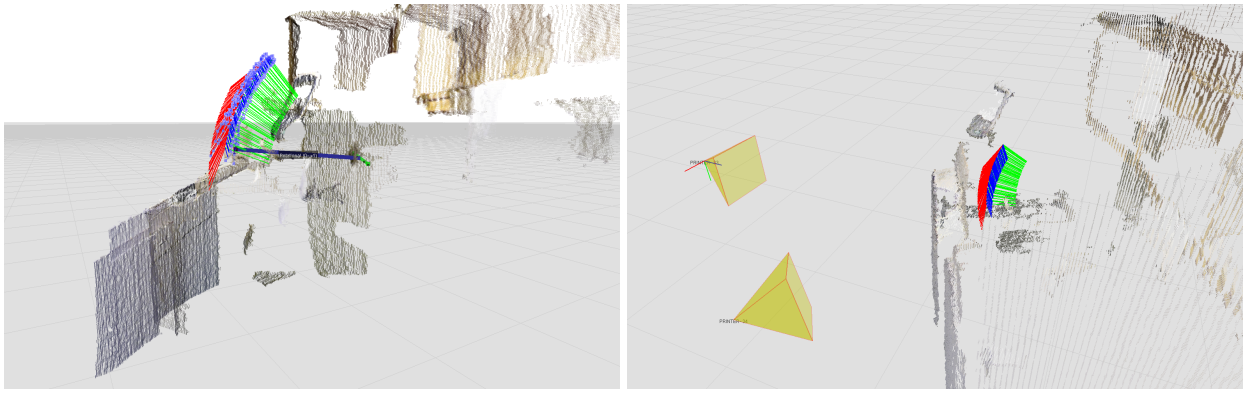


Figure 3-22: Prediction visualization shown in both states suggests a rotational motion of the printer on top (for scanner use)

3.5.2 Qualitative Results

Figure 3-23 shows the strong qualitative performance of our framework on various household objects including laptop, microwave, refrigerator, drawers etc. The system is capable of successfully determining the underlying kinematic model of several articulated objects by analyzing their motion during user-provided demonstrations. When the robot encounters an articulated object instance it has previously seen demonstrated, it is capable of recovering the kinematic model and parameters of the associated object in order to determine its expected motion.

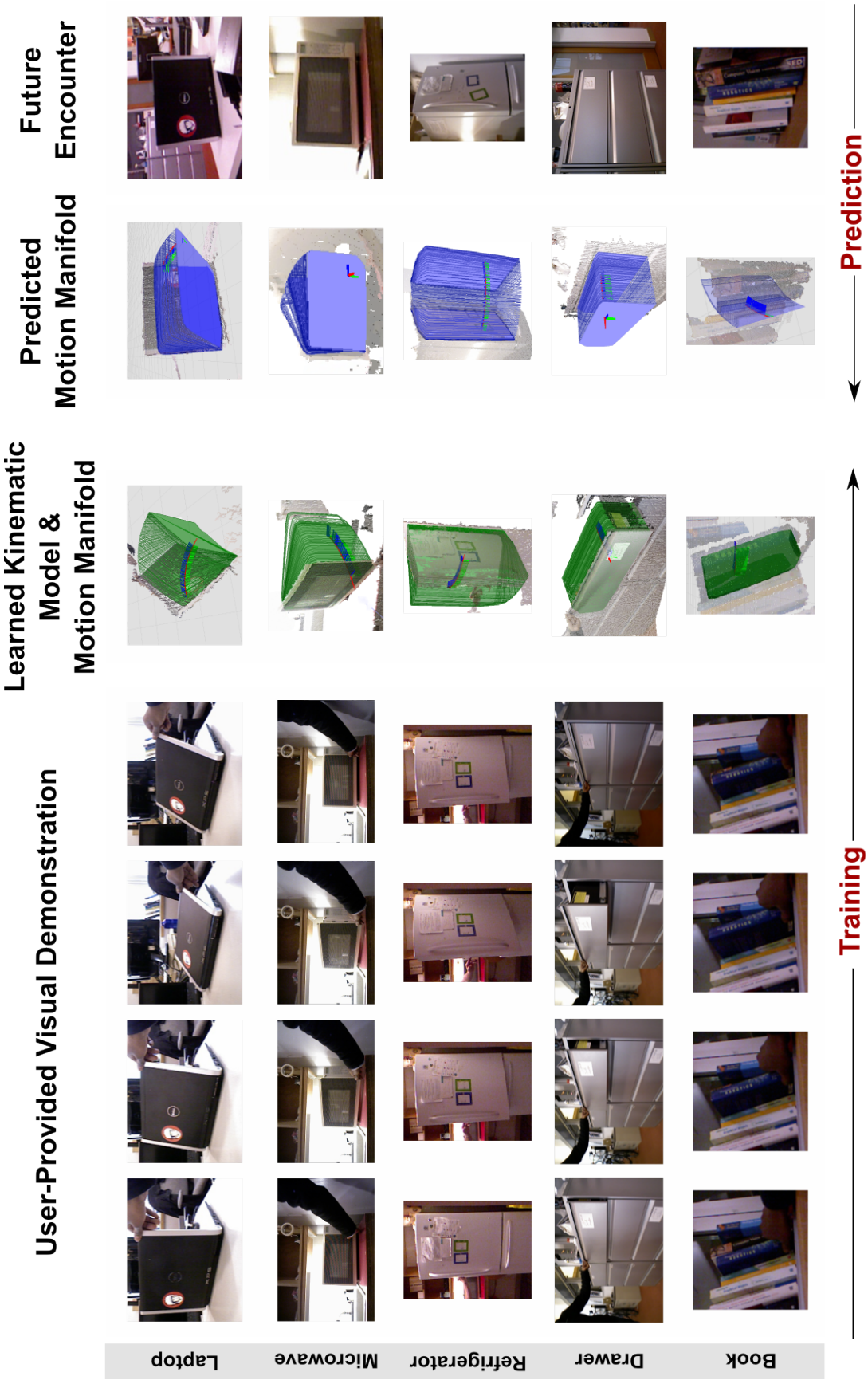


Figure 3-23: Figure showing the articulation learning and prediction capabilities of our proposed framework.

Chapter 4

Experiments and Analysis

4.1 Data and Experimental Setup

Our experimental setup consists of a single RGB-D sensor providing image and depth information. A visual demonstration involved a human teacher manipulating an articulated object and its parts, while providing sufficient visibility of the task to the robot. Additionally, each demonstration was repeated from three different view points to allow reasoning about the articulation model. We collected 30 sessions involving the visual demonstration of the human teacher manipulating different articulated objects commonly seen household environments, such as refrigerators, doors, drawers, cabinets etc. We use AprilTags [23] to provide ground truth estimates of the articulated object for evaluation purposes. In other words, the fiducial markers observed are used as a baseline to validate the performance of our framework with regards to pose estimation, kinematic model estimation and prediction. Special care is also taken to avoid any observations from the fiducial markers in our feature tracking pipeline, to provide a fair comparison.

4.2 Algorithm Evaluation

In order to validate our learning from visual demonstration framework, we first analyze its performance on simulated data. We evaluate our framework on its ability

to segment object parts and estimate its 6-DOF pose in an unsupervised manner, given a set of observations derived from a simulated articulated object. Using the known joint configuration, we simulate the motion of the articulated object by exercising all of its degrees of freedom. We compare the known kinematic model of the simulated object with that inferred by our framework, to assess our framework's accuracy. Subsequently, we evaluate our framework on learning articulated objects and their underlying kinematic models from real-world demonstrations. Additionally, with this reasoning capability, our framework can predict the motion of articulated objects at future encounters.

4.2.1 Learning with Simulated Data

In this section, we validate our articulation learning framework with simulated data. First, we simulate several kinematic joint configurations of articulated objects using Orocos' Kinematics and Dynamics Library (KDL) [1]. This is done by providing a Unified Robot Description Format (URDF) consisting of the joint configurations, and associated parameters of the individual joints in the articulated object. To provide sufficient variance in the articulations of objects, we simulate several kinematic configurations by chaining rigid, rotational, and prismatic joints together. We also assign velocity profiles to each of the individual articulated joints in order to assess the framework's capability to handle variability in articulated motion speeds. Additionally, we make no assumptions on the number of joints involved in the articulated object. This allows for chaining of kinematic links together to produce significantly different end-effector motions, thereby validating the true robustness of our proposed framework.

Motion Segmentation and Pose Estimation:

In the real-world scenario, the robot has no prior information on the number of object parts involved in the articulation, nor does it know the set of features that

[1] `www.orocos.org/kdl`

describe the object or its parts. As an onlooker, the robot can only infer the object part that is currently being manipulated, and must identify the different object parts involved in the manipulation. As described in earlier sections, we are primarily interested in inferring the articulation of an object given 3-D feature trajectories and their associated surface normals. In order to evaluate our framework with ground truth measurements, we construct a 5-DOF feature trajectory from the 6-DOF pose observations simulated for each kinematic joint configuration. Figure 4-1 shows the different steps involved in the validation of our framework with ground truth data. Given 6-DOF pose observations (**top left**) from a *Prismatic-Rotational* kinematic chain configuration, we sample a set of 5-DOF feature trajectories that emulate trajectories for features detected on an object part (**top right**). The reference frame shown on the bottom is connected to a reference frame in the middle via a prismatic link, which in turn is connected to another reference frame on top with a rotational link. The relative motion profiles of the individual feature trajectories are accurately analyzed and clustered based on rigid-body constraints as explained in the trajectory clustering section 3.2.2. The three object parts in Figure 4-1 (on the **bottom left**) are identified and clustered correctly with appropriate colors indicating unique labels. Given this feature trajectory segmentation, the algorithm then accurately estimates the 6-DOF pose of each cluster (on the **bottom right**) via a pose optimization step. The pose observations reconstructed are then directly used as inputs to the articulation learning algorithm to infer the kinematic structure of the articulated object.

Figure 4-2 shows the trajectory matching, clustering, and pose estimation for a *Rotational-Rotational-Rotational* kinematic chain configuration. An initial qualitative comparison as indicated by Figures 4-1, and 4-2 show the strong performance of the clustering, and pose estimation steps in our articulation learning pipeline.

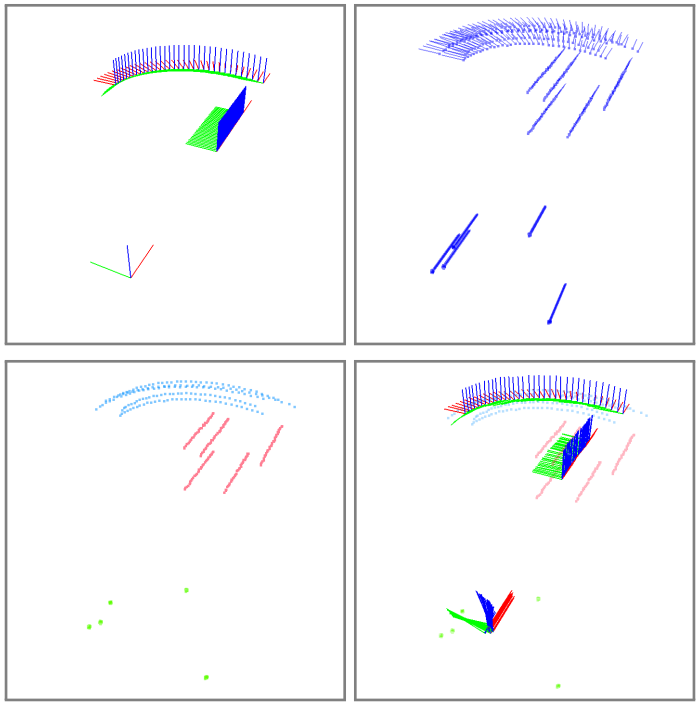


Figure 4-1: Visualizations of the intermediate steps in the evaluation of articulation of a simulated **Prismatic-Rotational** kinematic chain

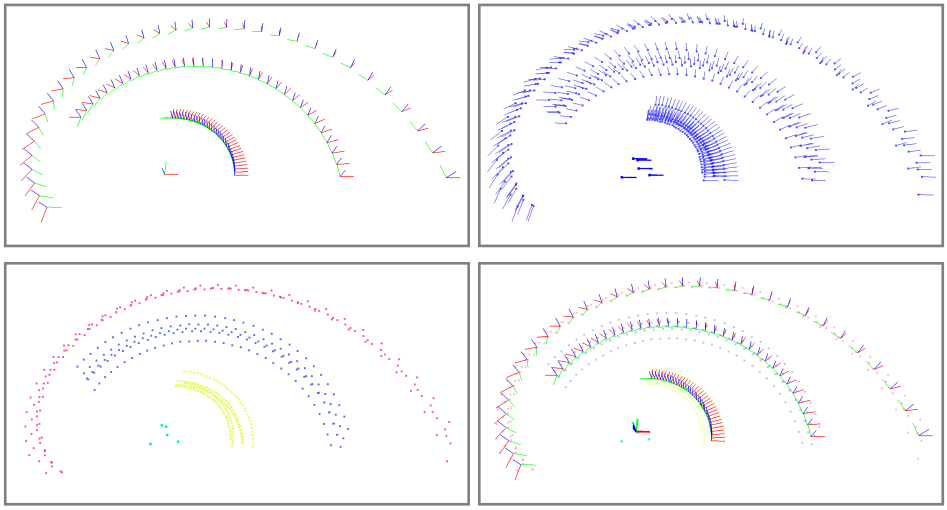


Figure 4-2: Visualizations of the intermediate steps in the evaluation of articulation of a simulated **Rotational-Rotational-Rotational** kinematic chain

Model	DOF	Components	Simulated Data	
			Pos. Error	Orient. Error
Prismatic	1	2	0.023 mm	0.013 rad
Prismatic-Prismatic	2	3	0.327 mm	0.030 rad
Prismatic-Rotational	2	3	0.493 mm	0.016 rad
Rotational	1	2	0.347 mm	0.017 rad
Rotational-Prismatic	2	3	0.676 mm	0.042 rad
Rotational-Rotational	2	3	1.323 mm	0.046 rad
Rotational-Rotational-Rotational	3	4	6.527 mm	0.033 rad

Table 4.1: Evaluation of position and orientation error associated with learning the underlying articulation in a simulated object, as compared to ground truth

Kinematic Model Estimation:

Once the 6-DOF pose estimates are extracted for each cluster, the motion manifold of the simulated object is learned to infer its underlying kinematic model. Figure 4-3 (on the **right**) shows the estimated kinematic model of the simulated object with its associated joint configuration parameters, alongside the ground truth kinematic chain structure. The articulation learning algorithm correctly predicts the structure of the *Prismatic-Rotational* kinematic chain, and estimates its joint configuration parameters with sufficient accuracy. The framework estimates a rotational radius of 0.18 m, with an average positional error of 0.001 m, and an average orientation error of 0.016 rad. Figure 4-4 shows our framework correctly identifying another kinematic chain structure, namely *Rotational-Rotational-Rotational*, and the associated joint parameters involved.

Table 4.1 summarizes the different joint configurations that were simulated and accurately identified by our framework. Additionally, the robustness of the articulation learning framework to noisy observations was also tested by perturbing the observations with a known noise factor. Table 4.1 reports the average position and orientation error of the fitted model resolved by our framework, given noisy 5-DOF

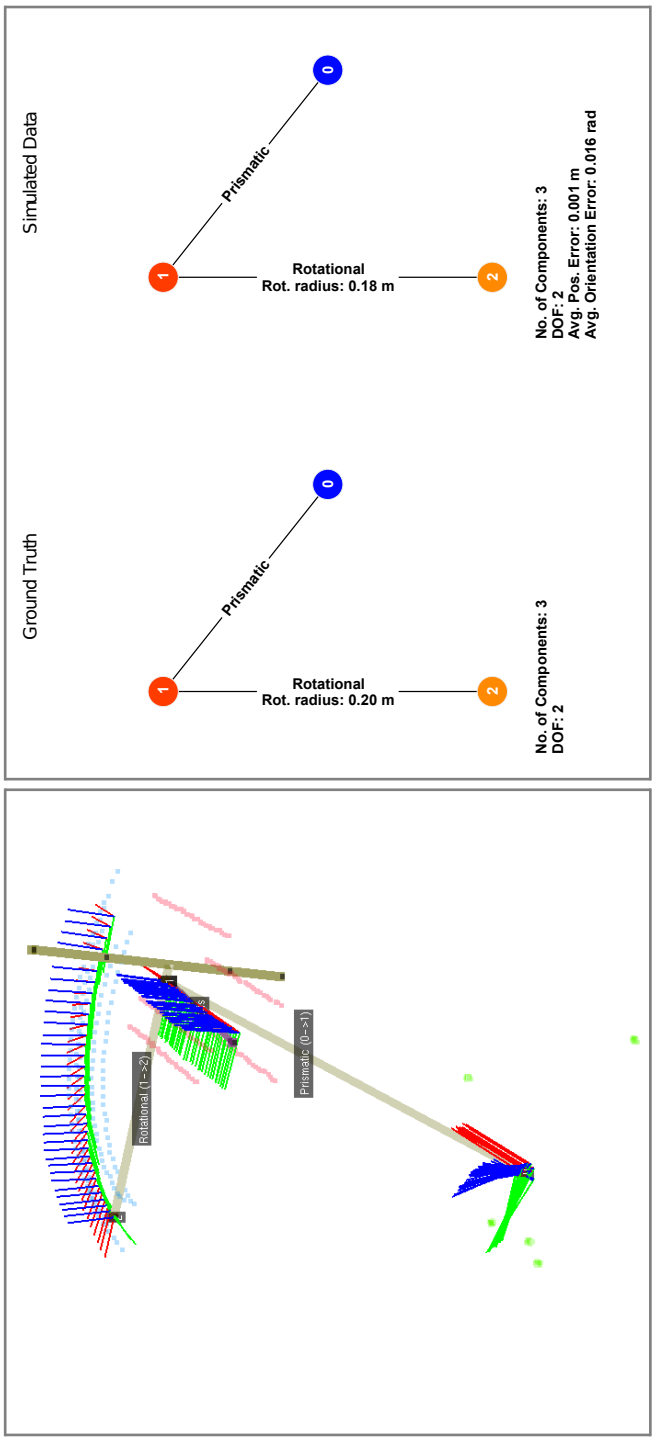


Figure 4-3: Comparison of kinematic graph estimated for a **Prismatic-Rotational** articulation model with simulated ground truth

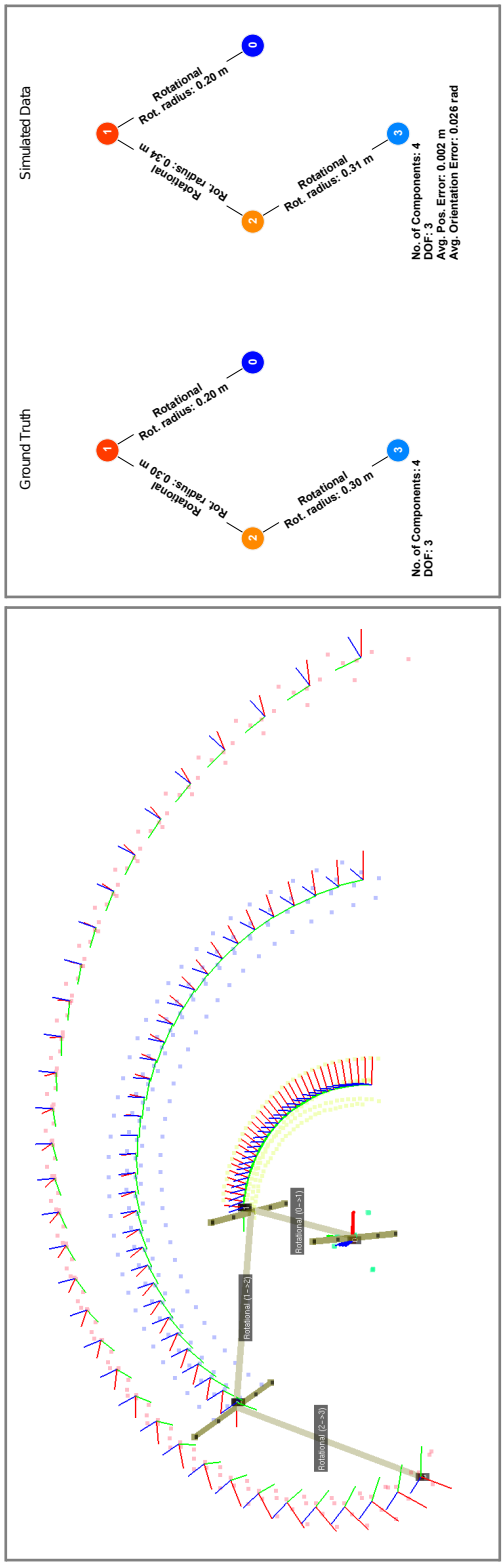
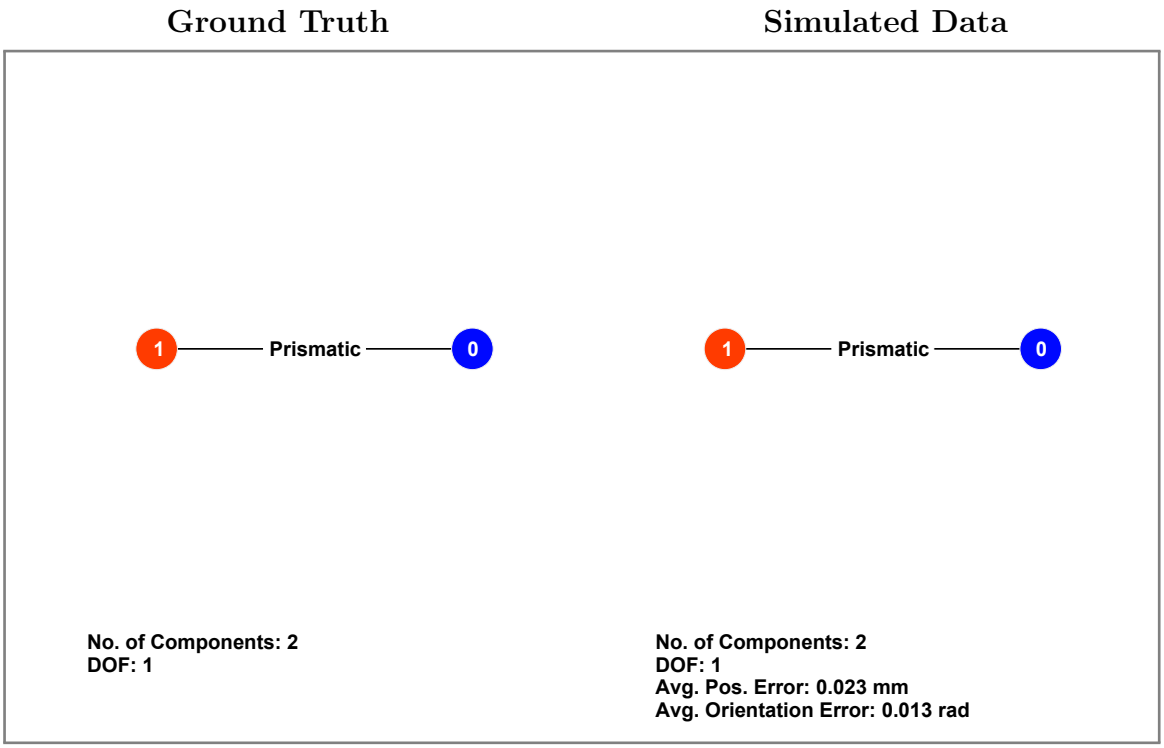
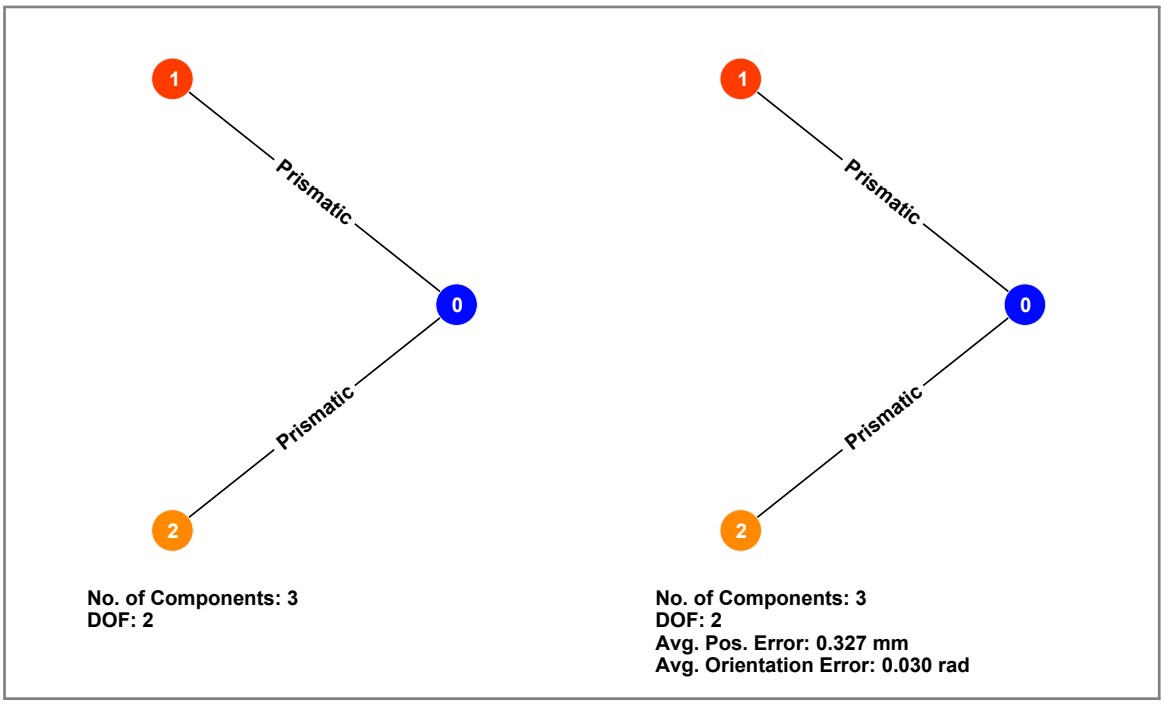


Figure 4-4: Comparison of kinematic graph estimated for a **Rotational-Rotational-Rotational** articulation model with simulated ground truth

observations sampled with $\sigma = 5$ mm spatial noise, from the simulated 6-DOF poses. Figures 4-5, 4-6, and 4-7 show the different joint configurations simulated and their corresponding kinematic structure correctly estimated by our framework. As seen in the figures, our framework correctly identifies the number of components, their degrees of freedom, and their underlying kinematic parameters in addition to the kinematic structure of the simulated object.

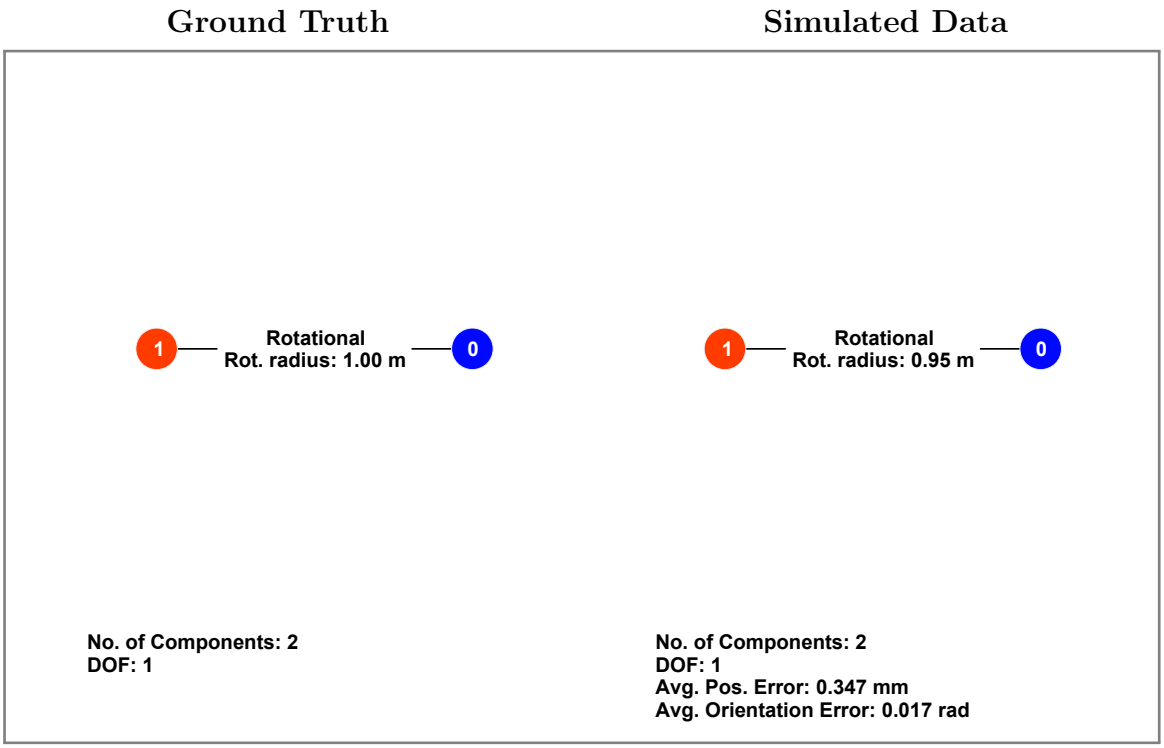


(a) Prismatic Link

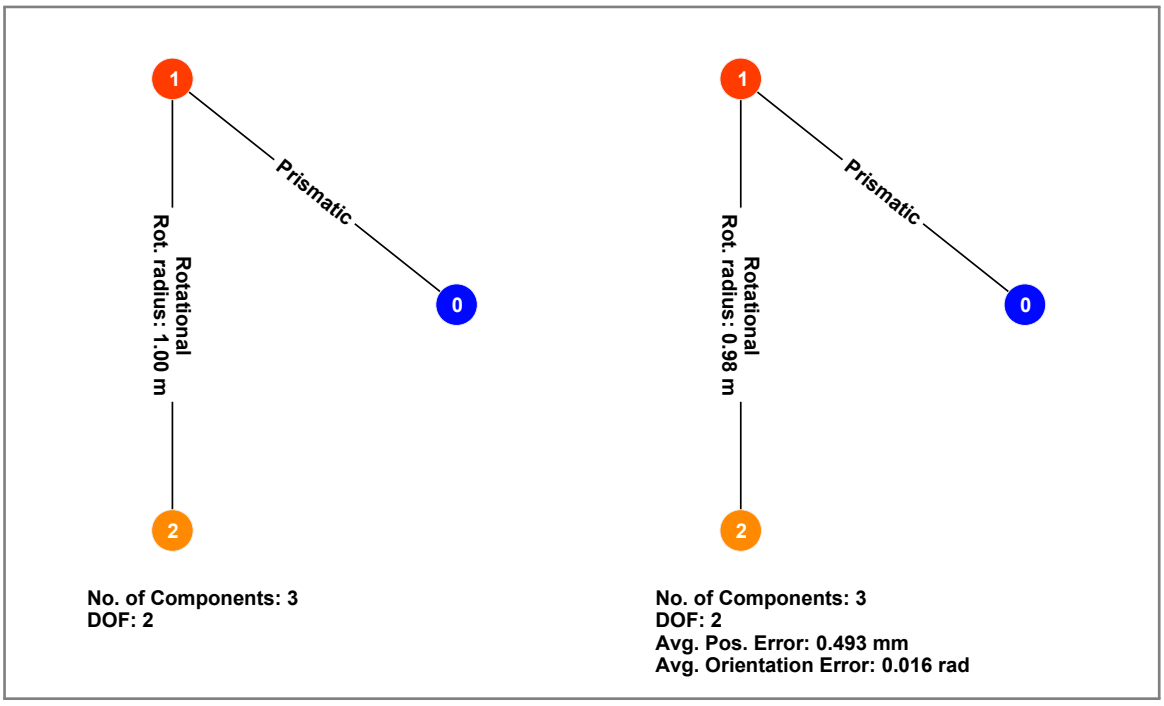


(b) Prismatic-Prismatic Chain

Figure 4-5: Kinematic structure (**Prismatic** and **Prismatic-Prismatic**) and model parameters estimated by the proposed articulation learning framework from simulated data, compared against known ground truth.

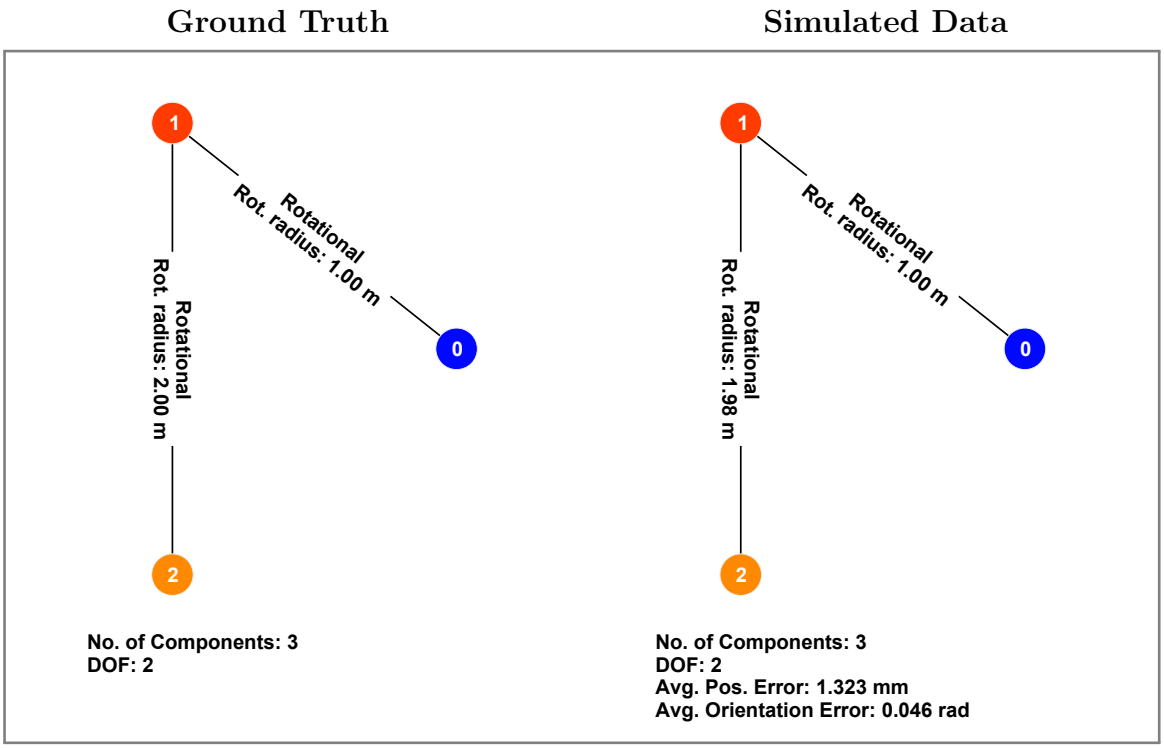


(c) Rotational Link

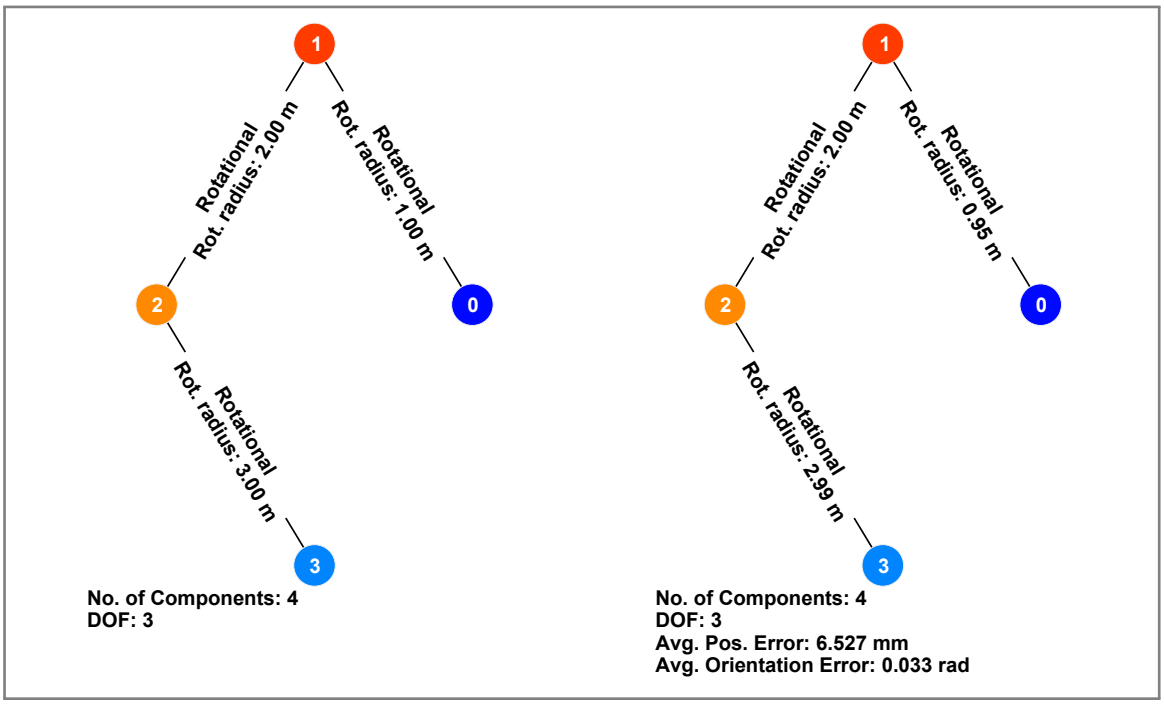


(d) Prismatic-Rotational Chain

Figure 4-6: Kinematic structure (**Rotational** and **Prismatic-Rotational**) and model parameters estimated by the proposed articulation learning framework from simulated data, compared against known ground truth.



(e) Rotational-Rotational Chain



(f) Rotational-Rotational-Rotational Chain

Figure 4-7: Kinematic structure (**Rotational-Rotational** and **Rotational-Rotational-Rotational**) and model parameters estimated by the proposed articulation learning framework from simulated data, compared against known ground truth.

4.2.2 Learning with RGB-D sensors

In order to enable robots that can learn from user-provided visual demonstration, we focus on harnessing the stream of high-fidelity RGB-D data. Due to its ready availability, our framework is tested and evaluated against data collected using the Microsoft Kinect.

We are particularly interested in robots that are capable of learning in unstructured environments, without the need for fiducial markers in the scene. We propose a marker-less feature tracker capable of providing rich 6-DOF pose information, without any prior knowledge of the object being tracked. We evaluate our feature tracker against existing feature trackers such as KLT, and Dense Trajectories, and use AprilTags as ground truth measurements to validate our marker-less articulation learning framework.

Data Pre-Processing

In order to evaluate the proposed marker-less articulation learning framework against the marker-based one, we initially remove all observations derived from the fiducial markers (AprilTags), before providing as input to our proposed algorithm. We do so to eliminate the bias in feature tracks provided by the structured features from the marker-based tracker. Therefore, as a pre-processing step to the marker-less tracking evaluation, the marker observations (AprilTags in our case) are detected and eliminated with the use of an in-painted mask image that prevents feature instantiation at these regions. Figure 4-8 shows the in-painted mask image constructed via AprilTag detection, and the corresponding feature extraction step in our pipeline that uses the input mask to avoid initializing features in the in-painted regions.

Figure 4-8: An example of the in-painted mask (in red) obtained via AprilTag detection, and the corresponding features extracted in the scene using the mask.

Feature Tracking Evaluation

In section 3.1, we present an improved spatio-temporal feature tracking algorithm that combines traditional dense optical flow methods with feature extraction and matching techniques to construct longer trajectories with little to no drift. In this section, we evaluate the feature trajectories constructed using our improved algorithm, and show its performance compared to existing feature tracking methods such as KLT, and Dense Trajectories. We refer the reader once again to Figure 3-8 for a qualitative comparison of feature drift observed in each of the algorithms described. Table 3.1 summarizes the parameters and their values used in our feature tracking implementation.

We construct feature trajectories on multiple video sequences, and determine the average trajectory lengths of all the features tracked with each of the aforementioned trackers. Due to the constant addition and removal of features in the feature trajectory construction step, the average trajectory length is computed over the top 30 percent of feature trajectories scored by their total trajectory lengths. In each of the video sequences shown in Figure 4-9, our feature tracker is able to construct longer feature trajectories on average than each of the feature trackers compared. In our comparisons, KLT performs comparably with our proposed algorithm. However, our feature tracker constructs up to 20 percent longer feature trajectories on average,

while exhibiting little to no drift. In their dense trajectories implementation, on the other hand, the authors perform dense sampling in the feature initialization step producing significantly greater number of trajectories, as indicated in Figure 4-10. Their implementation also re-initializes trajectories once they have attained a fixed length (15 in this case) in order to minimize drift. Additionally, we show in Figure 4-10 that our feature tracking algorithm maintains a reasonable number of feature trajectories over the course of the entire video sequence.

Pose Estimation Accuracy

In their previous work, Sturm et al. [27] uses the 6-DOF pose observations extracted via a marker-based tracker to learn the kinematic configuration of the articulated object. Additionally, the number of object parts involved in the observed articulation sequence was also known a priori. This reduces the space of possible kinematic configurations that the object can take.

Our work, on the other hand, does not make any assumptions on the number of object parts involved in the articulated object being manipulated. Furthermore, the feature trajectories and their corresponding object parts are subsequently in an unsupervised manner using the rigid-body relative motion constraint. Using these cluster assignments, we evaluate the accuracy of the 6-DOF pose measurements obtained via our tracking framework. We utilize AprilTags, to provide the sequence of $SE(3)$ pose observations for each of the object parts as ground truth for our accuracy evaluation.

For each visual lesson, the segmentation and $SE(3)$ poses of each object part is estimated and compared against the observed $SE(3)$ poses of the fiducial markers. The synchronization between the pose observations are ensured by only evaluating poses in the set intersection of the timestamps of the two observation sequences. Thus, for each demonstration, we obtain the $SE(3)$ pose of the fiducial marker, and the corresponding $SE(3)$ pose of the estimated object segment observed at exactly the same time. Subsequently, for each overlapping time step, we compare the relative pose of the estimated object segment with respect to the fiducial marker (in Figure 4-11).

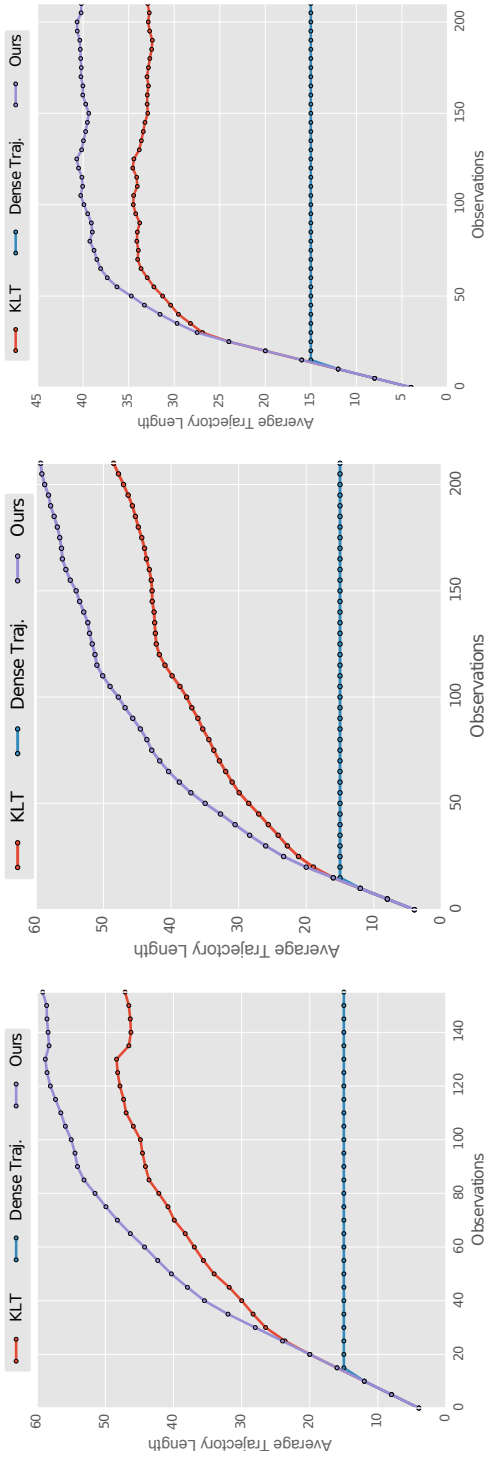


Figure 4-9: Average feature trajectory lengths constructed using KLT, Dense Trajectories, and our proposed feature tracker on the **Drawer**, **Refrigerator**, and **Chair** data sets.

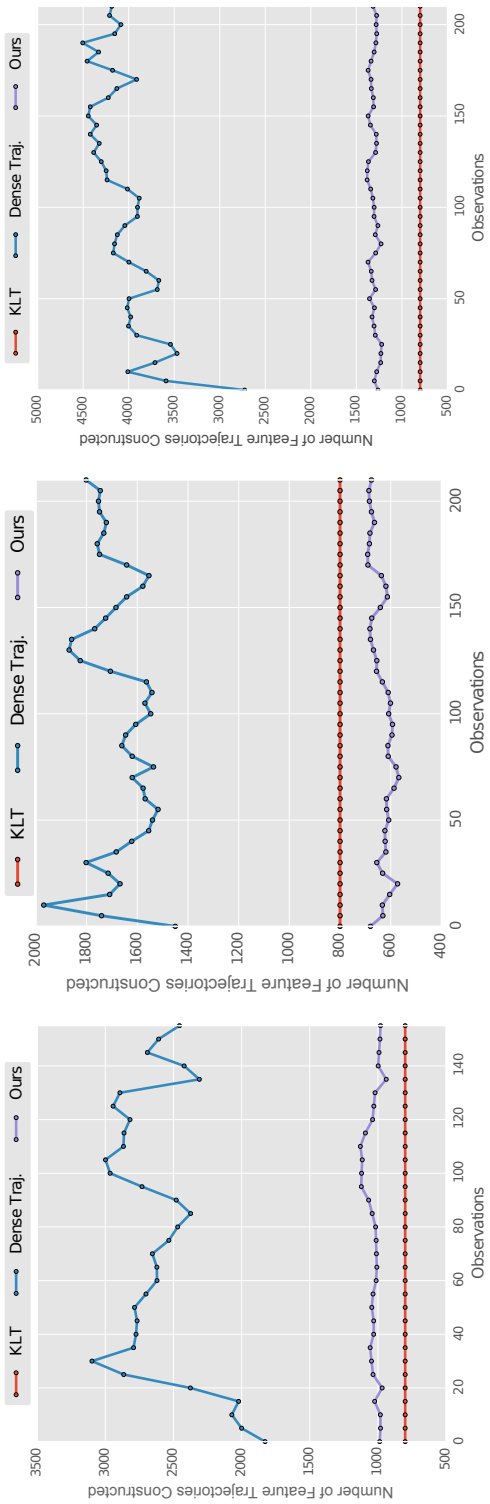


Figure 4-10: Total number of feature trajectories constructed using KLT, Dense Trajectories, and our proposed feature tracker on the **Drawer**, **Refrigerator**, and **Chair** data sets.

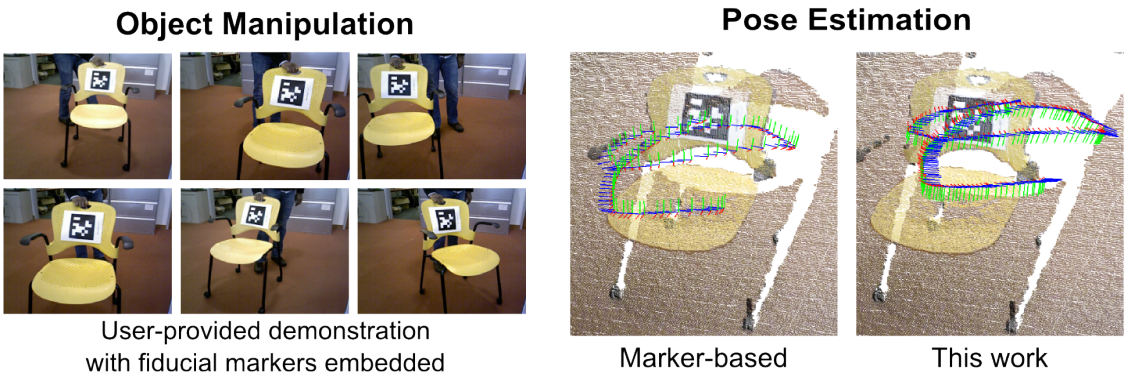


Figure 4-11: Figure illustrating the evaluation of pose estimation accuracy of our proposed framework with traditional marker-based tracking solutions.

We illustrate our results in Figures 4-12, 4-13, and 4-14 where we compare the absolute $SE(3)$ pose of an articulated object that is simultaneously tracked via April-Tags and our tracking framework, on separate visual demonstrations. For each of the tests, we are able to achieve accuracies consistently under $\pm$ 0.02 m spatially, and under $\pm$ 5 $^\circ$ in orientation with respect to fiducial markers despite the noisy observations provided by the Kinect RGB-D sensor. The gray bands indicate the maximum observed error between the $SE(3)$ pose of the articulated object determined via fiducial tags and our framework. One important observation is the high noise level in the observed position and orientation in Figure 4-12, between observations 25 and 45. Here, the angle θ subtended by the z-axis of the camera, and the surface normal of the object being observed is greater than 50 $^\circ$. As suggested in Nguyen et al. [22], the noise levels for the Kinect sensor asymptote at $|\theta| \approx 85^\circ$, with high inaccuracies in the depth registered at angles upwards of $|\theta| \geq 70^\circ$. As seen in Figure 4-12, the error in a few of the $SE(3)$ pose observations approach $0.04m$m spatially, and $\pm$ 20 $^\circ$ in orientation, due to the aforementioned noise levels in the Kinect sensor under certain viewing angles. Nonetheless, given the noisy RGB-D observations, our framework is able to robustly estimate the $SE(3)$ pose of the object with similar performance as compared to fiducial marker-based solutions.

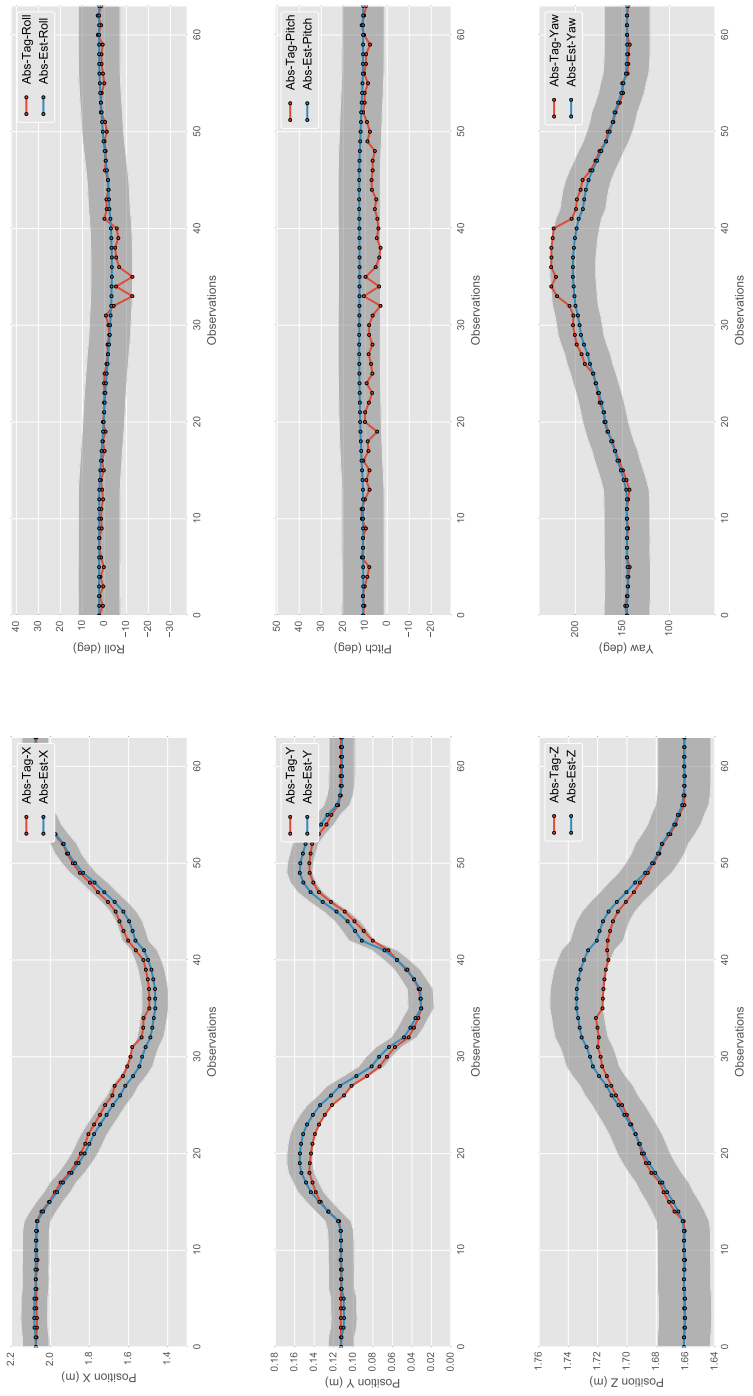


Figure 4-12: **Demonstration 1**: Comparison of $SE(3)$ pose estimates, while operating a refrigerator, determined via fiducial markers (Tag) and our framework (Ours). The gray shaded region indicates the maximum observed deviation of $SE(3)$ pose obtained via our framework from that observed via fiducial markers.

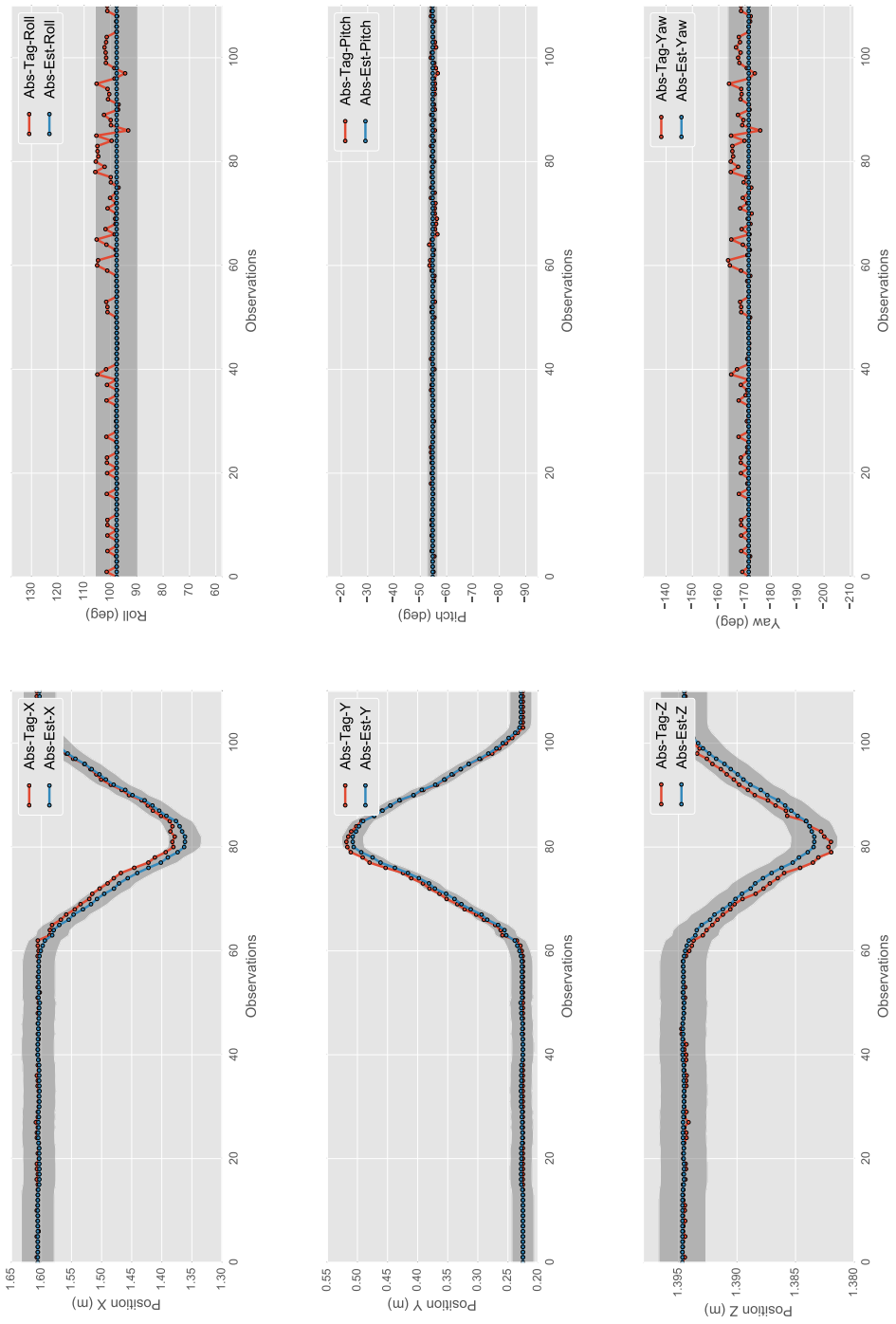


Figure 4-13: **Demonstration 2**: Comparison of $SE(3)$ pose estimates, while manipulating a drawer, determined via fiducial markers (Tag) and our framework (Ours). The gray shaded region indicates the maximum observed deviation of $SE(3)$ pose obtained via our framework from that observed via fiducial markers.

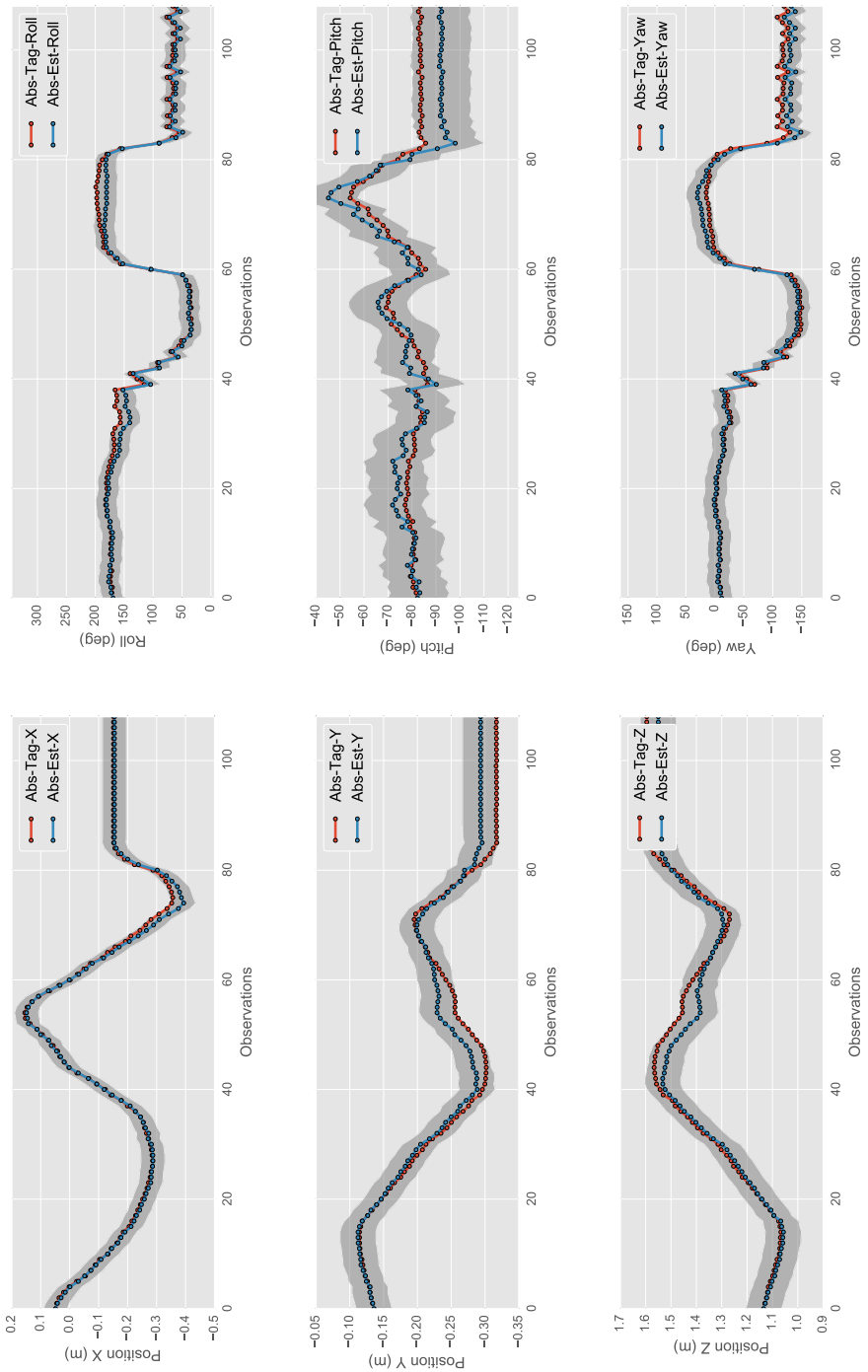


Figure 4-14: **Demonstration 3**: Comparison of $SE(3)$ pose estimates, while manipulating a chair, determined via fiducial markers (Tag) and our framework (Ours). The gray shaded region indicates the maximum observed deviation of $SE(3)$ pose obtained via our framework from that observed via fiducial markers.

Dataset	**Model**	**Error Type**	**Model Estimation** ***(AprilTags)***	**Model Estimation** ***(this work)***
Drawer	Prismatic		Prismatic	Prismatic
		Pos. error	0.003 ± 0.001 m	0.004 ± 0.002 m
		Orient. error	0.03 ± 0.01 rad	0.04 ± 0.02 rad
Refrigerator	Rotational		Rotational	Rotational
		Pos. error	0.006 ± 0.002 m	0.009 ± 0.002 m
		Orient. error	0.11 ± 0.08 rad	0.12 ± 0.07 rad
Microwave	Rotational		Rotational	Rotational
		Pos. error	0.003 ± 0.002 m	0.004 ± 0.002 m
		Orient. error	0.01 ± 0.02 rad	0.04 ± 0.02 rad
Door	Rotational		Rotational	Rotational
		Pos. error	0.002 ± 0.001 m	0.004 ± 0.001 m
		Orient. error	0.03 ± 0.01 rad	0.07 ± 0.03 rad

Table 4.2: Comparison of accuracies in kinematic model estimation determined using pose observations from AprilTags tracking against those estimated using the proposed framework.

Model Estimation Accuracy

Once the 6-DOF poses of the object parts are successfully estimated, we evaluate the kinematic structure and model parameters of the articulated object estimated via our framework. Again, we compare our results with those estimated using fiducial marker-based solutions. Table 4.2 summarizes the accuracy we are able to achieve with the proposed framework as compared to kinematic model estimation using fiducial marker-based solutions. On each of the evaluated demonstrations, our proposed framework correctly identifies the kinematic model of the observed object, and estimates its model parameters with accuracies similar to that of marker-based solutions. The average position and orientation error is computed by determining the average relative error between the projected pose described by the kinematic model, and the estimated poses provided by the pose estimation step. As indicated in Table 4.2, we are able to obtain model estimates that exhibit only up to 0.009 ± 0.002 m average positional error, and up to 0.12 ± 0.07 rad ($6.9 \pm 4.0\,^{\circ}$) average orientation error.

In addition to comparing the accuracy of model estimates, we also compare the model parameters determined by our framework with those estimated via marker-based solutions. The poses estimated via our framework are transformed in to the

Dataset	**Model Info**		**Model Parameters**	
Drawer	**DOF**	**Complexity**	**Pris. Origin**	**Pris. Axis**
Proposed Framework	1	8	(0.56,0.12,-0.81) m	(-0.75,0.03,-0.65)
Fiducial Markers	1	8	(0.57,0.07,-0.81) m	(-0.79,0.03,-0.60)
Refrigerator	**DOF**	**Complexity**	**Rot. Center**	**Rot. Axis**
Proposed Framework	1	9	(0.60,-0.18,1.73) m	(0.03,-0.99,0.12)
Fiducial Markers	1	9	(0.61,-0.22,1.71) m	(-0.02,-0.99,0.11)
Microwave	**DOF**	**Complexity**	**Rot. Center**	**Rot. Axis**
Proposed Framework	1	9	(-0.23,0.05,0.85) m	(0.01,0.99,-0.06)
Fiducial Markers	1	9	(-0.24,0.08,0.83) m	(0.05,0.99,-0.06)

Table 4.3: Kinematic model parameters determined using either methods show comparable results. With minimal supervision, our proposed framework can accurately identify the correct number of object parts, and estimate the true object pose to provide sufficiently accurate observations for articulation learning.

marker's reference frame based on the initial configuration of the articulated object. This allows us to directly compare model parameters estimated through our proposed framework and those estimated with the use of marker-based solutions. Table 4.3 summarizes the model parameters determined using our framework and fiducial markers on separate demonstrations. In each of the demonstrations, the model parameters estimated via our framework closely match those estimated with marker-based solutions.

4.2.3 Articulation Prediction

Over time, the robot perceives multiple user-provided visual demonstrations of different articulated objects in the household. Our framework learns the different articulated objects involved in each of the demonstrations, and reasons over their underlying kinematic model and appearance. This provides the robot with valuable information for future encounters, when it perceives the same instance of the object from a different (query) viewpoint. Given the known object appearance model, and its corresponding kinematic model, our framework is capable of predicting the motion of

identified object based on past experiences. Given sufficient image correspondences, the prediction algorithm successfully retrieves the relative view pose configuration between the learned and query viewpoints, and predicts the motion of the articulated object with reasonable accuracy and repeatability.

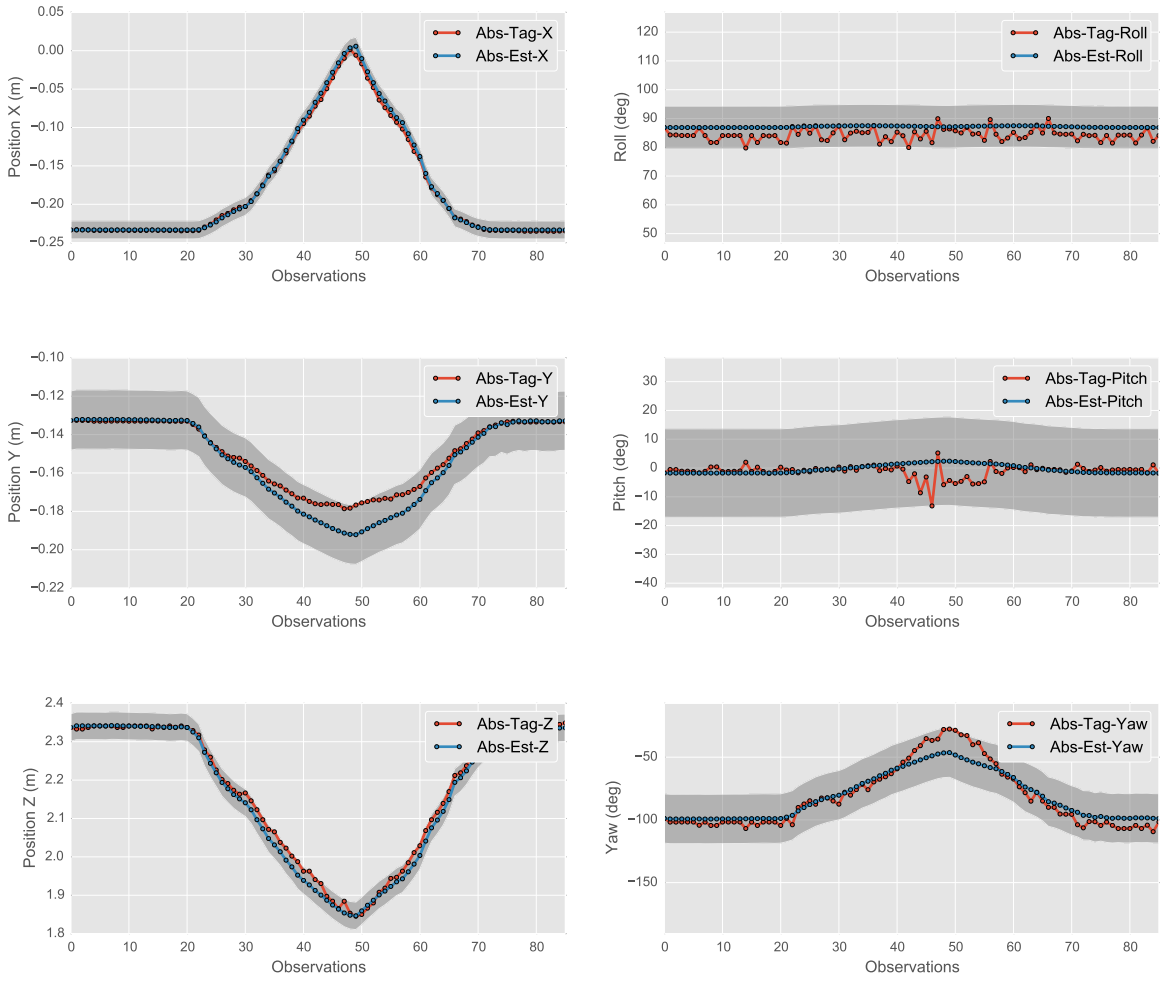


Figure 4-15: Comparison of spatial, and rotational accuracy of the $SE(3)$ pose **tracked** via AprilTags (Abs-Tag) against the **predicted** pose of the tracked object (Abs-Est). The gray shaded region indicates the maximum observed deviation of $SE(3)$ pose predicted via our framework from that observed via fiducial markers.

We evaluate the prediction algorithm by initially providing our framework with demonstrations of different objects being manipulated, observed from a variety of viewpoints. We can then compare the prediction accuracy of our algorithm against ground truth measurements provided by fiducial markers embedded on the manipulated objects. For a query viewpoint, our framework retrieves the appropriate model

of the articulated object, and predicts the motion that the articulated object may take, described by a sequence 6-DOF poses estimated from its expected kinematic model. Figure 4-15 compares the predicted motion manifold of a **refrigerator** via our prediction algorithm, and the tracked 6-DOF poses of AprilTags embedded on the object while it was manipulated. In the context of prediction accuracy, errors can be attributed to two primary sources: (i) the uncertainty error in the pose re-acquisition of the object instance from an RGB-D frame. (ii) the error in the estimated model parameters during the training phase. In our experiments, we assume that the underlying kinematic model and its model parameters learned during the training phase is accurate, and are able to achieve an average prediction accuracy of ± 0.02 m spatially and $\pm\ 5^\circ$ in orientation.

Chapter 5

Conclusion

In conclusion, we introduce a framework that enables robots to learn the kinematic model for everyday articulated objects based upon user-provided visual lessons, using an RGB-D sensor. There are three primary contributions of our approach that make it effective for articulation learning. Firstly, we proposed a novel feature tracking algorithm that bootstraps existing optical flow methods with traditional feature detection, and matching to construct longer range feature trajectories while incurring little drift. Secondly, we described a motion segmentation algorithm that uses a kernel-based approach to efficiently cluster feature trajectories that exhibit rigid-body motions. Additionally, we estimate the full 6-DOF pose of the articulated object and its parts in a robust manner with a final pose optimization step. Thirdly, we provide our robot with the capability to identify and estimate the underlying kinematic model of an articulated object, and subsequently, predict its motion in future instances using RGB-D data. We presented experimental results that demonstrate the effectiveness of our algorithm in learning the kinematic model and estimating the 6-DOF pose and articulation of several real-world objects.

Appendix A

Implementation

In this section, we provide a quick overview of the different components developed for our learning from visual demonstration framework. In a typical setting, we expect robots to learn almost immediately from experience, and use their understanding of the environment to make informed decisions. However, we consider an offline approach for this framework. To this end, we expect the robot to perform most of its reasoning when idling. Assuming that the robot has readily available logs of visual demonstrations it has previously experienced, the proposed framework utilizes this information to build its understanding of objects, and their underlying articulation model.

In order to tie in several components together with the capability to visualize intermediate results, we utilize existing image processing and geometric libraries such as OpenCV[1], and PCL[2]. The ability of the robot to reason over its immediate surroundings calls for strong performance capabilities. This in turn resulted in most of the algorithms to be written in C++. Additionally, due to the strength of rapid prototyping capabilities inherent in the Python programming language, and the availability of extensive machine learning toolkits, almost half of our framework was implemented in Python.

The implementation of our framework is split up into a real-time component and

[1] www.opencv.org
[2] www.pointclouds.org

a post-processing one. The real-time component is responsible for the trajectory construction and correspondence estimation as described in the feature tracking section 3.1. Feature-based detection and tracking was implemented in C++ with a combination of libraries including OpenCV and PCL. The post-processing component of our pipeline deals with the articulation learning and prediction components, and is mostly written in both C++ and Python. Training models involved post-processing recorded log files, and storing the results in an HDF5 database for convenient retrieval. Within Python, we use NumPy for most of the data representation, and Scikit-Learn for the DBSCAN implementation. Additionally, KDL[3], and iSAM[4] libraries were used for simulating kinematic chains, and pose optimization respectively. All our experiments were run on a 2.9GHz Intel Core i7, with 8MB L3 Cache. Post-processing the log files took approximately two minutes, with each log file being approximately 30-60 seconds visual lessons captured via an RGB-D sensor.

[3]www.orocos.org/kdl
[4]people.csail.mit.edu/kaess/isam/

Bibliography

[1] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEEE, 2012.

[2] Herbert Baya, Andreas Essa, Tinne Tuytelaarsb, and Luc Van Goola. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3): 346–359, 2008.

[3] Jean-Yves Bouguet. Pyramidal implementation of the affine Lucas-Kanade feature tracker description of the algorithm. *Intel Corporation*, 2001.

[4] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *Computer Vision–ECCV 2010*, pages 282–295. Springer, 2010.

[5] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):500–513, 2011.

[6] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[7] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010*, pages 778–792. Springer, 2010.

[8] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011.

[9] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE, 2009.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[11] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer, 2003.

[12] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[13] Alvina Goh and René Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *Computer Vision and Pattern Recognition, CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007.

[14] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317. Springer, 2012.

[15] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[16] Dov Katz and Oliver Brock. Manipulating articulated objects with interactive perception. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 272–277. IEEE, 2008.

[17] Dov Katz, Yuri Pyuro, and Oliver Brock. Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. In *In Robotics: Science and Systems*. Citeseer, 2008.

[18] Dov Katz, Andreas Orthey, and Oliver Brock. Interactive perception of articulated objects. In *The 12th International Symposium of Experimental Robotics (ISER)*, 2010.

[19] Dov Katz, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Interactive segmentation, tracking, and kinematic modeling of unknown articulated objects. 2012.

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

[21] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[22] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 524–530. IEEE, 2012.

[23] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

[24] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[25] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision*, 80(1):72–91, 2008.

[26] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[27] Jürgen Sturm, Vijay Pradeep, Cyrill Stachniss, Christian Plagemann, Kurt Konolige, and Wolfram Burgard. Learning kinematic models for articulated objects. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1851–1855, 2009.

[28] Jürgen Sturm, Cyrill Stachniss, and Wolfram Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research*, 41(2):477–526, 2011.

[29] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University. pages 1–22, 1991.

[30] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[31] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Computer Vision and Pattern Recognition, CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[32] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3): 177–280, 2008.

[33] René Vidal, Roberto Tron, and Richard Hartley. Multiframe motion segmentation with missing data using PowerFactorization and GPCA. *International Journal of Computer Vision*, 79(1):85–105, 2008.

[34] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

[35] Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *Computer Vision–ECCV 2006*, pages 94–106. Springer Berlin Heidelberg, 2006.