Occlusion-Aware Multi-View Reconstruction of Articulated Objects for Manipulation

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

---

by
Xiaoxia Huang
August 2013

Accepted by:
Dr. Stanley T. Birchfield, Committee Chair
Dr. Ian D. Walker
Dr. John N. Gowdy
Dr. Damon L. Woodard

Abstract

The goal of this research is to develop algorithms using multiple views to automatically recover complete 3D models of articulated objects in unstructured environments and thereby enable a robotic system to facilitate further manipulation of those objects. First, an algorithm called Procrustes-Lo-RANSAC (PLR) is presented. Structure-from-motion techniques are used to capture 3D point cloud models of an articulated object in two different configurations. Procrustes analysis, combined with a locally optimized RANSAC sampling strategy, facilitates a straightforward geometric approach to recovering the joint axes, as well as classifying them automatically as either revolute or prismatic. The algorithm does not require prior knowledge of the object, nor does it make any assumptions about the planarity of the object or scene.

Second, with such a resulting articulated model, a robotic system is then able to manipulate the object either along its joint axes at a specified grasp point in order to exercise its degrees of freedom or move its end effector to a particular position even if the point is not visible in the current view. This is one of the main advantages of the *occlusion-aware* approach, because the models capture all sides of the object meaning that the robot has knowledge of parts of the object that are not visible in the current view. Experiments with a PUMA 500 robotic arm demonstrate the effectiveness of the approach on a variety of real-world objects containing both revolute and prismatic joints.

Third, we improve the proposed approach by using a RGBD sensor (Microsoft Kinect) that yield a depth value for each pixel immediately by the sensor itself rather than requiring correspondence to establish depth. KinectFusion algorithm is applied to produce a single high-quality, geometrically accurate 3D model from which rigid links of the object are segmented and aligned, allowing the joint axes to be estimated using the geometric approach. The improved algorithm does not require artificial markers attached to objects, yields much denser 3D models and reduces the computation time.

Dedication

I dedicate this work to my beloved family who made all of this possible for their endless encouragement and faith.

Acknowledgments

First, I would like to give my most heartfelt thanks to my adviser, Dr. Stanley Birchfield for his numerous guidance and support at every step of this work. His deep love and insight of science inspired me in the right direction and made my journey a truly enjoyable learning experience which will guild me throughout my life. He is always patient, encouraging and enlightening. Additionally, I am very grateful to Dr. Ian D. Walker, Dr. John N. Gowdy, and Dr. Damon L. Woodard, for their valuable input and instruction in directing the research to this point.

My gratitude is also extended to all the members of my research group who directly and indirectly provided helpful discussion, and assistance. Also I would like to thank all my friends at Clemson supporting me all the time.

Finally, I would like to thank my family for their immense love, support, and patience.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Robot manipulation

Since Unimate, the first industrial robot, designed by George Devol for a production line at the General Motors Ternstedt plant in Trenton, NJ, in 1961, robots have been widely used in a variety of areas such as manufacturing, medical, services, environment, transportation, entertainment and education in the past half century [3, 29, 68]. Figure 1.1 shows several examples of robots. As the primary application, industrial robots are used to perform operations such as welding, assembly, painting, picking and placing quickly, repeatedly and accurately in tedious and dangerous manufacturing environments to improve safety and efficiency of production and reduce environmental impact. Over the last three decades robotics has been integrated by medical industries to aid surgeons, and augment healthcare rehabilitation or training using more precise and less invasive methods. In the domestic applications, people have had increasing needs for robots to assist their daily lives for example cleaning houses, mowing lawns or delivering stuff in order to improve the quality of life. Since robotics involves multiple disciplines like mathematics, physics, computer science and

Figure 1.1: Examples of robots: Surgical System Robot (DaVinci), UMass Mobile Manipulator (UMass Amherst), Mirra Pool-cleaning Robot (iRobot), Nao Humanoid Robot (Aldebaran).

so forth, robotic toys and educational toolkits are widely introduced to help students to deeply understand basic concepts of such disciplines and inspire students to design, innovate and solve problems.

Up to now, the development of robots is roughly divided into three stages [68, 69]. The first generation is to perform simple and predictable tasks in constrained industrial environments, for example robotic arms, which have similar functions of human arms, assembling cars, handling machine tools or packaging food boxes. However, for real world applications functionality and environments often change. People need robots to have capabilities to serve autonomously. In order to satisfy these needs, the second generation incorporates a sensor system and a computer into a more complicated control system which analyzes various environmental information collected by the sensor system and plans corresponding operations for execution by the computer system [68]. The behaviors of such robots are largely limited by their control systems which memorize the knowledge about their structured surroundings. As robots move into unstructured environments such as homes, schools, and workplaces, it is unrealistic to expect robots to have advanced knowledge of all objects that will be encountered in the physical world, and they may not be able to function autonomously. Therefore the current generation introduces artificial intelligence into

the control system to enable robots to fully serve autonomously, or at least semi-autonomously in unstructured, dynamic environments. This transition is perhaps most apparent in the nascent emergence of socially assistive and service robotics applications in which robots help people in their homes and workplaces with basic tasks such as cleaning, personalized care, and behavioral therapy. Such application areas are expected to represent a key area of growth for the robotics industry for the coming years.

To handle unstructured, unpredictable environments robots face many challenges in several aspects. At first, robots must improve their mobility to perform tasks. Currently most robots navigate in an environment either by using a priori map of the complete surrounding or building the environmental map as they move through it [7]. Such map-based navigation systems only work in specific places. However, real world environments have much variability and uncertainty [3, 47]. To deal with these difficulties, robotic systems need to introduce new representations of the environment such as 3D maps, use novel sensing models like RGBD sensors or improve current localization algorithms.

Second, dynamic and uncontrolled environments make robots manipulation such as opening doors, doing laundries or cleaning kitchens more challenging [59, 60, 39]. To successfully interact with its surrounding, robotic systems typically make assumptions of known features or models of objects in a scene. Even in a structured environment, manipulation tasks are not easy with such assumptions. For example in a cleaning-kitchen robotic system, it is not realistic to have knowledge of all types of dishes and cups. Therefore real world environments impose many difficulties on robot manipulations due to their uncertainty and complexity. In order to perform tasks and manipulations in open and unstructured environments in which prior knowledges and models are not available, robots need to develop abilities to actively learn about the

environments.

Third, robotic systems need to increase their sensing abilities to function autonomously in unstructured environments. The goal of a robot sensor system is to collect all information of its surrounding by "seeing," "touching," "feeling" and "hearing." Visual sensing serves as the "eyes" of robots which typically analyze and process images captured by cameras or other visual sensors to understand environments. Visual sensing is one of the most promising ways to explore and learn about the environment. Generally, computer vision techniques are used to analyze the sensory streams in a passive manner, and recently significant progress has been achieved with feature detectors and 3D reconstruction techniques. It is only natural to investigate how to apply techniques from computer vision to robotics applications rather than concentrate on only visual sensing or machine manipulation separately. The notion of active vision is that for some tasks the sensing problem can actually be made more tractable by actively affecting visual streams by controlling the sensors.

However, in some cases the sensing system fails because of noisy sensor data and ambiguities of real world. For example, due to noise it is hard to detect correspondences between images which is a very common problem in robotic applications, or robots may not recognize different objects such as plums and apples because they have similar appearances in some viewpoints of cameras [3, 47]. Such problems are difficult even in a fixed environment. They can be partially solved by adding more constraints about objects' position, color, dimension and other features. To handle unstructured, unpredictable environments, new sensing devices and approaches will be needed. In particular, rather than assuming that the robot has advanced knowledge of all the objects that will be encountered, the robot must be able to actively learn about its environment in order to effectively manipulate within it.

1.2 Articulated objects

Within the context of learning about the environment, one problem that has caught the attention of robotics researchers recently is that of reconstructing *articulated* objects [48, 89, 90, 91]. An articulated object can be modeled as a set of rigid links connected by one or more joints, either revolute or prismatic. Much of the information about articulated objects is encoded in the relative motion of objects such as human limbs moving with respect to the body, vehicle wheels moving in different ways from the main body of the vehicle, and so forth. Many applications such as household, elder care robots require the manipulation of such objects. While current manipulation systems often assume a priori object models, articulated objects pose an additional problem in that their structure changes dynamically. In order to perform manipulation tasks with such objects, it will be necessary to develop models that capture their articulation behavior.

Figure 1.2 shows the two object models for simplified cases of just two links and one joint. In the case of a revolute joint, the configuration between the two links is represented by the joint angle, while in the case of a prismatic joint, the configuration is represented by the displacement. A surprisingly large number of important objects encountered every day can be modeled in this fashion, such as refrigerators, microwave ovens, drawers, doors, laptop computers, scissors, staplers, and so forth. Another widely encountered articulated object is the body of humans or animals, the reconstruction and pose estimation of which have been of great interest to researchers in both computer vision and graphics [110, 74, 84]. Figure 1.3 shows several examples of articulated objects.

Reconstruction of 3D scenes from images has been an active research area in the computer vision community for decades. Tremendous progress has been made in

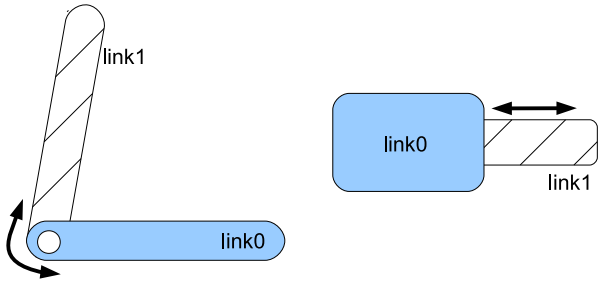

Figure 1.2: LEFT: Two rigid links connected by a revolute joint. RIGHT: Two rigid links connected by a prismatic joint.

Figure 1.3: Examples of articulated objects: refrigerator, microwave, drawers and human body.

recent years, with the advent of robust feature detectors and descriptors [56], along with the mathematical machinery to process multiple views [40]. We are now at a point where accurate point cloud reconstructions can automatically be made of textured, static scenes from a collection of semi-calibrated photographs [4, 35, 36, 17]. One limitation of traditional reconstruction approaches is that they assume a static scene, enabling them to exploit the redundancy available from multiple views when objects do not move between photographs. To overcome this limitation, several researchers [94, 108] have addressed the problem of *non-rigid* structure from motion by modeling the scene using a small number of basis shapes. Such approaches work well for objects that change shape in limited ways, but are inapplicable to objects with large changes in geometry.

Attention has been paid to reconstructing *articulated* objects from multiple

images [110, 80, 48, 90, 67] recently. Approaches to recovering articulated objects have focused primarily on either human pose recovery from a known skeletal model or estimation of joint positions from video. These approaches generally do not take full advantage of multi-view geometry, relying instead upon a known model or affine projection, and therefore do not reconstruct the surface of the articulated object in 3D. Moreover current approaches to articulated object reconstruction are limited to a single view. By tracking feature points throughout a video sequence, clustering the feature points, enforcing noise-robust models, and triangulating the rays, the 3D coordinates of the features points, as well as the parameters of the joint axes, can be recovered using any of several techniques. Such approaches, however, do not yield any information about the back side of the object that is not visible in the current view. In situations in which the robot wishes to manipulate or interact with such non-visible portions of the object, a single-view model is not sufficient.

We introduce the term *occlusion aware* to refer to the robot's knowledge of parts of the object that are not visible in the current view. This novel way of approaching the problem is motivated by recent developments in the structure from motion community, which has developed fully automated methods capable of reconstructing complete 3D models from a collection of images [86, 35, 32, 100]. That is, such methods reconstruct the 3D locations of points on all sides of the object, using only images from one or several cameras. Such knowledge has always been assumed in the context of grasping research based on 3D CAD models [6, 51, 65]. However, in a scenario in which the robot is interactively learning about the unknown objects in the scene, such models are not available; a new approach is needed.

In this work, we first present an occlusion-aware system for reconstructing articulated objects from images taken by a camera from different viewpoints. The proposed method, called Procrustes-Lo-RANSAC, or PLR, first builds two complete

3D point cloud models by applying structure-from-motion algorithms to images captured of the object in two different configurations. Then the method uses Procrustes analysis combined with a locally optimized RANSAC sampling strategy to automatically segment the points into the individual links. After aligning the links, the articulated structure of the object is estimated using a geometric approach. Second, with hand-eye calibration, the robot can align its coordinate system with that of the recovered articulated model and then manipulate the object by exercising the degrees of freedom captured by the model. The proposed approach, based on our earlier work in [43], does not have the limitations of previous systems, in that it uses perspective projection and does not make any planar assumptions about the scene. We show the results of the system on a variety of everyday objects, demonstrating the effectiveness of the approach. Third, a RGBD sensor, Microsoft Kinect, is introduced to improve the proposed approach by reconstructing high quality 3D articulated models using KinectFusion algorithm and the geometric approach. With such improvements, the system yields much denser models and increases the computation efficiency.

1.3 Outline of dissertation

The main goal of the work is to develop algorithms using multiple views to recover complete 3D models of articulated objects in domestic environments and thereby enable a robotic system to manipulate objects. The dissertation is organized in the following manner. Chapter 1 is the introduction of this work. Following the introduction a summary of the related work in three areas: multi-view reconstruction, articulated structure and object manipulation is described in Chapter 2. Chapter 3 presents the details of the proposed novel Procrustes-Lo-RANSAC (PLR) algorithm and demonstrates its performance for a variety of everyday objects. Once the algo-

rithm is addressed, its applications to a robotic system and the experimental results are described in Chapter 4. Chapter 5 begins by describing how to further improve the previous proposed approach of articulated objects reconstruction in Chapter 3, then addresses one possible solution which use a RGBD sensor (Microsoft Kinect) to recover high quality 3D articulated models. Finally conclusions, contributions of this work and some potential directions for future work are presented in Chapter 6.

Chapter 2

Related Work

Reconstruction and manipulation of articulated objects has become an active area in the computer vision and robotics community in recent years. There are a number of techniques and progress described in the literature. In the following sections, we present the related work from three aspects: multi-view reconstruction, articulated structure and object manipulation.

2.1 Multi-view reconstruction

In generally, multi-view reconstruction techniques use a sequence of images of an object or a scene from different viewpoints to recover its 3D structure [77]. Recently, many methods have been proposed by researchers in the computer vision and robotics community to handle various types of datasets either single or clustered objects, static or dynamic objects, and indoor or outdoor scenes. Most existing approaches can be categorized into the following two classes in terms of the scene representation.

2.1.1 Point cloud-based approaches

Structure from motion (SFM), in which the 3D point cloud of a scene is estimated by backprojecting corresponding points from multiple images into space, is a classic problem in computer vision. One approach is to exploit the so-called *rank constraint* to effectively factorize a matrix containing feature coordinates into matrices containing the shape of the scene and motion of the camera [92]. This factorization method was later extended to handle not only orthographic cameras but also paraperspective [71] projection and multiple bodies [16]. This latter work, by discovering the block diagonal structure of the measurement matrix in order to segment and reconstruct the geometry of multiple objects, is closely related to this work in terms of its overall goal.

More recent approaches to structure from motion have abandoned the batch processing approach of factorization in favor of a pipeline in which pairs of images are matched sequentially in order to build the 3D reconstruction. In one of the first approaches to exploit the impressive amount of data available in Community Photo Collections (CPCs), Snavely *et al.*[86] combine feature correspondences and an optimization routine to recover the 3D positions of the features along with the camera parameters. Goesele *et al.*[36] describe an approach which takes as input sparse 3D points from an SFM algorithm (such as the previous) and iteratively grows surfaces in order to reconstruct the geometry of the scene. Agarwal *et al.*[4] expanded these previous systems to handle a million images using a parallel distributed matching approach and bundle adjustment improvements aimed at minimizing equations with large numbers of variables. The approaches of Brown and Lowe [11], Sinha and Pollefeys [81], and Furukawa and Ponce [35] are focused on similar problems with more limited datasets. All of this work has been concentrated on static scenes, with

moving objects (such as tourists in the photos) considered noise to be removed.

A series of papers by Bregler and colleagues addressed the problem of non-rigid scene reconstruction. In their early work, Bregler *et al.*[9] showed that 3D reconstruction of non-rigid objects could be performed by modeling the object using a set of basis shapes. In follow-up work, Torresani *et al.*[95] incorporated feature tracking into the algorithm, so that the resulting system simultaneously solves for feature tracks, camera pose, and 3D non-rigid structure. Torresani and Bregler [93] then were able to apply this concept of basis shapes to derive a space-time rank constraint that results in more robust feature tracking when objects are non-rigid. In [94], the authors improve upon the earlier reconstruction algorithm by introducing learned shape priors to overcome ambiguities inherent in the original formulation. An alternative approach is proposed by Xiao *et al.*[108] who augmented the rotation constraints of the previous methods with basis constraints to uniquely determine the shape bases. Other related work is that of [70], who used a known model of a non-rigid object not to reconstruct the geometry but rather to detect the object and register it with the image.

2.1.2 Volume-based approaches

Similar to the pixel, the voxel is a volumetric method to represent visual scene in three dimensional world. The earliest approach based on volumetric representation to reconstruct 3D structures of a scene is the visual hull [55, 57]. The visual hull of an object is formed by intersecting projected silhouettes of the object from different views. The visual hull provides only the approximate shape of the object. Figure 2.1 shows a 2D example of the visual hull. Typically, the approach based on visual hull assumes that the foreground object in the collection of images is segmentable from

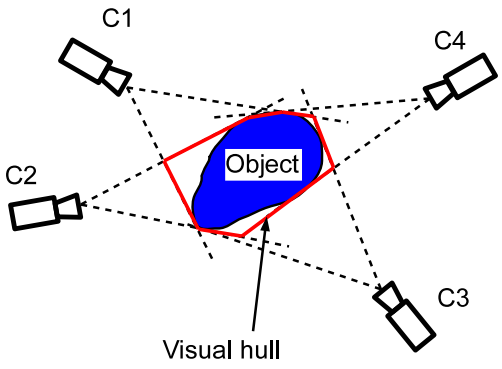

Figure 2.1: A 2D example of the visual hull: The visual hull of an object is formed by intersecting projected silhouettes of the object from different views.

the background. Once the visual hull is extracted from multiple objects' views, many methods such as [73, 87] based on octree representation, [76, 15] using Hough-like voting schemes and so forth were proposed to refine and reconstruct the object's 3D model [82].

One of the classic approaches is voxel coloring [78] which computes color consistency between images by traversing a set of discretized voxels and yields an accurate texture and color model of the object by identifying voxels' locations. The approach usually assumes objects are nearly Lambertian. The volume containing the object (e.g. visual hull) is first divided into a grid of voxels which are all initialized to be opaque. Then the approach checks each voxel to see if it has a consistent color in the input images. Voxels with inconsistent color are removed by setting them to be transparent. The 3D shape of the object is formed by the remaining opaque voxels. Voxel coloring approaches use restricted camera locations such as putting all cameras on one side of the object [18] or adding the ordinal visibility constraint [78] to simplify the voxel visibility computation which has to be decided before checking color consistency.

The voxel coloring approach was later extended to handle arbitrary camera positions by generalized voxel coloring approach [18, 58, 79] and space carving approach [54, 10]. Unlike voxel coloring approach both approaches scan the voxels multiple times and check color consistency using updated visibility information. Due to arbitrary camera locations, projected image pixels of a voxel are not possible to be visible in all input images [83, 82]. During each carving, both approaches need to find set of images in which projected image pixels of the voxel are visible. Generalized voxel coloring approach uses all these images to compute color consistency so that no voxels with inconsistent color remain in the final model. However space carving approach only uses part of these images such that the final model may contain voxels with inconsistent color [18].

More recent approaches based on voxels use level-set or graph-cuts techniques to optimize the problem of reconstructing 3D object shape. Level-set based methods [26, 24, 72] formulate the shape of an object in the 3D space as a time-varying implicit function, then iteratively evolve the geometry of the object by deforming an initial set of surfaces, and finally recover the object's shape by solving the zero set of the function. The latter methods [102, 96, 101, 41, 53] express a 3D object as a discrete weighted graph which defines a cost function, and extract the shape of the object by finding the max-flow/min-cut solution of the graph.

Traditional approaches of reconstructing objects from multiple images is limited by assuming objects do not move between photographs or change shape in limited ways. Such approaches are inapplicable to articulated objects with large changes in geometry. Therefore a new approach is needed. We take full advantage of multi-view reconstruction technique to interactively learn the structure of the unknown articulated object in 3D instead of relying upon a known model.

2.2 Articulated structure

Several approaches to reconstructing articulated objects from a monocular video sequence have been proposed in recent years. Early work by Sinclair et al. [80] estimates joint axes by clustering tracked feature points, from which camera projection matrices are recovered by assuming that the scene consists of planar surfaces rotating about vertical axes. When the motion is parallel to the 2D image plane, Ross et al. [74] use a probabilistic graphical model to recover the skeletal kinematic structure of the articulated object, while Zhang et al. [111] describe an approach for axis estimation using twists and exponential maps.

One promising approach builds upon the success of the factorization method for affine reconstruction [16, 92, 94]. By adding articulation constraints to the formulation, the so-called *rank constraint* (which restricts the rank of the measurement matrix consisting of the coordinates of tracked feature points) is extended by Tresadern and Reid [97] to detect the articulated objects, determine their degrees of freedom, and locate the joints. Using an iterative factorization approach, Paladini et al. [64] recover 3D shape and motion of non-rigid and articulated objects in the case of missing data. Yan and Pollefeys [109, 110] also investigate the subspace properties of articulated motion in a factorization framework by segmenting feature trajectories by local sampling and spectral clustering, then building the kinematic chain as a minimum spanning tree of a graph constructed from the segmented motion subspaces. More recent work by Fayad et al. [27] uses a hill-climbing approach that minimizes a single energy functional based on image reprojection error, with alternating steps utilizing graph cuts to assign points to links, then applies factorization to reconstruct 3D models of the links.

Other researchers focusing on human motion aim to recover the joint param-

eters of the human from video or motion capture [30, 38, 50, 63, 84]. Guan et al. [38] interactively recover the 3D shape and pose of a human from a single image using a previously learned model of the human body. Bălan et al. [12] optimize a search over body shape and pose, where the shape is represented as a mesh and fitted using a graphics model learned off-line from a dataset of detailed 3D range scans of people. Freifeld et al. [31] propose a computationally efficient 2D model of a person's contour to bridge the gap between 2D and 3D techniques in order to segment human bodies from images. Forsyth et al. [30] address the problem of tracking articulated objects, namely humans, in video. Ross et al. [74] model the relationships between feature point locations on articulated objects as stick figures with fixed lengths and connectivities. Other approaches to human skeletal tracking include [84, 50, 67].

Research that is most closely related to ours involves reconstructing articulated objects with unknown skeletal parameters. Sturm et al. [91] recover kinematic models of 1-DOF articulated objects such as microwave ovens by tracking the poses and orientations of rigid parts captured by the PhaseSpace motion capture system and addressing a mixture of parameterized and parameter-free (Gaussian process) representations to best explain the given observation. In related work, the same researchers [90] proposed an approach to learn articulation models of objects without using artificial markers. Rectangles in depth images obtained from a self-developed active stereo system are detected using a sampling-based approach. Then the robot uses generative models learned for the objects to estimate the type of articulation (revolute or prismatic). In contrast to their work, the proposed approach is not restricted to planar objects. Similar work by Katz et al. [48] reconstructs 3D kinematic structures of rigid articulated bodies in a single-view and sparse model based on feature tracking, motion segmentation and classical structure from motion techniques. In their latest work [46], Katz et al. segment, track and model articulated objects

with sufficient texture by using a RGBD sensor so that their approach can handle partial occlusions and small object motions.

2.3 Object manipulation

Using scene exploration with embedded sensors to reconstruct and manipulate a 3D model of unknown objects is an approach taken by several researchers [6, 52, 103, 75, 59]. Walck et al. [103] propose a method which automatically finds the position of the targeted object using a single eye-in-hand camera, captures multiple views of its shape using visual servoing, and models unknown objects using carved visual hull techniques. Bone et al. [6] model 3D objects by combining a silhouette model from a video camera with structured-light model from a laser projector. Klingbeil et al. [52] address the problem of opening new doors, avoiding the need to reconstruct 3D models by instead detecting door handles and extracting a small number of 3D features for alignment.

Surveying this literature, there remains a need in the robotics community to develop techniques to reconstruct 3D models of articulated objects, particularly models that incorporate the non-visible portions of the objects for occlusion-aware sensing and manipulation.

Chapter 3

Learning Articulated Objects

Figure 3.1 shows an overview of the system presented in this dissertation. First, a set of images is captured by a camera of the object from different viewpoints while the object remains stationary. Structure-from-motion techniques are used to the images to build a 3D model of the object. In order to learn the object's kinematic structure, the configuration of the object is interactively changed by exercising its degrees of freedom. Additional images are gathered of the object in the new configuration, and structure-from-motion yields a different 3D reconstruction. These two 3D models are segmented into the object's constituent components (rigid links) using the proposed Procrustes-Lo-RANSAC (PLR) method. A geometric approach utilizing an axis-angle representation is then used to estimate the axis of each joint. Based on these models, the robot with eye-in-hand can automatically compute the transformation between the object and robot coordinate systems, enabling it to manipulate the object around the articulation axis with a given grasp point, as shown in Figure 3.2.

We assume that the capability of performing sufficient exploratory interaction with the object to change its configuration is present. In this way, the approach bears some resemblance to interactive perception [45, 48, 105, 106, 107], except that we al-

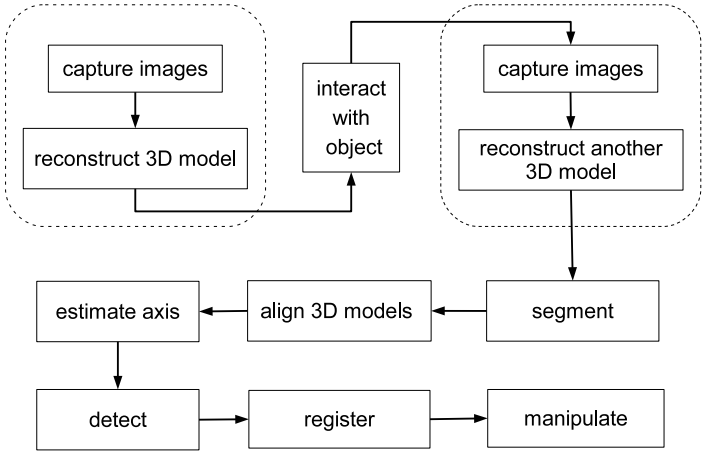

Figure 3.1: Overview of the system.

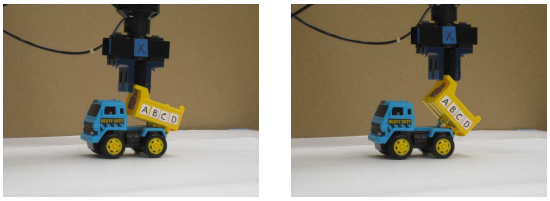

Figure 3.2: The PUMA 500 robotic arm manipulates a toy truck using the truck's kinematic model obtained by the occlusion-aware articulated reconstruction procedure.

low either a human or robot to perform the interaction due to the specific constraints of articulated motion in the objects. Automatically planning the end effector motion path for interactive perception in such situations remains an unsolved problem, because a preliminary model (at least) is needed in order to interact with the object, but the interaction is necessary to estimate the model. Therefore, having the user perform the interaction enables us to escape this difficult chicken-and-egg problem. As progress is made toward developing such autonomous exploratory behavior, the reconstruction method described in this paper still applies.

3.1 Building initial 3D model

We assume the object is a set of rigid links connected by revolute or prismatic joints, so that a *configuration* refers to a specific set of values for the joint angles or displacements. To reconstruct the 3D structure of an articulated object, we capture a set of images about the object from different camera viewpoints. This work does not require information about the camera location, orientation, or intrinsic parameters. Instead, the Bundler Structure from Motion (SfM) package [85, 86] is used to compute the camera parameters and projection matrices by matching key points. Bundler extracts focal length, image size, and other information from the EXIF tags of images, which is embedded by most consumer-level digital cameras. By assuming that the principal point is near the center of the image, Bundler then uses photo-consistency and the bundle adjustment algorithm to iteratively compute the desired parameters in order to minimize the reprojection error. Figure 3.3 shows the 3D models with camera locations of a toy truck in two different configurations by Bundler.

Once the cameras are calibrated, we apply the patch-based multi-view stereo (PMVS) algorithm [33, 34] to reconstruct dense 3D oriented points, where each point

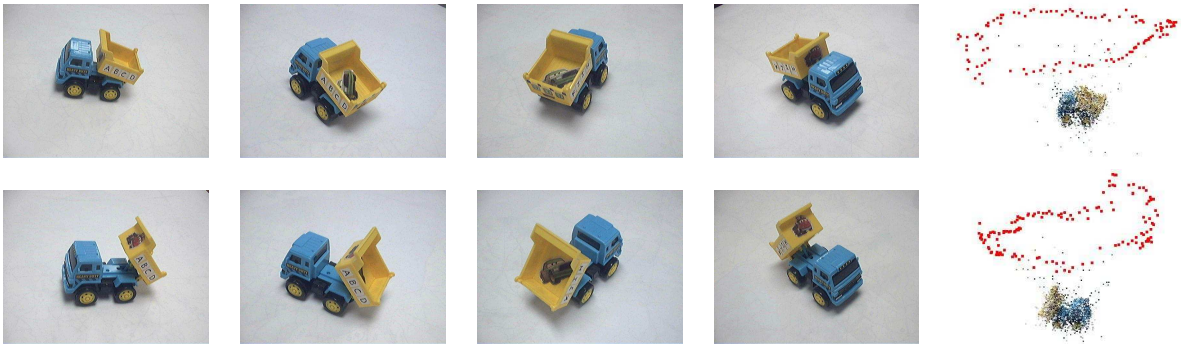

Figure 3.3: Top: Four images (out of 121 captured) of a toy truck, and the 3D model with all camera locations (red points) obtained by Bundler. Bottom: Four images (out of 147 captured) of the truck in a different configuration, along with the 3D model and camera positions (red points).

has an associated 3D location, surface normal, and a set of visible images. Taking calibrated images and camera parameters as inputs, PMVS begins with a sparse set of matched features and repeatedly expands the initial matches to nearby pixels, using visibility constraints to filter out false matches.

This procedure is then repeated to produce a second 3D model from another set of images obtained of the object in a different configuration in which all adjacent links have moved relative to each other. Note that only two configurations are needed, no matter how many links and joints. Figure 3.4 shows the dense 3D reconstruction of a toy truck in two different configurations. There is no constraint on the set of images, except that there must be sufficient overlap in the fields of view in order to facilitate feature matching across different views. In the experience, successive camera viewpoints should differ by no more than about 10 degrees, so that approximately 36 images are needed to capture an accurate 360-degree model; more images are needed to reconstruct the top or bottom of the object.

Figure 3.4: Top: Four images (out of 121 captured) of a toy truck, and the 3D reconstruction obtained. Bottom: Four images (out of 147 captured) of the truck in a different configuration, along with the 3D reconstruction.

3.2 Rigid link segmentation

Once the models have been constructed, the oriented 3D points of the models are segmented into the constituent rigid components of the object. We use the affine SIFT (ASIFT) feature detector [61], which is an affine invariant extension of the popular SIFT feature detector [56], to find features in every image of the two sets. For every feature point in an image of the first configuration, the matching feature point in the second configuration is found, which is defined as the one that minimizes the sum-of-squared differences (SSD) between gray-level patches surrounding the two features. These matched points are potential point correspondences. Then the same matching algorithm is run in the reverse order by swapping the roles of the images, and matches are retained if they agree in both directions. For each oriented 3D point, its closest ASIFT feature in each image is found by projected the point onto the image plane according to the determined camera parameters. Correspondence between oriented 3D points in the two models is thus established using the matching of these closest ASIFT features.

Given such correspondences, the Procrustes-Lo-RANSAC (PLR) algorithm

shown in Algorithm 1 is applied. The first part of this algorithm is motion segmentation shown in Algorithm 2. Procrustes analysis [25] is run iteratively in combination with a locally optimized RANSAC (Lo-RANSAC) sampling strategy [14] to find similarity transformations of rigid parts. Similarity transformations include rotation, translation, and scale, where the latter is needed because of the scale ambiguity in images.

The Procrustes algorithm is a classic method for aligning two point sets [25]. Let $X = [\,\mathbf{x}_1, \ldots, \mathbf{x}_n\,]$ and $Y = [\,\mathbf{y}_1, \ldots, \mathbf{y}_n\,]$ be two point sets, where each matrix has dimensions $d \times n$ for dimensionality d. (Normally $d = 3$, but for a 2D scene $d = 2$.) First we compute the centroid of each:

$$\mu_X = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \tag{3.1}$$

$$\mu_Y = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i. \tag{3.2}$$

Scale is handled by normalizing the coordinates using the Frobenius norm:

$$f_X = \sqrt{Tr(\bar{X}\bar{X}^T)}, \tag{3.3}$$

and similarly for f_Y, where Tr is the trace of the matrix and

$$\bar{X} = X - \mu_X \mathbf{1}_n^T. \tag{3.4}$$

The scaled, shifted coordinates

$$\hat{X} = (X - \mu_X \mathbf{1}_n^T)/f_X \tag{3.5}$$

$$\hat{Y} = (Y - \mu_Y \mathbf{1}_n^T)/f_Y, \tag{3.6}$$

where $\mathbf{1}_n$ is an n-element vector of all ones, are therefore centered at the origin with unit scale. The rotation between the point sets is then computed as

$$R = U\Sigma' V^T, \tag{3.7}$$

where the singular value decomposition (SVD) of $\hat{Y}\hat{X}^T$ is $U\Sigma V^T$. Instead of Σ, we use the matrix $\Sigma' = \mathrm{diag}(\,1, \quad 1, \quad \det(UV^T)\,)$ to ensure that $\det(R) = 1$, and therefore that R is a rotation matrix. Putting this all together yields

$$Y \approx \frac{f_Y}{f_X} R(X - \mu_X \mathbf{1}_n^T) + \mu_Y \mathbf{1}_n^T. \tag{3.8}$$

Given a point $\mathbf{x}_i$, then, the similarity transformation is given by $\sigma R\mathbf{x}_i + \mathbf{t}$, where $\sigma = f_Y/f_X$, and $\mathbf{t} = -\sigma\mu_X + \mu_Y$.

Algorithm 1 Procrustes-Lo-RANSAC (PLR) algorithm.

ProcrustesLoRansac($\bar{\mathbf{p}}, \bar{\mathbf{q}}$)
Input: Points $\bar{\mathbf{p}} = (\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(m)})$ in first configuration,
Points $\bar{\mathbf{q}} = (\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(m)})$ in second configuration,
where $\mathbf{p}^{(i)} \leftrightarrow \mathbf{q}^{(i)}$, $i = 1, \ldots, m$ are corresponding points
Output: Link labels $\bar{\lambda} = (\lambda^{(1)}, \ldots, \lambda^{(m)})$ for each point
Joint parameters $\{\mathbf{u}_{ab}, \omega_{ab}\}$ for adjacent links a and b

1 motSeg() //motion segmentation (see Algorithm 2)
2 FJP() //find joint parameters (see Algorithm 3)

Within the Lo-RANSAC framework, randomly selected triplets of correspondences are used to compute the transformation using Procrustes. The resulting transformation is used to align the cloud of points, and an alignment error is computed. This process is repeated with new random triplets until the error is smaller than some threshold, at which point all points which transform to coordinates close to

Algorithm 2 Motion segmentation algorithm.

```
motSeg()
 1  for i ← 1 to m do
 2        λ^(i) ← None
 3  link ← 0
 4  iter ← 0
 5  while iter < max-iter do
 6        i, j, k ← Rand3WithoutReplacement(1, m)
 7        R, t, σ ← Procrustes(p̄, q̄, {i, j, k})
 8        num-inliers ← 0
 9        for i ← 1 to m do
10              q^(i)' ← Transform(p^(i), R, t, σ)
11              if ‖q^(i) − q^(i)'‖ < τ And λ^(i) == None then
12                    λ^(i) ← Temp
13                    num-inliers ← num-inliers +1
14        if num-inliers > min-num-inliers then
15              new-label ← link
16              link ← link +1
17              iter ← 0
18        else
19              new-label ← None
20              iter ← iter +1
21        for i ← 1 to m do
22              if λ^(i) == Temp then
23                    λ^(i) ← new-label
```

Algorithm 3 Find joint parameters algorithm.

FJP()

1 **for** each link a **do**
2 $\quad b \leftarrow \textsc{FindClosestLink}(a)$
3 $\quad \mathcal{L}_a \leftarrow \{i : \lambda^{(i)} = a\}$
4 $\quad \mathcal{L}_b \leftarrow \{i : \lambda^{(i)} = b\}$
5 $\quad {}^A_B R_a,\ {}^A_B \mathbf{t}_a,\ {}^A_B \sigma_a \leftarrow \textsc{Procrustes}(\bar{\mathbf{p}},\ \bar{\mathbf{q}},\ \mathcal{L}_a)$
6 $\quad \bar{\mathbf{q}}' \leftarrow \textsc{Transform}(\bar{\mathbf{q}},\ {}^A_B R_a,\ {}^A_B \mathbf{t}_a,\ {}^A_B \sigma_a)$
7 $\quad {}^A_A R_b,\ {}^A_A \mathbf{t}_b,\ {}^A_A \sigma_b \leftarrow \textsc{Procrustes}(\bar{\mathbf{p}},\ \bar{\mathbf{q}}',\ \mathcal{L}_b)$
8 $\quad$ Given ${}^A_A R_b$ in (3.18), compute $\mathbf{u}_{ab}$ and θ from (3.19) and (3.20)
9 $\quad$ Compute ${}^{A^\pi}_A R$ from $\mathbf{u}_{ab}$ using (3.26)
10 $\quad$ Given θ, construct $\ddot{R}$ as in (3.29)
11 $\quad$ Compute $\bar{\mathbf{t}} = {}^{A^\pi}_A R\, {}^A_A \mathbf{t}_b$
12 $\quad$ Extract $\ddot{\mathbf{t}}$ and t_z from $\bar{\mathbf{t}}$, as in (3.31)
13 $\quad$ Compute $\ddot{\omega}$ from $\ddot{R}$ and $\ddot{\mathbf{t}}$ using (3.33)
14 $\quad$ Compute ω_{ab} from ${}^{A^\pi}_A R$, $\ddot{\omega}$, and t_z using (3.34)

their match are segmented from the rest. Once a link has been found, points on the link are removed, and the entire process is repeated until no transformation can be found, i.e., there are no more rigid parts. In the code, the labels $\bar{\lambda} = (\lambda^{(1)}, \ldots, \lambda^{(m)})$ are such that $\lambda^{(i)}$ indicates the link for $\mathbf{p}^{(i)}$ and $\mathbf{q}^{(i)}$, or NONE if $\mathbf{p}^{(i)}$ and $\mathbf{q}^{(i)}$ are not on any link. For each pair of adjacent links a and b, the free vector $\mathbf{u}_{ab} \in \mathbb{R}^3$ specifies the direction of the joint axis, and $\omega_{ab} \in \mathbb{R}^3$ is a point on the joint axis, specifically the average projection of all points on both links onto the axis.

3.3 Classifying joints

We assume that two rigid links are connected by either a revolute joint or a prismatic joint. The type of joint is automatically determined by examining the similarity transformation R, $\mathbf{t}$, and σ between the links determined by Procrustes alignment, where R is the rotation matrix, $\mathbf{t}$ is the translation vector, and σ is the

relative scaling between the two models. Although one might be inclined to use the translation vector $\mathbf{t}$ to distinguish between the two types of joints, it is important to note that $\mathbf{t}$ will not in general be zero for a revolute joint. This is because the axis of the coordinate system attached to the link does not necessarily (and usually will not) align with the axis of rotation. In other words, although we are interested in rotation about the axis, Procrustes computes the rotation about the origin of the coordinate system, which is somewhat arbitrarily determined by structure-from-motion. While these rotations themselves are identical, a non-zero translation $\mathbf{t}$ is needed to compensate for the misalignment. As a result, we instead determine the type of joint automatically by examining the rotation matrix R: If R is close to the identity matrix, then the joint is determined to be a prismatic joint; otherwise it is a revolute joint. This procedure is repeated for each pair of adjacent links.

3.4 Finding joint axes

We now describe the second part of the PLR method shown in Algorithm 1. For both revolute and prismatic joints, an axis is a ray in 3D space about or along which the movement occurs. Locating the axis of a prismatic joint is straightforward: The unit vector $\mathbf{t}/\|\mathbf{t}\|$ yields the direction of motion along the prismatic joint, while the mean of the points on the second link is used as a point on the axis. Revolute joints are more complicated.

3.4.1 Two links in 2D with revolute joint

To simplify the problem of estimating the revolute joint parameters, let us begin with the restricted case of an object consisting of just two links in 2D ($d = 2$). Let ${}^{A}\mathcal{P}$ be the set of points on the object in the first configuration, and let ${}^{B}\mathcal{Q}$ be

the set of points on the object in the second configuration. The leading superscript indicates the coordinate frame, either $\{A\}$ or $\{B\}$. The two coordinate frames differ not only by a Euclidean transformation, but also by an unknown scale, since the points were acquired by images from a camera.

Let us assume that the first point set has been segmented according to the two links, called Link 0 and Link 1. This yields ${}^{A}\mathcal{P} = {}^{A}\mathcal{P}_0 \cup {}^{A}\mathcal{P}_1$, where ${}^{A}\mathcal{P}_i, i \in \{0, 1\}$ is the point set for the ith link in the first configuration. Similarly, for the second configuration we have ${}^{B}\mathcal{Q} = {}^{B}\mathcal{Q}_0 \cup {}^{B}\mathcal{Q}_1$. See Figure 3.5 for an illustration.

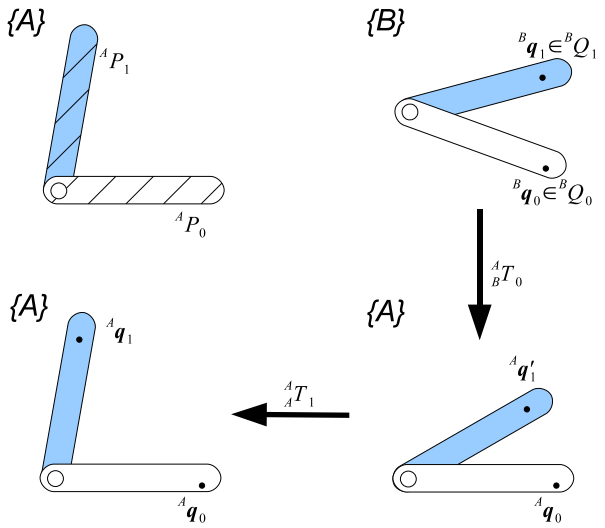

Figure 3.5: Aligning an object with two links and two configurations in 2D. Clockwise from top left: The first configuration consists of two point sets for the two links: ${}^{A}\mathcal{P}_0$ and ${}^{A}\mathcal{P}_1$; the second configuration consists of two different point sets in a different coordinate frame: ${}^{B}\mathcal{Q}_0$ and ${}^{B}\mathcal{Q}_1$; applying the transformation ${}^{A}_{B}T_0$ to all the points in the second configuration aligns Link 0 with the first configuration; applying the transformation ${}^{A}_{A}T_1$ to only the points in Link 1 aligns those points as well.

We assume that Link 0 is the base, or reference, link. Therefore, the first step is to align the points in this link between the two coordinate frames. Let

$$ {}^{A}_{B}T_0 = \begin{bmatrix} {}^{A}_{B}\sigma_0 \, {}^{A}_{B}R_0 & {}^{A}_{B}\mathbf{t}_0 \\ \mathbf{0}_d^T & 1 \end{bmatrix} \tag{3.9} $$

be the similarity transformation from coordinate frame $\{B\}$ to $\{A\}$ to align Link 0, where ${}^{A}_{B}R_0$ is the $d \times d$ rotation matrix, ${}^{A}_{B}\mathbf{t}_0$ is the $d \times 1$ translation vector, and $\mathbf{0}_d^T$ is the transpose of a $d \times 1$ vector of all zeros. Let ${}^{B}\mathbf{q}_0 \in {}^{B}\mathcal{Q}_0 \subset \mathbb{R}^d$ be a point in the second configuration of Link 0, expressed in the second coordinate frame $\{B\}$. The transformation above can be used to express the same point in the first coordinate frame, $\{A\}$:

$$ {}^{A}\mathbf{q}_0 = {}^{A}_{B}\sigma_0 \, {}^{A}_{B}R_0 \, {}^{B}\mathbf{q}_0 + {}^{A}_{B}\mathbf{t}_0, \tag{3.10} $$

or equivalently

$$ {}^{A}\tilde{\mathbf{q}}_0 = {}^{A}_{B}T_0 \, {}^{B}\tilde{\mathbf{q}}_0, \tag{3.11} $$

where ${}^{B}\tilde{\mathbf{q}}_0 = [\, ({}^{B}\mathbf{q}_0)^T \quad 1\,]^T$ are the homogeneous coordinates of ${}^{B}\mathbf{q}_0$, and similarly for ${}^{A}\tilde{\mathbf{q}}_0$.

It would be possible to align Link 1 in a similar manner, leading to ${}^{A}\tilde{\mathbf{q}}_1 = {}^{A}_{B}T_1 \, {}^{B}\tilde{\mathbf{q}}_1$, where ${}^{B}\mathbf{q}_1 \in {}^{B}\mathcal{Q}_1 \subset \mathbb{R}^d$ is a point in the second configuration of Link 1, expressed in the second coordinate frame, $\{B\}$. However, by first aligning the base link (Link 0), the Procrustes algorithm begins closer to the true alignment, thereby leading to more robust convergence. Therefore, we first apply the transformation ${}^{A}_{B}T_0$ to all of the points to yield "almost aligned" coordinates for the other link:

$$ {}^{A}\tilde{\mathbf{q}}'_1 = {}^{A}_{B}T_0 \, {}^{B}\tilde{\mathbf{q}}_1, \tag{3.12} $$

where the prime indicates that the points in Link 1 are not aligned. In other words, ${}^{A}\tilde{\mathbf{q}}'_1$ is a point from Link 1 of the second configuration, expressed in Frame $\{A\}$ according to the transformation obtained by Link 0. After this step of "almost aligning" the points, we again use the Procrustes analysis method combined with the Lo-RANSAC

sampling strategy to find the transformation ${}^{A}_{A}T_1$ so that

$$ {}^{A}\tilde{\mathbf{q}}_1 = {}^{A}_{A}T_1 \, {}^{A}\tilde{\mathbf{q}}'_1 \tag{3.13} $$

specifies the coordinates of Link 1 of the second configuration aligned with Link 1 of the first configuration, expressed in Frame $\{A\}$. Here the scaling factor should be equal to 1, so it can safely be ignored. In non-homogeneous coordinates, we have

$$ {}^{A}\mathbf{q}_1 = {}^{A}_{A}R_1 \, {}^{A}\mathbf{q}'_1 + {}^{A}_{A}\mathbf{t}_1. \tag{3.14} $$

3.4.2 Finding the 2D axis of rotation

It is a simple matter to show that any Euclidean transformation (rotation plus translation) can be rewritten as a rotation applied to translated points:

$$ \begin{aligned} {}^{A}\mathbf{q}_1 &= {}^{A}_{A}R_1 \, {}^{A}\mathbf{q}'_1 + {}^{A}_{A}\mathbf{t}_1 && (3.15) \\ &= {}^{A}_{A}R_1 \left({}^{A}\mathbf{q}'_1 - \omega \right) + \omega, && (3.16) \end{aligned} $$

where

$$ \omega = \left(I_d - {}^{A}_{A}R_1 \right)^{-1} {}^{A}_{A}\mathbf{t}_1, \tag{3.17} $$

and I_d is the $d \times d$ identity matrix. This equivalency is illustrated in Figure 3.6. Instead of rotating the point, then translating, the alternate formulation involves shifting the origin of the coordinate system, applying the rotation, then shifting the origin back. The point $\omega \in \mathbb{R}^d$ is the temporary origin about which the rotation is applied. Therefore, in 2D ω specifies the axis of rotation in the sense that ω is the point about which the rotation occurs. (In 3D more information is needed, as

explained below.)

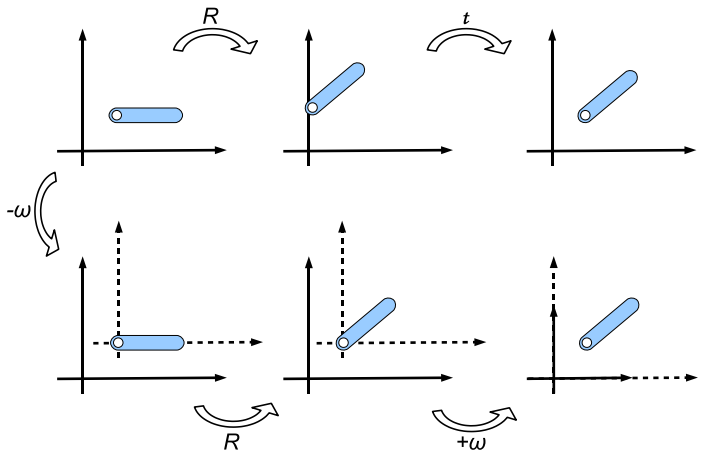

Figure 3.6: An illustration of the axis of rotation in 2D. TOP: Using (3.15), a Euclidean transformation involves applying a rotation about the origin, followed by a translation. BOTTOM: Using the equivalent expression in (3.16), the transformation involves shifting the origin of the coordinate system, then applying the rotation about the temporary origin, then shifting the origin back.

To verify that this equation makes sense, note that in (3.17), the axis of rotation ω is undefined if ${}^A_A R_1 = I_d$, i.e., if there is no rotation. Also note that $\|{}^A_A \mathbf{t}_1\| \leq 2\|\omega\|$, which can be seen geometrically because translating by $\|\omega\|$, then rotating, then translating again by $\|\omega\|$ cannot cause a translation of more than $2\|\omega\|$ as shown in Figure 3.7. Specially, there is no translation when the rotation angle is zero; the translation $\|{}^A_A \mathbf{t}_1\| = \sqrt{2}\|\omega\|$ when the rotation angle is 90 degrees; and the translation $\|{}^A_A \mathbf{t}_1\| = 2\|\omega\|$ when the rotation angle is 180 degrees. Algebraically, the result follows from the fact that for any $n \times n$ unitary matrices U and V (note that rotation matrices are unitary), $\|U - V\| \leq \max\{|\lambda_U - \lambda_V| : \lambda_U \in L_U, \lambda_V \in L_V\}$ if the right hand side $< \sqrt{2}$, otherwise $\|U - V\| \leq 2$, where L_U is the set of eigenvalues of U, and L_V is the set of eigenvalues of V [13]. Also the eigenvalues of I_2 are 1 and 1, i.e., $(1, 0)$ and $(1, 0)$ in the complex plane, and the eigenvalues of a 2×2 rotation matrix are complex conjugates lying on the unit circle of the complex plane.

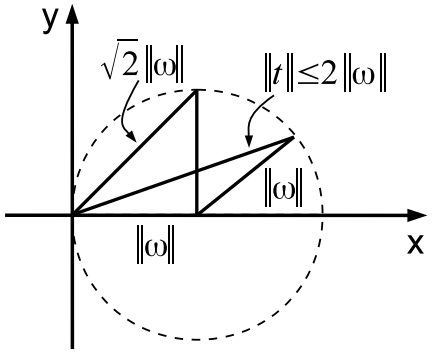

Figure 3.7: An illustration of the translation in 2D: $\|t\| \leq 2\|\omega\|$. Special cases: no translation if $\theta = 0$; $\|t\| = \sqrt{2}\|\omega\|$ if $\theta = 90°$; and $\|t\| = 2\|\omega\|$ if $\theta = 180°$ where θ is the rotation angle.

3.4.3 Extending to 3D

Now we shall extend the two-link case to 3D by letting $d = 3$. As before, we use Procrustes analysis combined with Lo-RANSAC to find the Euclidean transformation ${}^{A}_{B}T_0$ (now a 3×3 matrix) aligning Link 0 in the second configuration to the same link in the first configuration. Then the technique is applied again to find the transformation ${}^{A}_{A}T_1$ to align Link 1 in the second configuration, where the rotation matrix can be written as

$$ {}^{A}_{A}R_1 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{3.18} $$

This rotation matrix can be parameterized using the axis-angle representation as a unit vector $\mathbf{u} \in \mathbb{R}^3$ indicating the direction of a free vector parallel to the axis of rotation, and an angle θ describing the magnitude of the rotation about the axis in the right-hand sense. Similarly, the translation ${}^{A}_{A}\mathbf{t}_1$ is parameterized by computing an arbitrary point $\omega \in \mathbb{R}^3$ on the axis. The rotation angle $\pi \leq \theta \leq 0$ can be computed

from the matrix by the simple formula

$$\theta = \arccos\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right), \tag{3.19}$$

where the constraint $\pi \leq \theta \leq 0$ arises from the ambiguity that a rotation of θ about $\mathbf{u}$ is equivalent to a rotation of $-\theta$ about $-\mathbf{u}$.

To find the axis $\mathbf{u}$, we note that the eigenvalues of a 3×3 rotation matrix are 1 and $\cos\theta \pm i \sin\theta$, where $i = \sqrt{-1}$. Since any vector $\mathbf{u}$ parallel to the rotation axis must remain unchanged by the rotation, the vector must satisfy ${}^{A}_{A}R_1\mathbf{u} = \mathbf{u}$. Therefore, from the definition of eigenvalues and eigenvectors, the axis is the eigenvector corresponding to the eigenvalue $\lambda = 1$. One way to estimate $\mathbf{u}$, then, is to compute the eigenvalues and eigenvectors of ${}^{A}_{A}R_1$ and to retain the eigenvector associated with the eigenvalue of 1. If there is no rotation, i.e., ${}^{A}_{A}R_1 = I_3$, then all three eigenvalues are 1, and the rotation axis is undefined.

An alternate, simpler formula for computing the axis of rotation is the following:

$$\mathbf{u} = \frac{\hat{\mathbf{u}}}{\|\hat{\mathbf{u}}\|}, \qquad \text{where} \quad \hat{\mathbf{u}} = \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}, \tag{3.20}$$

and $\|\cdot\|$ is the L_2 norm. Note that this formula not only does not work when $\theta = 0$ (in which case the axis is undefined) but also when $\theta = \pi$ (in which case the formula yields an unhelpful $\hat{\mathbf{u}} = [\,0 \quad 0 \quad 0\,]^T$).

Once the axis of rotation $\mathbf{u}$ has been found, the 3D rotation about this axis can be thought of as a 2D rotation in the plane Π_u perpendicular to $\mathbf{u}$. Figure 3.8 shows the vector $\mathbf{u}$ in the current coordinate frame $\{A\}$. Let us define a new coordinate frame $\{A^\pi\}$ such that the z^π axis is aligned with $\mathbf{u}$. Therefore, the plane Π_u is the

same as the $x^\pi y^\pi$ plane in $\{A^\pi\}$. To transform from $\{A\}$ to $\{A^\pi\}$, we rotate about the y axis by α, then about the original x axis by β. By geometry (see the figure), these angles are given by

$$\cos\alpha = u_z/\eta \tag{3.21}$$

$$\sin\alpha = u_x/\eta \tag{3.22}$$

$$\cos\beta = \eta \tag{3.23}$$

$$\sin\beta = u_y, \tag{3.24}$$

where $\eta = \sqrt{u_x^2 + u_z^2}$ and $\mathbf{u} = [\,u_x \quad u_y \quad u_z\,]^T$. Since the rotation axes are fixed, the matrices are composed in right-to-left order, yielding an overall transformation of

$$\begin{aligned} {}^{A^\pi}_{A}R &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\beta & -s_\beta \\ 0 & s_\beta & c_\beta \end{bmatrix} \begin{bmatrix} c_\alpha & 0 & s_\alpha \\ 0 & 1 & 0 \\ -s_\alpha & 0 & c_\alpha \end{bmatrix} && (3.25) \\ &= \begin{bmatrix} u_z/\eta & 0 & u_x/\eta \\ u_x u_y/\eta & \eta & -u_y u_z/\eta \\ -u_x & u_y & u_z \end{bmatrix}, && (3.26) \end{aligned}$$

where $c_\alpha = \cos\alpha$, $s_\alpha = \sin\alpha$, and similarly for c_β and s_β.

Now that we have found ${}^{A^\pi}_{A}R$, we can apply this rotation matrix to the data to align the x and y axes with the $x^\pi y^\pi$ plane, then rotate about the z axis by θ, then rotate back:

$${}^{A}_{A}R_1 = {}^{A^\pi}_{A}R^T \underbrace{\begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_z^\theta} {}^{A^\pi}_{A}R \tag{3.27}$$

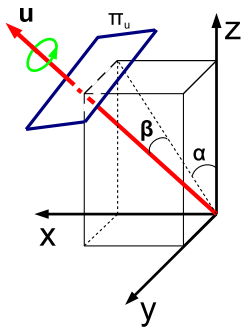

Figure 3.8: The axis of rotation $\mathbf{u}$ is parameterized by angles α and β, and the plane Π_u is perpendicular to $\mathbf{u}$.

Including translation, we have

$$\begin{aligned} {}^{A}\mathbf{q}_1 &= {}^{A}_{A}R_1\,{}^{A}\mathbf{q}'_1 + {}^{A}_{A}\mathbf{t}_1 \\ &= {}^{A^\pi}_{A}R^T\,R^\theta_z\,{}^{A^\pi}_{A}R\;{}^{A}\mathbf{q}'_1 + {}^{A}_{A}\mathbf{t}_1. \end{aligned}$$

As we noted before, we can rewrite this application of rotation followed by translation as a shift of the origin, followed by a rotation, followed by a shift back:

$${}^{A}\mathbf{q}_1 = {}^{A^\pi}_{A}R^T\,R^\theta_z\,{}^{A^\pi}_{A}R\left({}^{A}\mathbf{q}'_1 - \omega\right) + \omega,$$

where

$$\omega = \left(I_d - {}^{A^\pi}_{A}R^T\,R^\theta_z\,{}^{A^\pi}_{A}R\right)^{-1}\,{}^{A}_{A}\mathbf{t}_1, \tag{3.28}$$

where I_d is the $d \times d$ identity matrix. As before, if ${}^{A^\pi}_{A}R$ is the identity matrix (no rotation), then the point ω about which we are rotating is undefined because $I_d - R^\theta_z$ is singular. The point ω is the unique point in 3D where the appropriate 3D rotation about it aligns ${}^{A}\mathbf{q}'_1$ with ${}^{A}\mathbf{q}_1$.

Although (3.28) should work, we have found better results are obtained by an alternate approach in which we use a 3D rotation to align the x-y plane with Π_u (or

equivalently, the z axis with $\mathbf{u}$), then apply the 2D formula in (3.17) to compute ω, then rotate back. Let us define $\ddot{R}$ as the upper 2×2 part of R_z^θ:

$$\ddot{R} = \begin{bmatrix} c_\theta & -s_\theta \\ s_\theta & c_\theta \end{bmatrix}. \tag{3.29}$$

Define $\ddot{\mathbf{q}}$ as the first two elements of ${}^{A^\pi}\mathbf{q}'_1 = {}^{A^\pi}_{A}R\,{}^{A}\mathbf{q}'_1$, and q_z as the third element, so that

$${}^{A^\pi}\mathbf{q}'_1 = \begin{bmatrix} \ddot{\mathbf{q}} \\ q_z \end{bmatrix}. \tag{3.30}$$

Define $\ddot{\mathbf{t}}$ as the first two elements of $\bar{\mathbf{t}} = {}^{A^\pi}_{A}R\,{}^{A}_{A}\mathbf{t}_1$, and t_z as the third element, so that

$$\bar{\mathbf{t}} = \begin{bmatrix} \ddot{\mathbf{t}} \\ t_z \end{bmatrix}. \tag{3.31}$$

Now we have

$${}^{A}\mathbf{q}_1 = {}^{A^\pi}_{A}R^T \left(\begin{bmatrix} \ddot{R}\,(\ddot{\mathbf{q}} - \ddot{\omega}) + \ddot{\omega} \\ q_z \end{bmatrix} + \begin{bmatrix} \mathbf{0}_2 \\ t_z \end{bmatrix} \right), \tag{3.32}$$

where $\mathbf{0}_2$ is a 2×1 vector of all zeros and

$$\ddot{\omega} = \left(I_2 - \ddot{R} \right)^{-1} \ddot{\mathbf{t}}, \tag{3.33}$$

where I_2 is the 2×2 identity matrix. Once $\ddot{\omega}$ is found, the final 3D point ω is given by

$$\omega = {}^{A^\pi}_{A}R^T \begin{bmatrix} \ddot{\omega} \\ t_z \end{bmatrix}. \tag{3.34}$$

3.5 Experimental results

The performance of the PLR algorithm was evaluated on a variety of different real-world objects, such as those that might be found in a home, office, or kitchen.

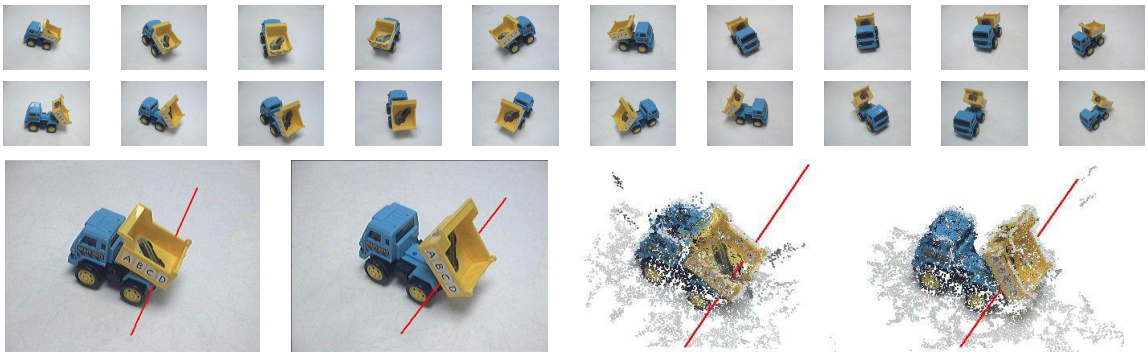

Figure 3.9: Axis estimation of the truck. The top two rows show ten images (out of 121) of the first configuration, and ten images (out of 147) of the second configuration. The last row shows the estimated axis (red line) overlaid on an image and 3D model from each configuration.

For the experiments, we used a Logitech Quickcam Pro 5000 for collecting images, mounted on a PUMA 500 robotic arm for manipulation.

We first demonstrate the PLR algorithm on the toy truck encountered earlier. Figure 3.9 shows some of the images used to reconstruct the two 3D models, along with the estimated axis overlaid on two of those images and on the 3D models. A total of 121 and 147 images, respectively, were needed to reconstruct the two models. Visually, the axis appears to be quite close to the true axis, indicating that the algorithm was able to accurately segment the links and estimate the position and orientation of the axis.

The next experiment involved refrigerators. Figure 3.10 shows a synthetic refrigerator created by the 3D modeling software known as Blender [1]. During the interaction, 16 images were captured for each configuration respectively. The first shows two images selected from the two sets overlaid with the axis of rotation (red line). The 3D reconstructions and estimated rotation axis of rotation are shown in the figure. Also we demonstrate the result on a real refrigerator, shown in Figure 3.11.

We also tested the approach on more practical items. Figure 3.11 shows the

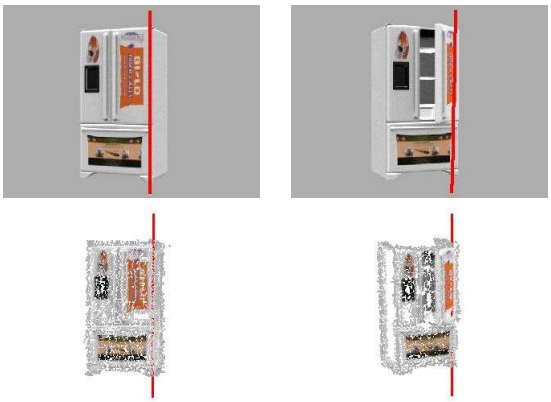

Figure 3.10: Two images of the synthetic refrigerator (out of 16 and 16, respectively) in two different configurations (top), and the 3D reconstructions with the estimated axis overlaid (red line).

results on five different objects, including a full-sized door, a cabinet, a microwave, a refrigerator, and a drawer. All objects contain a single revolute joint except the last, which contains a single prismatic joint.

Some results of the related work are shown in Figure 3.12. Yan and Pollefeys [109, 110] recover the kinematic chain of articulated objects such as a person dancing with his upper body (Figure 3.12(a)) based on factorization, but their approach yields sparse 3D models and only works for revolute joints. Ross et al. [74] formulate the structure of an articulated object as a probabilistic model and fit it via unsupervised learning. Figure 3.12(b) shows the articulated skeleton of a walking giraffe learned by the model. Their approach is sensitive to the initial segmentation, and it produces 2D models. Katz et al [48] track features of articulated objects as a manipulator interact with them such as a toy train shown in Figure 3.12(c). Then they recover the axes of links. Their approach is able to handle both prismatic and revolute joints, and produces sparse 3D models. Sturm et al. [91, 89] estimate the kinematic model of an articulated object based on the trajectory of the robot's end effector. Figure 3.12(d) shows a robot opens a dishwasher. Their approach does not yield

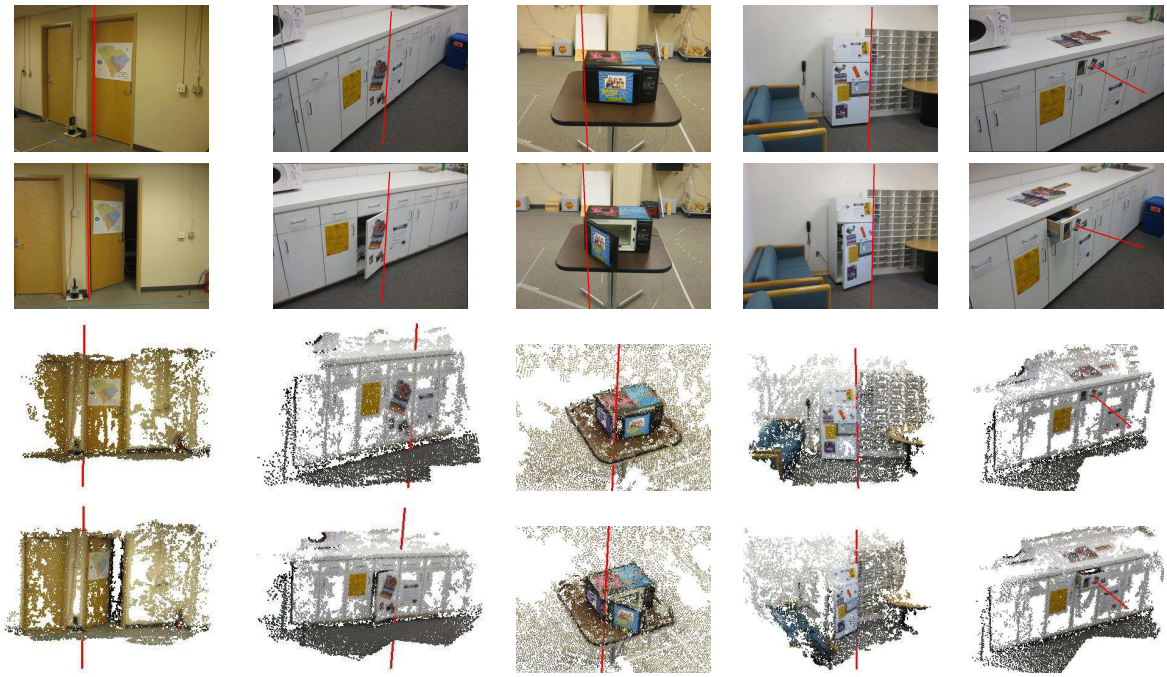

Figure 3.11: Five examples of articulated reconstruction. All objects contain a single revolute joint except the last, which contains a single prismatic joint. Each column shows two images of the object in two configurations and two 3D reconstructions with the estimated axis overlaid (red line). The images are arbitrarily selected from two sets of images used for 3D reconstructions. From left to right, the number of images used are 22/19 (door), 17/20 (cabinet), 99/94 (microwave), 24/25 (refrigerator), and 13/18 (drawer).

(a) (b) (c) (d) (e)

Figure 3.12: Results of related work. (a) The recovered kinematic chain of a person dancing with his upper body in Yan et al. [109, 110]. (b) The estimated articulated skeleton of a walking giraffe in Ross et al. [74]. (c) Tracking features of a toy train as a manipulator interacts with it in Katz et al. [48]. (d) A robot opens a dishwasher in Sturm et al. [91, 89]. (e) The recovered articulation model of a drawer in Sturm et al. [90].

3D models. Also Sturm et al. [90] use an active stereo camera to detect doors and drawers. Figure 3.12(e) shows the recovered articulation model of a drawer. Their approach is limited to handle planar objects.

Compared with these work, the proposed method in this work is able to reconstruct fairly dense 3D models, segment the points into the individual links, and accurately estimate the axis of rotation or translation. Also it supports both revolute and prismatic joints, and does not make any assumptions regarding planarity of the object. This method automatically classifies joints type and works for objects with multiple joints. The resulting models therefore include dense 3D point clouds representing the surfaces of the objects, along with the joint parameters.

To quantify the accuracy of the estimated axes, we computed the angle between the axis and the normal of a plane in the scene, where the plane was obtained by fitting plane parameters to points from the cloud corresponding to the real plane in the scene. For example, in the cases of the door, cabinet, and refrigerator, the floor plane was obtained, and the error was deemed to be the angle between the estimated axis and the normal of the floor. For the microwave, truck, and drawer, the same procedure was followed, except that a plane was fit to the table instead of the floor,

	PR		PLR		PLRI	
object	angle(°)	std	angle(°)	std	angle(°)	std
truck	1.0	0.4	0.9	0.0	0.8	0.0
door	2.8	1.7	1.1	0.2	1.1	0.2
cabinet	1.0	0.4	0.8	0.2	1.0	0.2
microwave	1.8	0.4	1.1	0.2	1.2	0.2
refrigerator	4.6	0.7	4.9	0.3	4.9	0.3
drawer	2.3	1.5	0.2	0.2	0.7	0.5
synthetic	1.0	0.4	0.9	0.0	0.8	0.0
total	14.5	5.5	**9.9**	**1.1**	10.5	1.4

Table 3.1: The average and standard deviation of the angular error of the estimated axis using the proposed algorithm (PLR), along with two others, namely Procrustes-RANSAC (PR) and Procrustes-Lo-RANSAC-ICP (PLRI).

and the error was subtracted from 90 degrees in the latter two cases, since the truck and drawer axes are parallel to the table. The results of this quantitative assessment are shown in Table 3.1. The last column of the table shows the results on a synthetic refrigerator created by a 3D drawing program. This table also shows, for comparison, two other versions of the system, one which uses RANSAC instead of Lo-RANSAC (called PR), and one which augments the proposed technique with iterative closest point (called PLRI). For each object, each algorithm was executed five times, and the average and standard deviation of the error are shown. Overall, the PLR algorithm outperforms the other two, both in terms of a lower average error and a lower standard deviation.

Figure 3.13 shows examples of objects with multiple joints. The PLR algorithm was able to reconstruct 3D models and estimate the multiple axes. For the dump truck, the angle between the estimated axes was 2.5 degrees, while the angle between the axes for the scraper truck was 7.6 degrees. However, it is difficult from these numbers to assess the accuracy of the system, since the cheap plastic construction of both trucks allows for considerable motion between the parts in all directions, so that

the axes are not perfectly parallel even in the real objects.

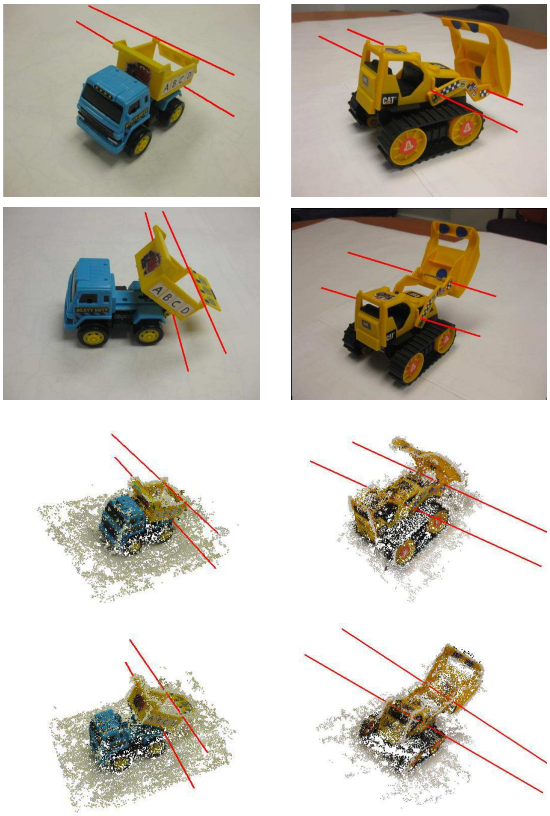

Figure 3.13: Articulated reconstruction of multiple axes for a toy dump truck and toy scraper truck. The display is similar to the previous figure. The number of images used for the two configurations are 125/119 (dump truck) and 111/134 (scraper truck).

Chapter 4

Manipulating Articulated Objects

Once the occlusion-aware 3D articulated model has been obtained, it can be used by a robot to manipulate the object. Given a particular point on the object, the robot can move its end effector to that position, even if the point is not visible in the current view. This is one of the main advantages of the occlusion-aware approach, namely, that the robot is not limited only to the side of the object that is currently visible, but rather that a full 3D model is available. Having grasped the object at a point, it can then use its knowledge of the articulation axis in order to move in such a way as to exercise the articulation.

In the robotic manipulation system, we use a PUMA 500 robotic arm and a handhold digital camera mounted on the robot end-effector hand. The first step for manipulation is to estimate the transformation between the object model and the robot coordinate frame. To make this a Euclidean transformation, we first must overcome the scale (σ) ambiguity. The scale of the object can be estimated in one of several ways. If the camera is attached to the robot during capture time, then the known positions of the end effector can be compared with the estimated camera positions to determine the overall scale of the scene. Alternatively, a separate step

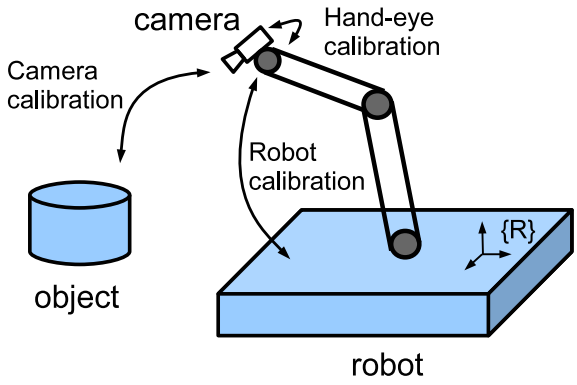

Figure 4.1: In order to locate an object with respect to the robot world base $\{R\}$, there requires three calibrations: camera calibration, hand-eye calibration and robot calibration.

can compute the projective distance from the camera to the table, which is then compared with the known height of the table. A third alternative is to simply use a known length on the object.

To locate an object with respect to the robot world base $\{R\}$, there requires three calibrations [99]: camera calibration which is to obtain the relative position and orientation between the object and the camera, hand-eye calibration which is to estimate the relative position and orientation between the camera and the robot hand, and robot calibration which is to obtain the relative position and orientation between the robot hand and the robot base, as shown in Figure 4.1.

We assume the camera is rigidly mounted on the robot hand and the object is placed in the camera field of view. There are four coordinate frames: the object frame $\{O\}$ centered at the object center, the camera frame $\{C\}$ centered at the camera lens center, the robot hand frame $\{H\}$ centered at the robot end-effector and the robot

base frame $\{R\}$ centered at the robot base. Let

$$ {}^{C}_{O}\mathcal{T} = \begin{bmatrix} {}^{C}_{O}R & {}^{C}_{O}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.1} $$

be the relative pose of the camera with respect to the object frame $\{O\}$, where the leading superscript indicates the frame in which the transformation is and the leading subscript indicates the frame to which the transformation is relative. Similarly, we have the relative pose of the camera with respect to the robot hand ${}^{C}_{H}\mathcal{T}$ and the relative pose of the robot hand with respect to the robot base ${}^{H}_{R}\mathcal{T}$.

Once above three poses are estimated, the 3D position and orientation of the object relative to the robot base frame is computed by

$$ {}^{O}_{R}\mathcal{T} = \sigma \; {}^{C}_{O}\mathcal{T}^{-1} \; {}^{C}_{H}\mathcal{T} \; {}^{H}_{R}\mathcal{T}. \tag{4.2} $$

Therefore, for any given particular point ${}_{O}\mathbf{p} = [\, X \quad Y \quad Z \quad 1 \,]^{T}$ (in the homogeneous coordinate) on the object, the robot can locate its end effector to that position using

$$ {}_{R}\mathbf{p} = {}^{O}_{R}\mathcal{T} \; {}_{O}\mathbf{p} \tag{4.3} $$

even if the point is not visible in the current view. Usually the robot kinematic system provides the relative pose of the robot hand ${}^{H}_{R}\mathcal{T}$. In the following we describe hand-eye calibration and object pose estimation in detail.

4.1 Hand-eye calibration

To compute the homogeneous transformation between the camera and the robot hand, we mount the camera on the robot hand and place a calibration object

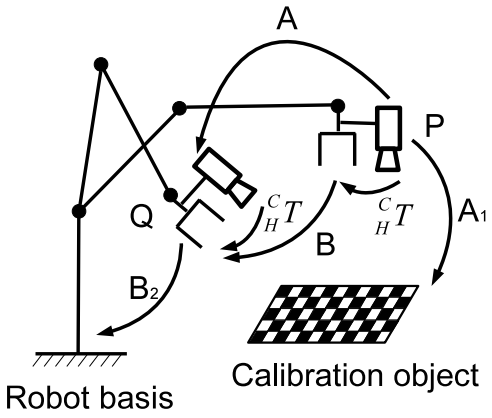

Figure 4.2: A robotic arm with a camera mounted at the robot hand moves from one position P to another position Q. A is the motion of the camera undergone with this robot movement and the corresponding motion of the robot hand is B.

in the front of the robotic arm. Generally, any appropriately characterized object could be used as a calibration object. Here we use a chessboard. Let

$$ {}^{C}_{H}\mathcal{T} = \begin{bmatrix} {}^{C}_{H}R & {}^{C}_{H}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.4} $$

be the relative rotation and translation of the camera with respect to the robot hand frame $\{H\}$.

A classic approach to formulate the problem is by using a robotic arm that makes a series of motions with a camera mounted at the robot hand, and at the same time the camera captures pictures of a calibration object placed in the front of the robotic arm at each motion [99]. Figure 4.2 shows the robotic arm moving from one position P to another position Q. Let

$$ A = \begin{bmatrix} R_a & \mathbf{t}_a \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.5} $$

be the motion of the camera undergone with this robot movement. The corresponding

motion of the robot hand is

$$B = \begin{bmatrix} R_b & \mathbf{t}_b \\ \mathbf{0} & 1 \end{bmatrix}. \tag{4.6}$$

The two motions are conjugated by the hand-eye transformation ${}^{C}_{H}\mathcal{T}$ [5, 88, 99, 42, 19, 66]. This yields

$$A\ {}^{C}_{H}\mathcal{T} = {}^{C}_{H}\mathcal{T} B \tag{4.7}$$

where $A = A_1 A_2^{-1}$ and $B = B_1 B_2^{-1}$. A_i is the pose of the camera with respect to the world coordinate system at each motion, and it can be estimated using the camera extrinsic calibration techniques. B_i is the pose of the robot hand with respect to the robot base system, that is usually provided by the robot kinematic system. The only unknown in the equation is ${}^{C}_{H}\mathcal{T}$. Applied each homogeneous transformation matrix, (4.7) is split into

$$R_a\ {}^{C}_{H}R = {}^{C}_{H}R\ R_b, \tag{4.8}$$

$$R_a\ {}^{C}_{H}\mathbf{t} + \mathbf{t}_a = {}^{C}_{H}R\ \mathbf{t}_b + {}^{C}_{H}\mathbf{t}. \tag{4.9}$$

Note once ${}^{C}_{H}R$ in (4.8) is solved, (4.9) can also be solved.

Many methods [5, 88, 99, 42, 19, 66] were proposed by researchers to solve this homogeneous matrix equation. Tsai and Lenz [99] first decouple the rotational part from the translation using the screw (angle-axis) representation and solve the equation using least squares with a closed form solution. The method yields a simple numerical solution, but its performance is limited by errors of the linear system, the parameters of the robot kinematic system and the estimations of the camera rotation and translation [42]. Park and Martin [66] solve the problem using methods of Lie theory in a similar way to [99], but also deal with the noise presented in the measurements of A and B. Horaud and Dornaika [42] use unit quaternion to present rotation and also find a closed form solution for rotation and translation simultane-

ously by a non-linear technique. In [19] dual quaternions are introduced by Daniilidis to perform hand-eye calibration. By using the dual-quaternion parameterization and singular value decomposition (SVD) the proposed method can quickly find a new simultaneous solution for the hand-eye rotation and translation. Based on Camera Calibration Toolbox for Matlab [8] and Hand-Eye Calibration Toolbox [104], we estimate the pose of the camera relative to the robot hand using the proposed approaches by [99, 42, 19, 66].

4.2 Object pose estimation

To estimate the relative pose of the object with respect to the camera mounted on the robot hand, we place the object in the camera field of view and take an image of the object at some viewpoint. There are several approaches we can use to estimate the relative 3D position and orientation of the object. One is perspective n-point (PNP) algorithm [62, 2], another is the POSIT algorithm [23].

Both approaches require 2D-3D correspondences between the current image and the 3D model as an input argument. To produce these correspondences, we use the affine SIFT (ASIFT) feature detector [61] to find features in the current image and the image sets used for reconstructing the 3D models. For every feature point in the current image, the matching feature point in the image sets is found, which is defined as the one that minimizes the sum-of-squared differences (SSD) between gray-level patches surrounding the two features. These matched points are potential point correspondences. Then, the same matching algorithm is run in the reverse order by swapping the roles of the images, and matches are retained if they agree in both directions. The image with the largest number of matches in the image sets is then used to produce 2D-3D correspondences between the current image and the 3D

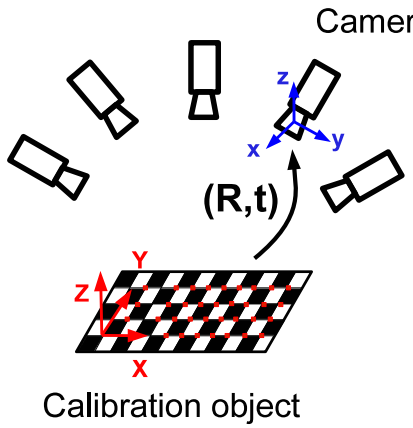

Figure 4.3: A camera moves around a chessboard placed on a plane and captures multiple views of the calibration object from different view points.

model. For each oriented 3D point, its closest ASIFT feature in the selected image is found by projecting the point onto the image plane according to the determined camera parameters. Correspondence between the current image and the 3D model is thus established using the matching of these closest ASIFT features.

4.2.1 Perspective n-point (PNP) algorithm

In OpenCV [2], the solvePnP function is equivalent to finding the extrinsic camera parameters. OpenCV uses a chessboard as the calibration object which is held by a person or placed on a plane. Figure 4.3 shows that a camera moves around a chessboard placed on a plane and captures multiple views of the calibration object from different view points.

The image formation process is to map points in the world coordinate (X, Y, Z) to points in the image plane (x, y). The mapping is represented as following:

$$\mathbf{p} = \mathcal{T}\ P \tag{4.10}$$

where $\mathbf{p} = [\, x \quad y \quad w \,]^T$, $P = [\, X \quad Y \quad Z \quad W \,]^T$ in the homogeneous coordinate and

$$\mathcal{T} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [\, R \quad \mathbf{t} \,] = K \; D. \tag{4.11}$$

where K is the camera intrinsic matrix and D is the camera extrinsic matrix. In general, the camera intrinsics parameters include focal lengths (f_x, f_y) in pixels along x and y axes, and a principal point (c_x, c_y) which is a displacement of the optic axis away from the center of coordinate on the image plane. The camera extrinsic parameters are the relative position $(\mathbf{t})$ and orientation (R) of the camera with respect to the object frame.

In Figure 4.3, the object coordinate is defined so that $Z = 0$. Applied $R = [\, r_1 \quad r_2 \quad r_3 \,]$ and (4.11), (4.10) is rewritten as

$$\begin{aligned} \mathbf{p} &= K \, [\, r_1 \quad r_2 \quad r_3 \quad \mathbf{t} \,] \begin{bmatrix} X \\ Y \\ 0 \\ W \end{bmatrix}, & (4.12) \\ &= K \, [\, r_1 \quad r_2 \quad \mathbf{t} \,] \begin{bmatrix} X \\ Y \\ W \end{bmatrix}, & (4.13) \\ &= H \; P'. & (4.14) \end{aligned}$$

where H is the homography matrix that maps a planar object's points onto the image's points. We assume the camera intrinsics parameters are known (They can be estimated by the same way with multiple views of the object). For each camera position there are six unknowns including three angles for the rotation and three offsets

for the translation. To solve H, we at least need 4 pairs of 2D-3D correspondences which yield eight equations. OpenCV extracts the corners of the chessboard as the feature points which are shown in Figure 4.3 by red points on the chessboard. In practice, due to noise of images capturing and errors of numerical calculation, the more correspondences we use, the better result we get. Once the homography matrix H is solved by linear algebra techniques, the camera extrinsic parameters can be obtained simultaneously. More details can be found in [112]. We feed the 2D-3D correspondences to OpenCV solvePnP routine. It turns out that solvePnP works well for planar objects (chessboards) or near planar objects (human's faces) due to the above algorithm OpenCV used.

4.2.2 POSIT algorithm

The POSIT (Pose from Orthography and Scaling with Iterations) algorithm is to estimate the pose of an object from a single view using at least four or more feature points [23]. The algorithm does not require the correspondences to be planar. The approach iteratively approximates the position and orientation of the object obtained by POS (Pose from Orthography and Scaling) algorithm which simplifies a perspective projection with a scaled orthographic projection. A scaled orthographic projection is an orthographic projection followed by a scaling. In Figure 4.4, an object AB is projected to AB' by an orthographic projection on a plane Q which is parallel to the image plane. Then AB' is projected to ab on the image plane by a perspective projection. AB' is scaled down to ab by a scaling factor that is defined by the focal length and the depth of the object.

Since an object pose can be exactly estimated by the POS algorithm with feature points on the object and their projected points on the image by a scaled

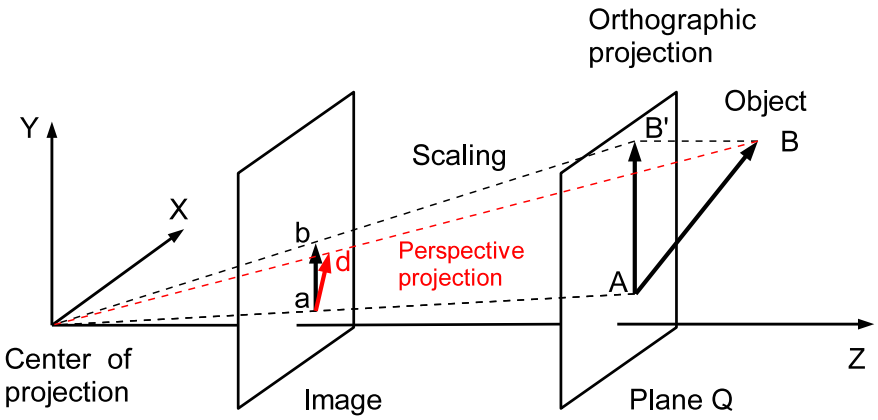

Figure 4.4: Scaled orthographic projection and perspective projection. The scaled orthographic projection ab of an object AB is the orthographic projection AB' followed by a scaling. ad is the perspective projection of the object AB.

orthographic projection (SOP), the POSIT algorithm first assumes that the given image points are projected by a SOP instead of by a perspective projection. Then the POS algorithm is applied to these points and an approximate pose of the object is obtained. Next, the feature points and the SOP image are updated by the approximate pose. Then the POS algorithm is applied to the new SOP image again to improve SOP image points. These steps are repeated until an accurate object pose is found.

Alternative approaches include the proposed algorithm in [22] and the SoftPOSIT algorithm [20]. DeMenthon and Davis [22] introduce nonlinear terms in the perspective projection model of an object. Additional image uncertainty about these nonlinear terms can be obtained in two 4D spaces by linear constraints, which leads to high-order complexity of algorithm performance. Then binary tree search techniques are used to find the object pose. The SoftPOSIT algorithm computes correspondences between feature points on the object and their image points when they are unknown

using the iterative softassign algorithm [37], and then estimates the object pose by POSIT algorithm [23]. Unlike [22, 23], the SoftPOSIT algorithm does not require small sets of matching points between the object and the image. We feed the 2D-3D correspondences to the publicly available different versions of POSIT codes [21], and it turns out that the classic POSIT algorithm [23] works well for 3D objects (e.g. a toy truck) that have depth.

4.3 Experimental results

To manipulate objects, we first estimated the scale factor (σ) between the object and the reconstructed 3D articulated model. We used a known length (e.g. the distance between the center of the front wheel and the back wheel of the toy truck which is about 50 millimeters) to compute the scaler.

We perform hand-eye calibration each time when we rigidly mount a camera on the robot hand. A chessboard was placed in the camera field of view, then the camera moved around the chessboard and took a sequence of images about it from different view points. As we described in Chapter 4, hand-eye calibration involves estimating the relative position and orientation of the camera with respect to the robot hand, i.e., the extrinsic parameters of the camera. In the experiments, we first used the same set of chessboard images to find the intrinsic parameters of the camera which is required to be known during the hand-eye calibration.

A publicly available Matlab toolbox [8] is used to calibrate the camera. There are total 20 different views of the chessboard as shown in Figure 4.5. The number of squares on the chessboard is 7×9, and the window size of each square is 10×10 millimeters. Figure 4.6 shows extracted corners (red crosses) for the first two calibration images. The corners were detected to an accuracy of about 0.1 pixel [8].

An initial closed-form solution is computed for the calibration parameters using these extracted corners on all images, then the total reprojection error is minimized over all the calibration parameters by a non-linear optimization technique. Table 4.1 shows the intrinsic parameters of a Logitech Quickcam Pro 5000 after 21 iterations of non-linear optimization. All 20 camera extrinsic parameters (position and orientation) are shown in Figure 4.7. In Table 4.1 pixel errors $[\,0.10786 \quad 0.10637\,]$ are the standard deviation of the reprojection errors (in pixel) in both x and y directions respectively. We can see the errors are very small. Figure 4.8 shows extracted corners (red crosses) and their corresponding reprojected grid corners (blue circles) for the first calibration image.

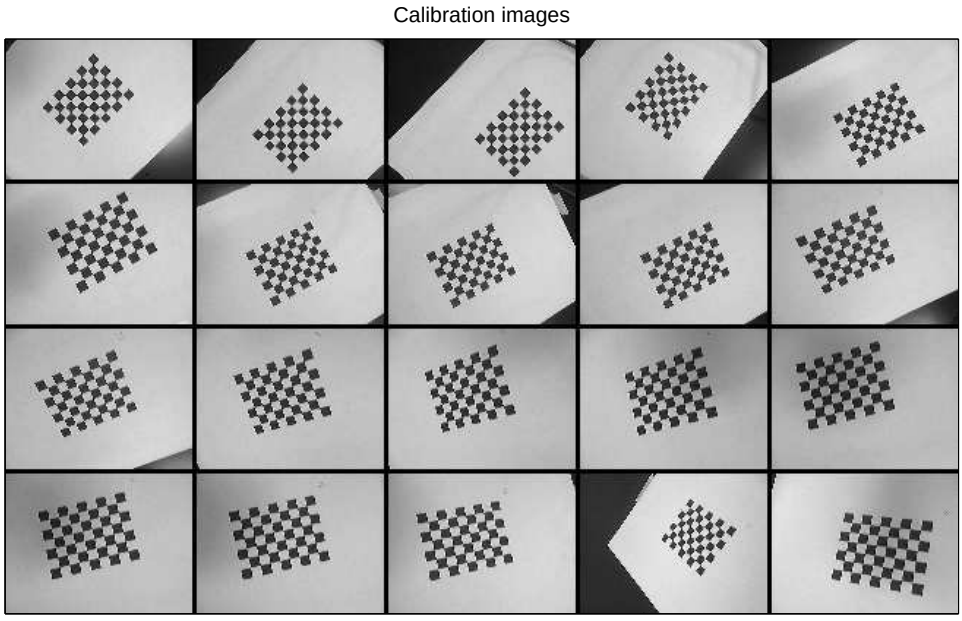

Figure 4.5: Total 20 different views of the chessboard for hand-eye calibration.

Then referring to [104], we computed the relative position and orientation of the camera with respect to the robot hand using above estimated camera parameters and the relative poses of the robot hand. Table 4.2 shows the hand poses of all 20

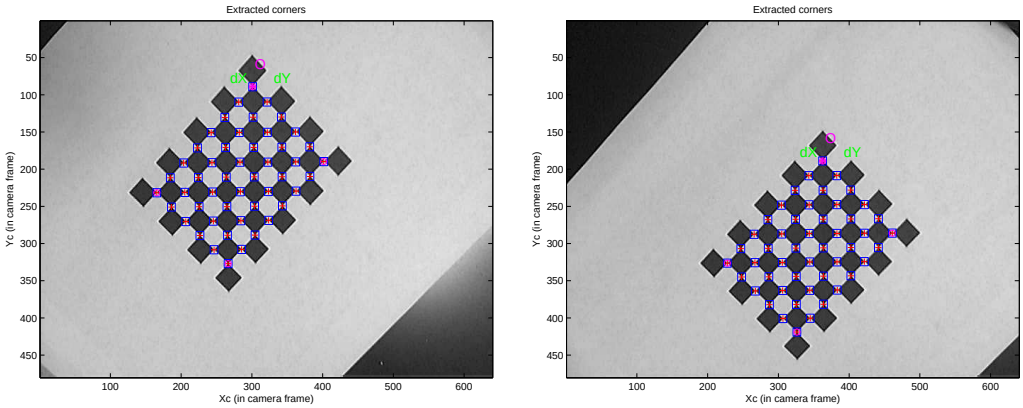

Figure 4.6: Extracted corners (red crosses) for the first two calibration images. The blue squares around the corner points show the limits of the corner finder window (The default size is 5×5 in millimeters).

Parameter	Value
Focal Length	$[\,808.13754 \quad 810.55808\,] \pm [\,5.36571 \quad 5.37796\,]$
Principal point	$[\,332.94102 \quad 277.22743\,] \pm [\,3.91433 \quad 3.90086\,]$
Skew	$[\,0.00000\,] \pm [\,0.00000\,]$, angle of pixel axes $= 90.00000 \pm 0.00000$
Distortion	$[\,0.10437 \quad -0.28098 \quad 0.00598 \quad -0.00375 \quad 0.00000\,] \pm$ $[\,0.02146 \quad 0.24123 \quad 0.00221 \quad 0.00213 \quad 0.00000\,]$
Pixel error	$[\,0.10786 \quad 0.10637\,]$

Table 4.1: The calibration results of a Logitech Quickcam Pro 5000.

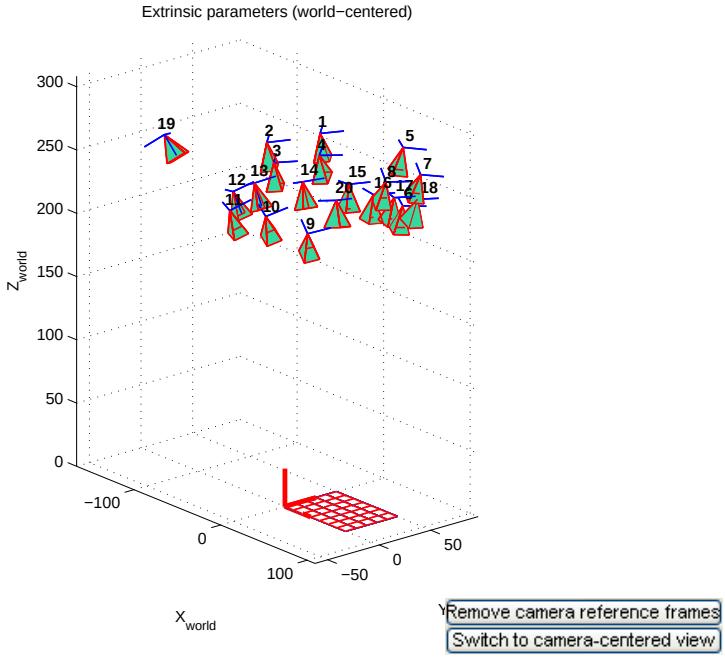

Figure 4.7: All 20 camera positions and orientations.

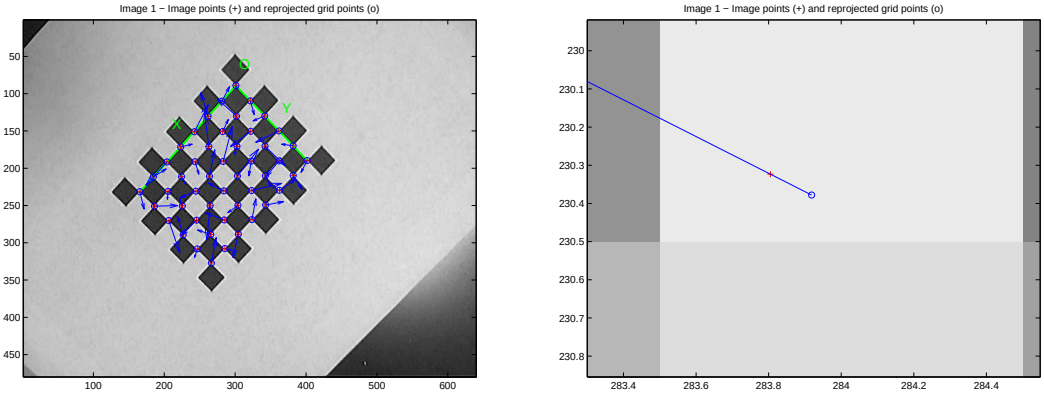

Figure 4.8: LEFT: Extracted corners (red crosses) and their corresponding reprojected grid corners (blue circles) for the first calibration images. The blue arrows represent the reprojection error and direction. RIGHT: The zoom in image of one corner in the left image.

calibration images that are provided by the robot kinematic system. Each robot hand pose is presented by six values, in which first three values are the position (X, Y, Z) in millimeters, and next three values are the rotation angles (R_x, R_y, R_z) in radians relative to the robot base frame respectively. Four algorithms [19, 42, 66, 99] were used to perform hand-eye calibration, and it turned out algorithms [42, 66, 99] gave the almost same results and are usually a bit better than the Dual quaternion approach in [19]. Following is the result of hand-eye calibration using Tsai and Lenz's algorithm [99].

$$ {}^{C}_{H}\mathcal{T} = \begin{bmatrix} -0.7332 & -0.6733 & -0.0954 & -20.3796 \\ 0.6800 & -0.7241 & -0.1149 & 33.2782 \\ 0.0083 & -0.1491 & 0.9888 & 67.7126 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{4.15} $$

The average reprojection error is 1.0729 which is small.

Next, we placed an object (e.g., a toy truck) in the camera field of view, and took an image of the object at some viewpoint. Figure 4.9 shows two example images captured for manipulation, showing the variety of poses encountered. The vast difference between such views and the images used for creating the models leads to difficulty for the SIFT algorithm to find matches. We have found that ASIFT produces significantly more matches for this problem. On four example images, the ratios of the number of matches found by ASIFT to SIFT are 58:2, 70:3, 37:1, and 72:4. The latter example, in which ASIFT found 72 matches but SIFT found only 4, is shown in Figure 4.10. Then we fed the 2D-3D correspondences to the publicly available different versions of POSIT codes [21], and obtained the relative pose of the

Image	Robot hand pose
1	9.96, 334.2, 12.04, 180, 0, 86.93
2	1.67, 294.04, 12.04, 180, 0, 86.93
3	52.43, 264.85, 12.04, 179.93, -4.31, 86.93
4	87.05, 270.39, 13.79, 9.74, -168.83, -86
5	75.75, 399.66, 3.56, 163.75, -7.58, 69.16
6	75.75, 397.51, -43.65, 168.45, -7.58, 69.16
7	122.75, 383.33, -13.28, 168.05, -16.8, 67.2
8	129.96, 340.87, -13.28, 175.89, -20.03, 66.47
9	123.79, 256.94, -35.25, -171.13, -15.32, 66.81
10	57.85, 256.94, -35.25, -165.7, -8.19, 64.93
11	1.79, 256.87, -35.25, -166.15, 2.77, 63.32
12	-65.1, 314.2, -35.25, -175.97, 15.62, 59.49
13	-102.43, 370.58, -44.91, 174.3, 21.65, 60.11
14	-72.39, 406.94, -44.98, 166.81, 17.92, 59.17
15	-18.77, 422.18, -44.98, 166.43, 5.75, 56.11
16	28.63, 415.35, -44.98, 166.5, -1, 54.48
17	59.64, 415.38, -44.98, 166.4, -6.93, 53.05
18	95.85, 415.38, -44.98, 166.05, -14.47, 51.17
19	-216.38, 363.45, -31.64, -176.74, 42.64, 3.31
20	-35.18, 427.86, -55.78, 162.81, 16.65, 35.35

Table 4.2: The relative robot hand poses for all 20 calibration images.

Figure 4.9: Top: Ten images taken at arbitrary viewpoints for pose estimation of the truck. Bottom: The corresponding images that, among all the images used for 3D reconstruction, have the highest number of matched features points for each image.

toy truck with respect to the camera:

$$ {}^{O}_{C}\mathcal{T} = \begin{bmatrix} 0.5693 & 0.7552 & 0.4709 & -0.0781 \\ 0.6536 & -0.3796 & -0.5698 & 0.0621 \\ -0.2515 & 0.6321 & -0.7097 & 1.3988 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{4.16} $$

Once ${}^{C}_{H}\mathcal{T}$ and ${}^{O}_{C}\mathcal{T}$ were obtained and the robot hand pose (${}^{H}_{R}\mathcal{T}$) with respect to the robot base frame was provided by the robot kinematic system, we can compute the 3D position and orientation of the object relative to the robot base frame by (4.2). Then the robot can locate its end effector to any given particular point by (4.3) even if the point is not visible in the current view. Given a manually specified grasp location, the task was to move the articulated body to a particular configuration. Based on the kinematic model, the trajectory required to achieve this configuration was determined automatically. Figure 4.11 shows parts of the video sequence of the PUMA manipulating a toy truck.

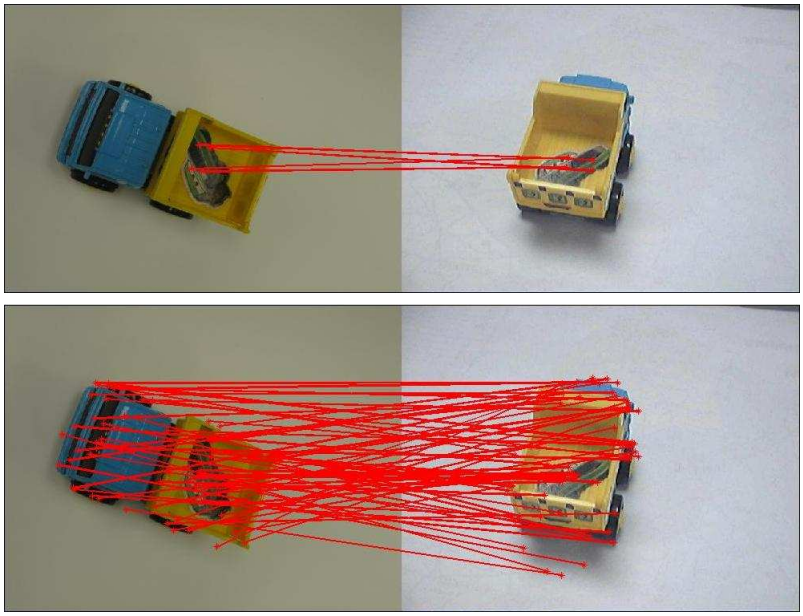

Figure 4.10: Comparison of SIFT (top) and ASIFT (bottom) feature matching. The red lines indicate matches between features detected in an image used for pose estimation (left) and an image used to create the 3D model (right). In this case SIFT finds 4 matches, while ASIFT finds 72.

Figure 4.11: Frames of a video sequence in which the robot manipulated a toy truck.

Chapter 5

Using RGBD Sensors

There is plenty of room for further improvements to the approach presented. For real-world applications, it will be important to remove the need for artificial markers attached to objects, whose sole purpose is to facilitate correspondence between images. One solution to address this issue is to use RGBD sensors that yield a depth value for each pixel immediately by the sensor itself rather than requiring correspondence to establish depth. The proposed approach is able to be adapted to such sensors with only minimal changes in the processing pipeline, leading to much denser 3D models, as well as reducing the computation time required.

5.1 RGBD sensors

Before we introduce RGBD sensors, we first present the image formation process and the range imaging both of which are key parts of RGBD sensors.

5.1.1 Image formation

The image formation process of a camera is a mapping of points in 3D world to 2D points on an image plane. Figure 5.1 shows a pinhole camera model, in which a ray of light from a point ($B = [\, X_0 \quad Y_0 \quad Z_0 \,]$) on an object in the world passes through an aperture, and hits on a recording surface (film), then produces an image point ($b = [\, x_0 \quad y_0 \,]$). The position of the image point b can be computed by:

$$-x_0 = f\frac{X_0}{Z_0} \tag{5.1}$$

$$-y_0 = f\frac{Y_0}{Z_0} \tag{5.2}$$

where f is the focal length of the camera. The value at image points are primary color of pixels, which is usually the gray intensity for grayscale images or red, green and blue channel values for color images. From Figure 5.1, we can see the object AB and $A'B'$ had the same projection (image) on the film even through they are different in the size and the distance from the aperture. Therefore the image formation process loses the information of one dimension and leads to a scale ambiguity [40].

5.1.2 Range imaging

To reconstruct 3D structure of a scene, it needs a new sensor device, called range camera, to capture or measure the missed information of the third dimension. Range camera can produce a 2D range image in which each pixel's value is the distance of visible points in a scene to a reference point or frame. For example in Figure 5.1 if the aperture is chosen to be the reference point, Z_0 or Z_0' is the value of the image point b for the object AB or $A'B'$. Range images are also referred to as depth images. Since range cameras can work according to a wide variety of different techniques, in

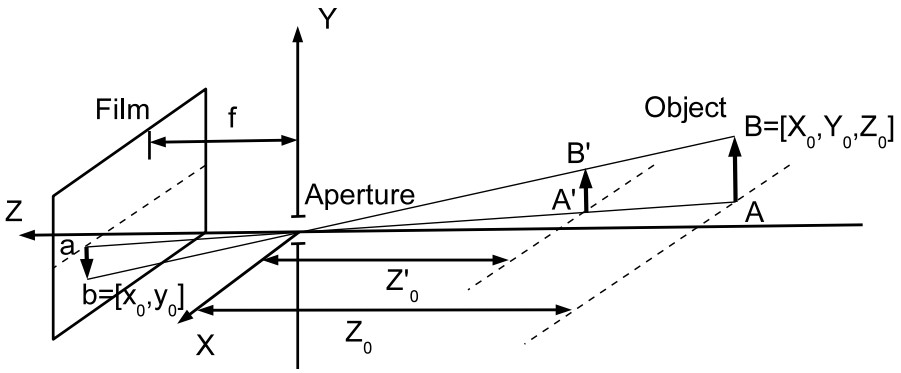

Figure 5.1: Pinhole camera model: a ray of light from a point on an object in the world passes through an aperture, and hits on a recording surface (film), then produces an image point.

the following sections we only introduce the most common approaches.

5.1.2.1 Stereo triangulation

A stereo camera system consists of a pair of cameras mounted side by side to obtain left and right images, simulating human binocular vision. Figure 5.2 shows the standard stereo model for two pinhole cameras whose optical axes are parallel. C and C' are lens centers of cameras, and the left and right image planes are coplanar. In the model, a world point Q is projected into a pair of corresponding points q and q' on the left and right image, and lies at the intersection of two rays from q through C and from q' through C' respectively.

The stereo triangulation is to find the intersection of the two rays in 3D space. From Figure 5.2, it's easy to find two triangles (Qqq') and (QCC') are similar, therefore we get

$$\frac{b + x - x'}{b} = \frac{f + Z}{Z}. \tag{5.3}$$

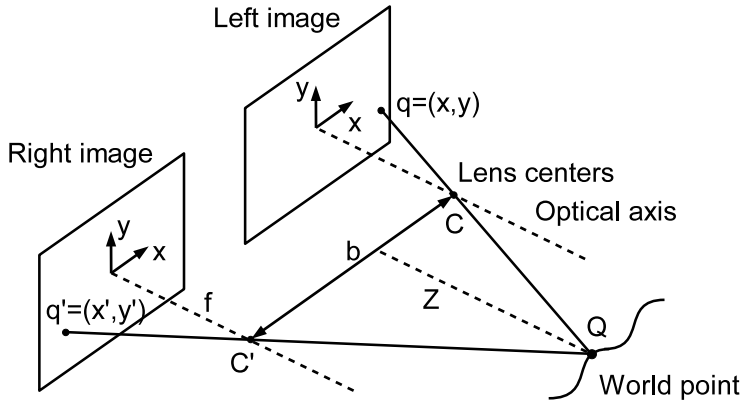

Figure 5.2: Standard stereo model for two pinhole cameras whose optical axes are parallel. A world point Q is projected into a pair of corresponding points q and q' on the left and right image, and lies at the intersection of two rays from q through C and from q' through C' respectively.

Solving (5.3) for Z, we have

$$Z = \frac{bf}{x - x'} \tag{5.4}$$

where b, the distance between lens centers C and C', is called the baseline of the stereo camera, f is the focal length of cameras, x and x' are image points coordinates. We assume the parameters b and f can be obtained from a prior stereo system calibration. The only left parameters x and x' are the most important problem for the stereo triangulation, especially it is hard to find correspondences for less or non texture images. Due to noises during the correspondences detecting, the two rays may not intersect in space. For this situation, the midpoint of the segment orthogonal to both rays is assumed to be the intersection [98]. Also for a general stereo system with non-parallel cameras, it requires to include more parameters such as camera extrinsic parameters (rotation and translation) to compute the world points. Figure 5.3 shows a general stereo model in which two cameras are not parallel, and two rays from C

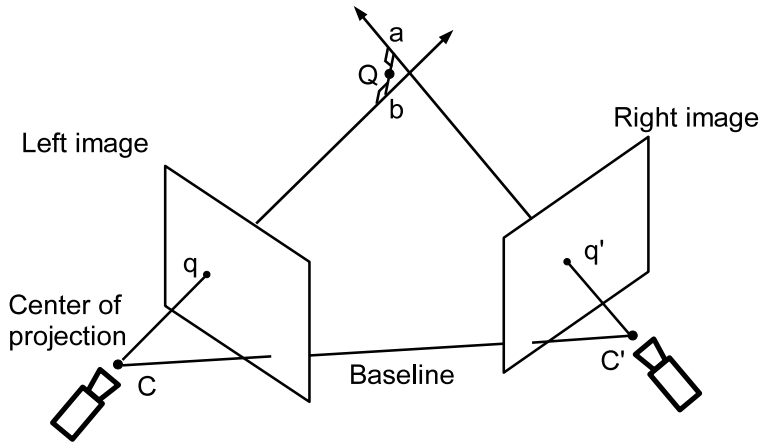

Figure 5.3: General stereo model for two non-parallel cameras. Two rays from C through q and from C' through q' do not intersect due to some noises. The midpoint Q of the segment ab that is orthogonal to both rays is assumed to be the world point.

through q and from C' through q' do not intersect due to some errors. Therefore the midpoint Q of the segment ab that is orthogonal to both rays is assumed to be the world point.

5.1.2.2 Structured lighting

Structured lighting is an approach to extract geometric structure of objects in a scene by projecting known light patterns onto the scene. The approach effectively solves correspondences problem by searching the patterns in the camera image instead of using classic feature matching techniques. Common used light patterns are rays, planes, grids, parallel stripes, encoded lights and so on. A standard structured light 3D scanner consists of one light source and one camera. Figure 5.4 shows a simple model in 2D , in which a laser light source projects a light spot on an object surface and a camera observes the spot. For the triangle (OPL), according to law of sines,

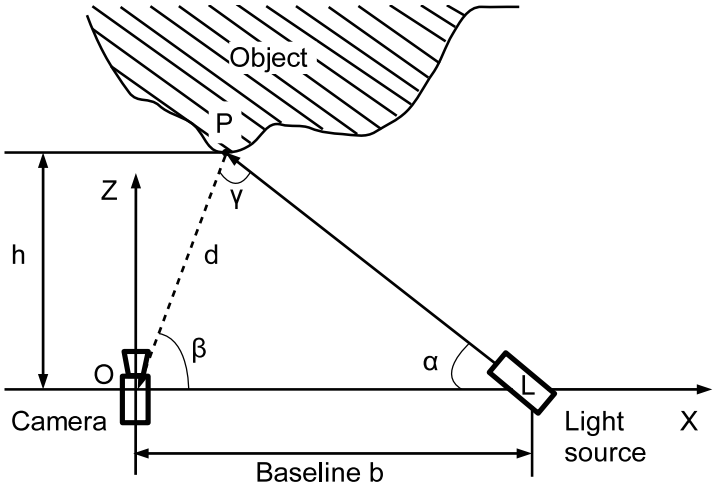

Figure 5.4: A simple structured light 3D scanner model consists of a light source and a camera, in which the laser light source projects a light spot on an object surface.

we have

$$\frac{d}{\sin\alpha} = \frac{b}{\sin\gamma} \tag{5.5}$$

where $\gamma = \pi - \alpha - \beta$. Solving d, we get

$$d = \frac{b\ \sin\alpha}{\sin\gamma} = \frac{b\ \sin\alpha}{\sin(\pi-\alpha-\beta)} = \frac{b\ \sin\alpha}{\sin(\alpha+\beta)} \tag{5.6}$$

where b, α and β are assumed to be known. Therefore the position of the object point can be obtained $P = (d\ \cos\beta, d\ \sin\beta)$. In the similar way, 3D object points can be computed based on above trigonometry and known geometry of the camera and the light source. Instead of projecting a light spot onto the object, a single stripe of laser light is projected and scanned across the surface of the object to obtain a high resolution 3D structure. At the mean time, parallel stripes are widely used to speed up the measuring patterns.

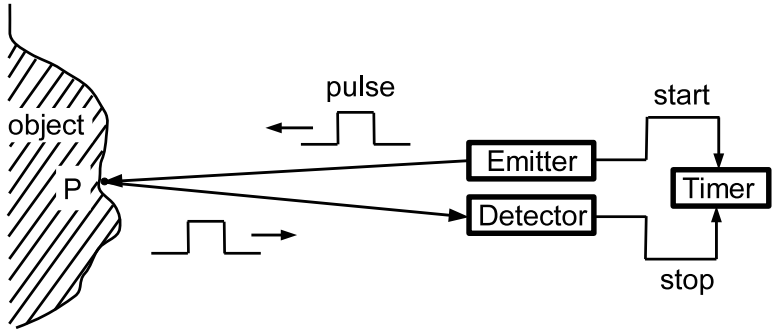

Figure 5.5: A pulsed time-of-flight system: emitted laser or light pulses are projected onto 3D object surfaces and reflected back to a detector, a timer measures the absolute time of the pulses traveling from the emitter to the 3D scene and back.

5.1.2.3 Time-of-flight

Another often used approach measuring the depth of a scene is time-of-flight technique. Figure 5.5 shows a pulsed time-of-flight system, in which emitted laser or light pulses are projected onto 3D object surfaces and reflected back to a detector, at the same time a timer measures the absolute time of the pulses traveling from the emitter to the 3D scene and back. The distance of 3D objects from the observer can be computed using the measured time and the known speed of light. The system requires high accuracies of time measurements and has a large cover range up to about $60m$. However due to light scattering, measurements of bounced pulses are inexact.

5.1.3 Microsoft Kinect

RGBD sensors are novel motion sensing systems such as the Microsoft Kinect and the Asus Xtion sensor which can capture and track the movement of objects and people in 3D space. In this research, we use the Microsoft Kinect which was first launched in November 2010 by Microsoft for the Xbox 360 video game console. In

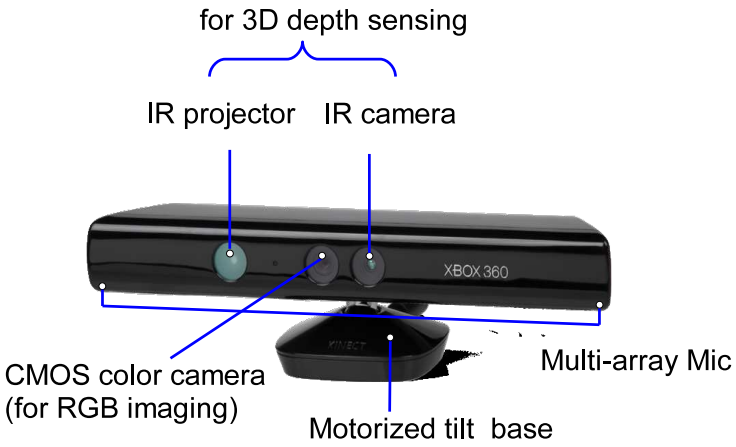

Figure 5.6: Microsoft Kinect diagram: the device hardware mainly includes a RGB camera, a depth sensor, and a multi-array microphone.

the past couple years, such a new sensor has caused an explosion of innovation in the robotics world and computer vision community. Figure 5.6 shows a Microsoft Kinect diagram. The device hardware mainly includes a CMOS color camera for RGB imaging, an infrared (IR) laser projector and a CMOS IR camera both used for 3D depth sensing, and a multi-array microphone for interacting with users.

The RGB color camera is a simple webcam, which is similar to one on laptops or phones. The camera is able to record color videos at a frame rate up to 30 Hz with the default 8-bit VGA resolution (640×480 pixels). High resolution videos can be captured by the device but at a lower frame rate.

The depth sensor is developed by PrimeSense company operating in principle to the structured lighting technique for generating 3D depth images. The sensor consists of an infrared (IR) laser projector and a CMOS IR camera. Invisible infrared light is used to prevent perceivable disturbances from the environment. According to structured lighting approach, the projector projects a known light pattern onto a scene and the camera observes the pattern's deformation due to lights striking the

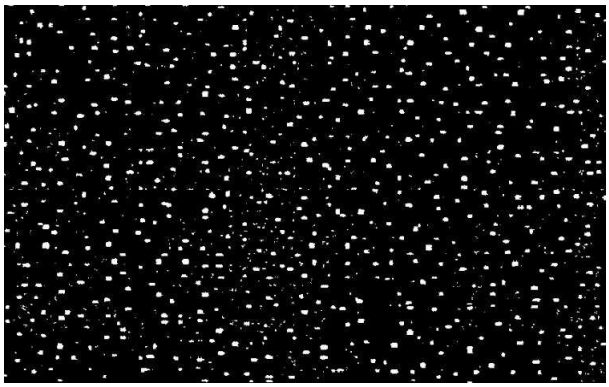

Figure 5.7: Light pattern used by Microsoft Kinect is a static cloud of variably intense dots that appears to be random.

3D surface, then the distance of the 3D surface to the camera is computed. The light pattern used by Kinect is a speckle pattern (Figure 5.7), which is a static cloud of variably intense dots that seem to be random [49]. The ranging limit of the depth sensor is $1.2m$ to $3.5m$ in practical although the device can work at up to $6m$. Depth images have VGA resolution (640×480 pixels) with $2,048$ levels of sensitivity (11-bit depth).

There are two main Kinect development softwares. One is Microsoft Kinect for Windows SDK (software development kit) which is released in June 2011 by Microsoft. The SDK includes drivers for using Kinect sensor devices, APIs for accessing the device hardwares, and source code samples of skeletal tracking, audio processing and so on. The SDK is free and enables developers to create applications by using Kinect sensor technology in C++, C#, or Visual Basic.Net on Windows 7. The other is OpenNI which is an open source SDK used for the development of 3D sensing middleware libraries and applications. OpenNI includes a variety of middleware libraries, tools, wrappers, and applications for multiple devices (not limit to Microsoft Kinect), also it works for multiple computer platform and supports developing softwares in Java, which are not allowed by Microsoft Kinect for Windows SDK.

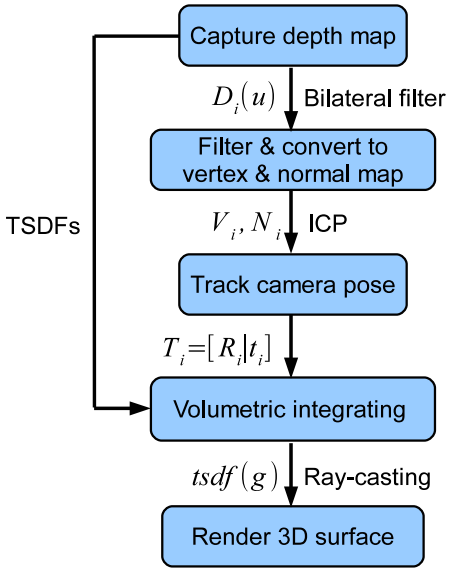

Figure 5.8: Overview of KinectFusion algorithm.

5.2 KinectFusion

KinectFusion algorithm [44] is to reconstruct real-time 3D indoor scenes using Microsoft Kinect, first developed as a research project at the Microsoft Research lab in Cambridge, U.K.. As a user holds and moves Kinect around an object in a scene or the object is moved around Kinect to be scanned, the sensor captures a sequence of depth images and integrates data to a single high-quality, geometrically accurate 3D model. The latest version (v1.7) of Microsoft Kinect for Windows SDK has included KinectFusion to allow users to scan and model 3D object using a Kinect sensor. Figure 5.8 shows the overview of KinectFusion algorithm. In the following sections, we introduce how the algorithm works step by step.

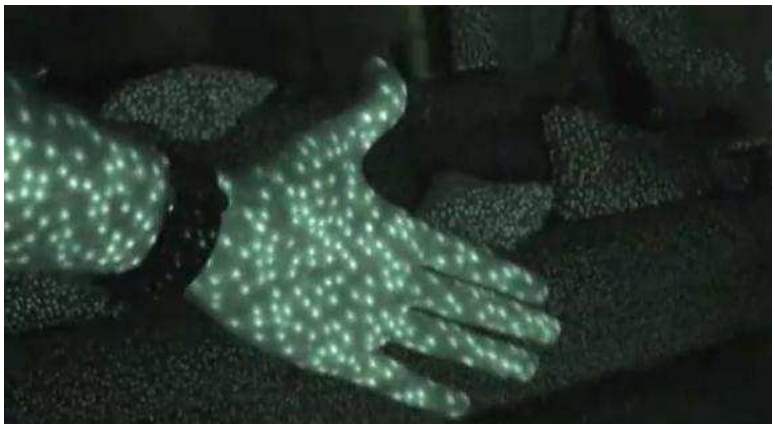

Figure 5.9: Speckled dots are projected on a scene from the IR projector [28].

5.2.1 Raw depth maps

Raw depth maps are generated by the depth sensor of Kinect. The IR projector projects a speckle light pattern (Figure 5.7) with about $30,000$ to $300,000$ dots onto a scene as shown in Figure 5.9 [28]. Some projected dots are distorted for example dots on the hand, and become ellipses whose orientation depends on the distance from objects to the projector. The IR camera perceives these information, and 3D correlations are computed for mapping the observed pattern to the known light pattern carried by the hardware. Once good matchings are found, the 3D positions of points are calculated according to triangulation geometry which is defined by the position of the IR projector and the IR camera, i.e. structured light technique (Figure 5.4). Figure 5.10 shows a depth image of a microwave in an office environment, in which depths are represented by gray levels. Darker pixels are, closer objects are to the sensor.

Figure 5.10: Depth image of a microwave in an office environment. Depths are represented by gray levels.

5.2.2 Vertex and normal maps

In raw depth maps, there are many pixels without depth values due to noises. To remove erroneous measurements, KinectFusion algorithm applies a bilateral filter to raw depth maps. This is just for converting depth maps to vertex and normal maps. Raw depth data are still be used for recovering the 3D structure. A vertex map is a 3D point cloud about visible objects in a scene with respect to the camera coordinate system, in which the origin locates at the principal point and the z-axis points to the scene. For each point (x, y) with a depth value (d) in a depth map, the corresponding 3D vertex $v(x, y)$ in metric units is obtained by

$$v(x,y) = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = d\ K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5.7}$$

where K is the intrinsic matrix of the IR camera. The normal vector $n(x, y)$ of the 3D vertex can be computed by cross product neighboring points:

$$n(x, y) = (v(x + 1, y) - v(x, y)) \times (v(x, y + 1) - v(x, y)). \tag{5.8}$$

It indicates the direction of the surface at the vertex, and is usually represented by an arrow starting with the vertex and pointing away from the surface.

5.2.3 Camera tracking

As Kinect moves around the object as shown in Figure 5.11, KinectFusion uses ICP (Iterative Closest Point) algorithm to estimate camera poses $T_i = [R_i|t_i]$. ICP algorithm aligns two point clouds by minimizing total errors of all corresponding points which are usually determined by finding the closest points between two point sets. Instead, KinectFusion algorithm assumes the motion between the consecutive positions of the sensor is small, and uses projective data association to find the correspondences between the frames. Let's say for the previous frame, the vertex map (V_{i-1}), the normal map (N_{i-1}) and the camera pose (T_{i-1}) are given. ICP will not run for the first frame, and the default first camera faces to the origin of the object coordinate system and locates slightly behind the camera. To find corresponding points for the current frame in the previous frame, each vertex (v_{i-1}) in the previous frame is transformed back to the IR camera coordinate system and perspective projected onto the image plane. Then a 3D vertex in the current frame is found by using the projected image point from the previous frame and the corresponding depth value in the current frame (5.7). The camera pose for the current frame is initialized to be the pose of the previous frame. Then the found corresponding vertex is transformed to a 3D point (v_i) in the global coordinate system using the initial guess. Two vertices (v_{i-1}

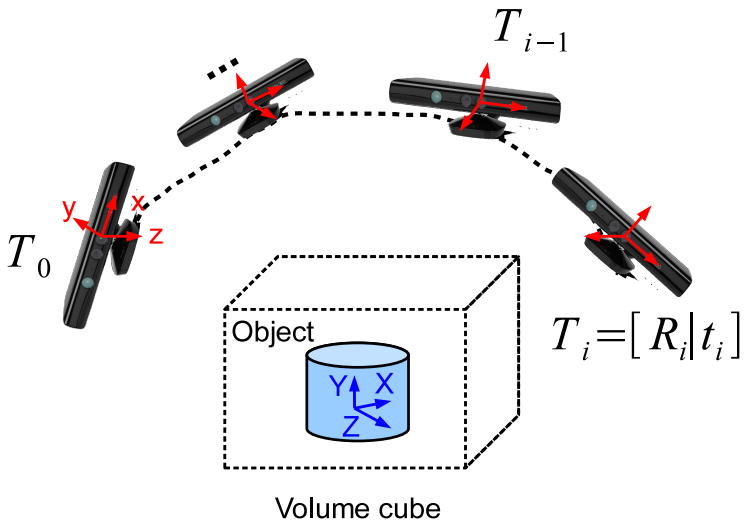

Figure 5.11: Camera tracking in KinectFusion.

and v_i) which either are too far or whose normal difference is too large are assumed to be outliers and rejected by the system. Otherwise two vertices are determined as the correspondences. Then a new transformation is computed using these set of correspondences. Above process is repeated until finding a final transformation (T_i) which minimizes the point-to-plane error between two surfaces. Also KinectFusion algorithm generates a vertex and normal map pyramid to improve camera tracking performance. The pyramid is normally 3 levels, and it down samples the depth map twice.

5.2.4 Volumetric integrating

To integrate the current raw depth data into the 3D model, KinectFusion algorithm uses a volumetric representation and Truncated Signed Distance Functions (TSDFs). A default 3D volume is a $3 \times 3 \times 3$ cube in meters as shown in Figure 5.11, which is subdivided uniformly into a set of voxels for example 512 voxels per axis. The

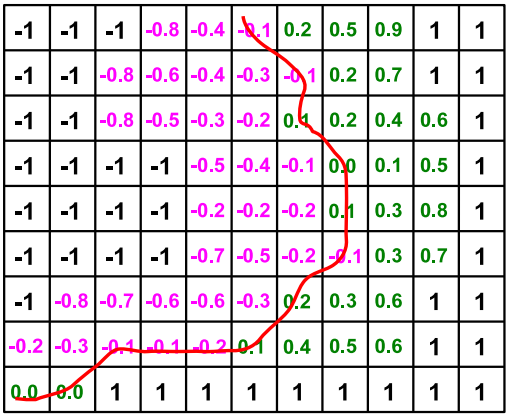

Figure 5.12: A TSDF volume grid, in which a positive TSDF value indicates the voxel is outside the surface, whereas a negative value represents the voxel is inside the surface, and the zero-crossing of TSDF defines the surface interface (red curve) where the values change sign.

size of the 3D volume and the number of voxels are changeable, and both determine the resolution of the 3D model. For each voxel, the algorithm computes a TSDF value in the range $[-1, 1]$ which is the distance to the nearest isosurface. A positive TSDF value indicates the voxel is outside the surface, whereas a negative value represents the voxel is inside the surface, and the zero-crossing of TSDF defines the surface interface where the values change sign. Figure 5.12 shows a TSDF volume grid, in which the red curve is the surface. Then the algorithm merges the new TSDF values for the current frame with the TSDF values of the current 3D model using weighted average.

5.2.5 Surface rendering

After integrating new data to the 3D model, KinectFusion algorithm applies a ray-casting technique to render the final model. In the output image each pixel casts a ray through the focal point as shown in Figure 5.13. The algorithm traverses

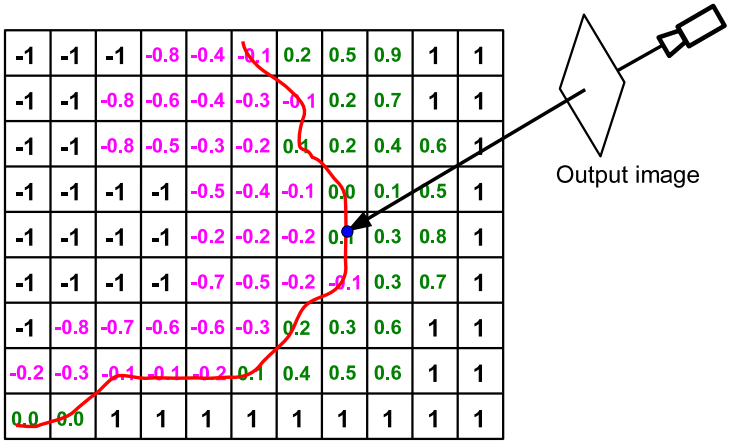

Figure 5.13: Ray-casting. In the output image each pixel casts a ray through the focal point. Voxels along the ray are traversed and the first surface that the ray hits is found by observing the sign change of TSDF values.

voxels along the ray, and finds the first surface that the ray hits by observing the sign change of TSDF values. Then the intersection point is computed using points around the surface boundary.

5.3 Learning articulated objects

To extend this work, we apply a new RGBD sensor (Microsoft Kinect) to the system and reconstruct kinematic structure of articulated objects. Figure 5.14 shows an overview of the system using Kinect sensor. First, a sequence of depth and color images are recorded by a Kinect sensor of the object from different viewpoints while the object remains stationary. KinectFusion algorithm is used to the images to build a 3D model of the object. In order to learn the object's kinematic structure, the configuration of the object is interactively changed by exercising its degrees of freedom. During the interaction with the object, we apply ICP technique and projective data association to detect outliers which mostly belong to the movable part of the

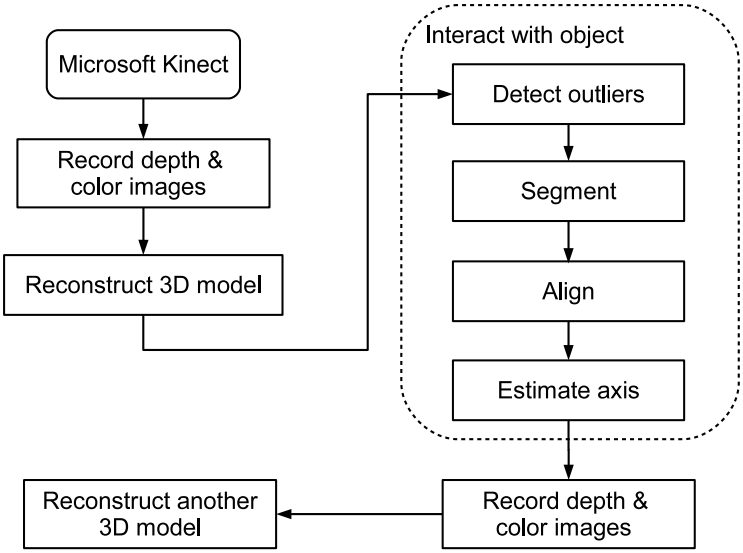

Figure 5.14: Overview of the system using Kinect sensor.

object. The object is segmented into rigid links using these outliers. Then the axis of each joint between neighboring links is found using the geometric method proposed in the previous system. After interacting with the object, the sensor moves around the object to capture the second set of depth and color images and reconstruct another 3D model.

5.3.1 Building 3D models

As Kinect captures color and depth images of a scene from different viewpoints, KinectFusion algorithm uses depth images to track the sensor position and integrates new data to a single 3D model. Figure 5.15 shows four color and depth images (out of hundreds of recorded) of a microwave in an office environment, and the 3D reconstruction of the microwave is shown on the bottom row. There is no constraint on the sequence of images, except that the motion between the consecutive positions of the sensor must be small in order to facilitate correspondences detecting using projective

Figure 5.15: Top: Four color images (out of hundreds of recorded) of a microwave. Middle: Four corresponding depth images of the microwave. Bottom: 3D models.

data association for ICP alignments. Another set of images of the microwave in a different configuration and the 3D reconstructions are shown in Figure 5.16.

5.3.2 Segmenting rigid links

In order to learn the object's kinematic structure, we interact with articulated objects by exercising its degrees of freedom, e.g. opening or closing the door of a microwave. Figure 5.17 shows four color and depth images (out of a set of captured) as we open the door of a microwave in an office. We apply ICP and projective data association techniques on two frames to detect outliers which are defined by two 3D points are either too far away or whose normal difference is too large. Among these outliers, most belong to the movable parts of the object for example the door of the microwave in Figure 5.17, however due to measurement errors of the sensor and

Figure 5.16: Top: Four color images (out of hundreds of recorded) of the microwave in a different configuration. Middle: Four corresponding depth images of the microwave. Bottom: 3D reconstructions.

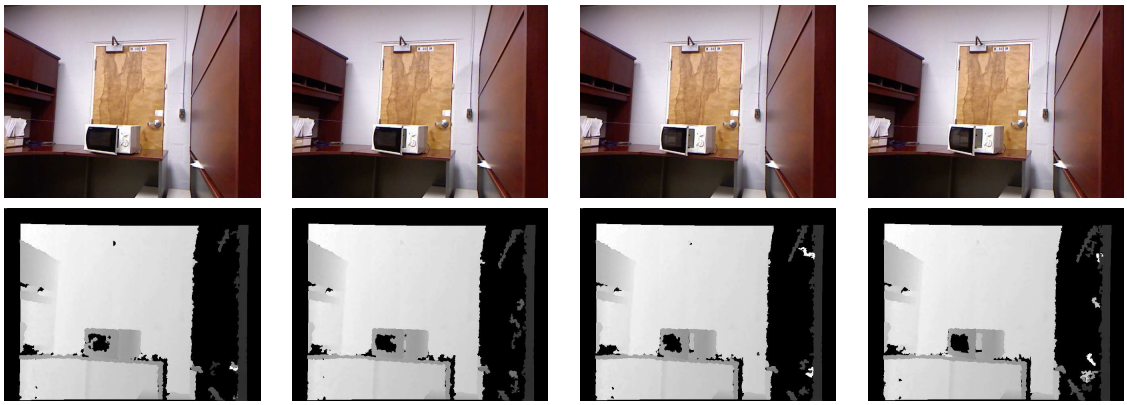

Figure 5.17: Top: Four color images (out of a set of captured) of a microwave as the door is being opened. Bottom: Four corresponding depth images of the microwave.

inaccuracies of ICP alignments there are still some detected outliers (noises) on the rest parts of the scene. One solution is to find outliers between any pair of frames at intervals which have larger motion than the consecutive frames. Therefore we can adjust ICP algorithm parameters (thresholds of distance and normal difference between two vertices) to remove noises. At the same time, in the system the Kinect sensor is stationary while objects are interacted by users. We extract the background of the scene and subtract it from detected outliers to segment the movable parts of the scene which are normally clustered whereas noises are dispersed.

5.3.3 Finding joint axes

Here we take an articulated object with two links (Link 0 and Link 1) as an example. Once rigid links are segmented for any pair of frames as the object is interacted by a user, we first align Link 0 (the base) of two configurations using camera poses achieved from ICP algorithm such that Link 1 is "almost aligned". We again apply ICP technique on the "almost aligned" Link 1 to find the transformation between them in two frames. The geometric approach utilizing an axis-angle representation

proposed in the previous system is then used to estimate the axis of the joint.

Figure 5.18 shows the estimated rotation axis (the red line) of a microwave in an office for two frames. The extended approach using a RGBD sensor is able to reconstruct much denser 3D models, segment the points into the individual links, and estimate the axis of rotation. Also it does not require artificial markers attached to the objects and is able to handle the less textured or untextured objects. The resulting models therefore include dense 3D point clouds representing the surfaces of the objects, along with the joint parameters. Another advantage of the approach is it does not require more than two configurations of the objects, therefore we can obtain one joint axis for any two frames with object motions and average these axes parameters (direction and location) to yield a final estimation.

One limitation of the above proposed approach is the assumption of KinectFusion algorithm, that is the motion between two consecutive sensor positions is small. With this assumption, KinectFusion detects the correspondences between the consecutive frames using projective data association technique instead of using the classic feature detectors, and further tracks camera poses by applying ICP algorithm. Since we use the detected correspondences and estimated camera poses to compute the transformation of rigid links between the frames and segment rigid links, the accuracy of detected correspondences affects the performance of the approach. For the case of a large sensor motion, we can apply the classic features detectors such as ASIFT feature detector or line detector to filter the mismatched correspondences found by KinectFusion.

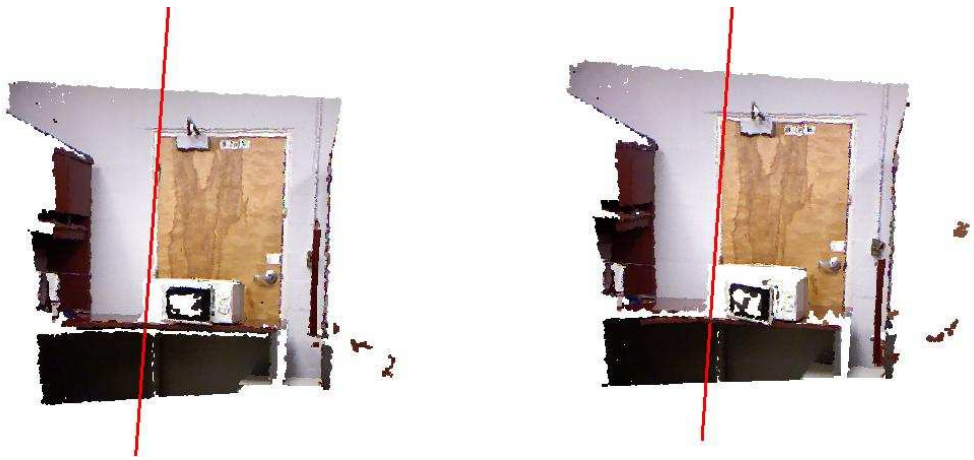

Figure 5.18: Red lines are the estimated rotation axes of a microwave for two different frames while the object is interacted by users.

Chapter 6

Conclusion

Autonomous operation in unstructured and dynamic environments has become an important trend in robotics. As robots move into unstructured environments such as homes, schools, and workplaces, new approaches to sensing and manipulation are required to handle the greater variety of objects encountered. For example, rather than expecting the robot to have advanced knowledge of all objects that will be encountered in the physical world, the ability to actively learn about the scene will be crucial. Recently reconstructing *articulated* objects has caught the attention of researchers. These objects consist of rigid links connected by one or more revolute or prismatic joints. A number of everyday objects, such as laptop computers, staplers, scissors, cabinet drawers, doors, and some cell phones fit such a model. Even a desk or chair sliding on the floor can be modeled, to some degree, as a set of prismatic and revolute joints.

6.1 Contributions

This work has presented novel algorithms regarding articulated objects reconstruction and operation. The main contributions of this dissertation are listed below.

1. An algorithm called Procrustes-Lo-RANSAC (PLR) is proposed to extract the 3D surface and kinematic structure of articulated objects. Multiple pictures are taken of an object by a single uncalibrated camera in two different configurations, and 3D models are reconstructed using structure-from-motion techniques. From these models, the rigid links of the object are segmented and aligned, allowing the joint axes to be estimated using a geometric approach. From the performance aspect, the algorithm

    - recovers *occlusion-aware* multi-view models which refer to a system that has knowledge about parts of the object that are not currently visible.
    - does not require prior knowledge of the object,
    - does not make any assumptions regarding planarity of the object,
    - supports both revolute and prismatic joints,
    - automatically classifies joints type,
    - works for objects with multiple joints,
    - only requires two different configurations of the object,
    - shows its effectiveness on a range of environmental conditions with various types of objects useful in domestic robotics.

2. The resulting occlusion-aware 3D articulated model can be used to enable a robot to manipulate the object. Two capabilities are supported by such a

model:

- Given a particular point on the object, the robot can move its end effector to that position, even if the point is not visible in the current view. This is one of the main advantages of the occlusion-aware approach, namely, that the robot is not limited only to the side of the object that is currently visible, but rather that a full 3D model is available.
- Given a particular grasp point, the robot can grab the object at that point and move in such a way so as to exercise the articulated joint.

3. A RGBD sensor (Microsoft Kinect) is used to improve the proposed approach. Kinect sensor captures not only a sequence of color images but also depth images of a scene, which facilitates the correspondences detection and camera tracking. KinectFusion algorithm is applied to yield a single high-quality, geometrically accurate 3D model from which rigid links of the object are segmented and aligned, allowing the joint axes to be estimated using the proposed geometric approach. From the performance aspect, the algorithm

   - does not require artificial markers attached to objects,
   - yields much denser 3D models,
   - reduces the computation time.

6.2 Future work

There is plenty of room for further improvements in this work. First, the proposed algorithm (PLR) is challenged by less textured or untextured objects. This limitation can be solved by attaching artificial markers to objects which is very common in the computer vision community. However for real-world applications it is

important to remove this need whose sole purpose is to facilitate correspondence between images. One possible solution to address this issue is to incorporate additional features such as line features into the proposed algorithm to increase performance. Comparing to point features (SIFT, SURF and so on), line features mostly locate at edges of objects or scenes and encode geometrical and structural information about them. Apart from including other 2D features, the algorithm could use 3D features like fast point feature histograms (FPFH) descriptors or viewpoint feature histogram (VFH) descriptors to find the correspondences between 3D models.

Another improvement would be to use new RGBD sensors instead of regular cameras as we introduced in Chapter 5. However KinectFusion is limited to work well when the motion between two consecutive sensor positions is small, which facilitates correspondences finding between frames during tracking camera poses using ICP and projective data association techniques. Moreover, this work could be extended to handle other types of joints presented in articulated objects section such as ball joints.

Bibliography

[1] Blender, http://www.blender.org/.

[2] Intel OpenCV library, http://www.intel.com/research/mrl/research/opencv/.

[3] 2009. Computing Community Consortium. A Roadmap for US Robotics: From Internet to Robotics. http://www.us-robotics.us/reports/CCC%20Report.pdf.

[4] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *Proceedings of the International Conference on Computer Vision*, Sept. 2009.

[5] N. Andreff, R. Horaud, and B. Espiau. Robot hand-eye calibration using structure from motion. *International Journal of Robotics Research (IJRR)*, 20(3):228–248, Mar. 2001.

[6] G. M. Bone, A. Lambert, and M. Edwards. Automated modeling and robotic grasping of unknown three-dimensional objects. In *Proceedings of the International Conference on Robotics and Automation*, May 2008.

[7] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, Nov. 2008.

[8] J.-Y. Bouguet, 2010. Complete Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

[9] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[10] A. Broadhurst, T. W. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *Proceedings of the International Conference on Computer Vision*, pages 388–393, July 2001.

[11] M. Brown and D. G. Lowe. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *Proceedings of the International Conference on 3D Digital Imaging and Modelling*, pages 56–63, 2005.

[12] A. O. Bălan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker. Detailed human shape and pose from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[13] M. D. Choi. Notes on the norm estimates for the sum of two matrices. *Acta Mathematica Sinica*, 19(3):595–598, July 2003.

[14] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. *Proceedings of the 25th DAGM Symposium*, pages 236–243, Sept. 2003.

[15] R. Collins. A space-sweep approach to true multi-image matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 358–363, June 1996.

[16] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the International Conference on Computer Vision*, pages 1071–1076, 1995.

[17] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011.

[18] W. B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *Workshop on the Seventh International Conference on Computer Vision (ICCV)*, pages 100–115, Sept. 1999.

[19] K. Daniilidis. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, 18(3):286–298, June 1999.

[20] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. Softposit: Simultaneous pose and correspondence determination. *International Journal of Computer Vision*, 59(3):259–284, 2004.

[21] D. DeMenthon, 2003. POSIT, http://www.cfar.umd.edu/~daniel/Site_2/Code.html.

[22] D. DeMenthon and L. Davis. Recognition and tracking of 3D objects by 1D search. In *Proceedings of the Image Understanding Workshop*, pages 653–659, 1993.

[23] D. Dementhon and L. S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1–2):123–141, June 1995.

[24] Y. Duan, L. Yang, H. Qin, and D. Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. In *Proceedings of the European Conference on Computer Vision*, pages 238–251, May 2004.

[25] D. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-D rigid body transformations: A comparison of four major algorithms. *Machine Vision and Applications*, 9(5-6):272–290, Mar. 1997.

[26] O. Faugeras and R. Keriven. Variational principles, surface evolution, pdes, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):335–344, Mar. 1998.

[27] J. Fayad, C. Russell, and L. Agapito. Automated articulated structure and 3D shape recovery from point correspondences. In *Proceedings of the International Conference on Computer Vision*, pages 431–438, Nov. 2011.

[28] M. Fisher. Matt's Webcorner,http://graphics.stanford.edu/~mdfisher/Kinect.html.

[29] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3-4):143–166, Mar. 2003.

[30] D. A. Forsyth, O. Arikan, L. Ikemoto, J. O'Brien, and D. Ramanan. Computational studies of human motion: Part 1, Tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2/3), 2006.

[31] O. Freifeld, A. Weiss, S. Zuffi, and M. J. Black. Contour people: A parameterized model of 2D articulated human shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[32] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *Proceedings of the International Conference on Computer Vision*, Sept. 2009.

[33] Y. Furukawa and J. Ponce, 2007. PMVS, http://www.cs.washington.edu/homes/furukawa/research/pmvs.

[34] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[35] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, Aug. 2010.

[36] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *Proceedings of the International Conference on Computer Vision*, 2007.

[37] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996.

[38] P. Guan, A. Weiss, A. O. Bălan, and M. J. Black. Estimating human shape and pose from a single image. In *Proceedings of the International Conference on Computer Vision*, 2009.

[39] E. Guizzo. A robot in the kitchen. *IEEE Spectrum*, Apr. 2010.

[40] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2003.

[41] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1430–1437, June 2009.

[42] R. Horaud and F. Dornaika. Hand-eye calibration. *International Journal of Robotics Research*, 14(3):195–210, June 1995.

[43] X. Huang, I. Walker, and S. Birchfield. Occlusion-aware reconstruction and manipulation of 3D articulated objects. In *Proceedings of the International Conference on Robotics and Automation*, May 2012.

[44] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th ACM Symposium on User Interface Software and Technology*, pages 559–568, 2011.

[45] D. Katz and O. Brock. Manipulating articulated objects with interactive perception. In *Proceedings of the International Conference on Robotics and Automation*, pages 272–277, May 2008.

[46] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz. Interactive segmentation, tracking, and kinematic modeling of unknown 3D articulated objects. In *Proceedings of the International Conference on Robotics and Automation*, May 2013.

[47] D. Katz, J. Kenney, and O. Brock. How can robots succeed in unstructured environments? In *Workshop on Robot Manipulation: Intelligence in Human Environments at Robotics: Science and Systems*, June 2008.

[48] D. Katz, A. Orthey, and O. Brock. Interactive perception of articulated objects. In *12th International Symposium on Experimental Robotics (ISER)*, Dec. 2010.

[49] S. Kean, J. Hall, and P. Perry. *Meet the Kinect: An Introduction to Programming Natural User Interfaces*. Apress, first edition, 2011.

[50] A. G. Kirk, J. F. O'Brien, and D. A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2005.

[51] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz. Real-time CAD model matching for mobile manipulation and grasping. In *IEEE-RAS International Conference on Humanoid Robots*, pages 290–296, Dec. 2009.

[52] E. Klingbeil, A. Saxena, and A. Y. Ng. Learning to open new doors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010.

[53] K. Kolev, M. Klodt, and T. B. andDaniel Cremers. Continuous global optimization in multiview 3D reconstruction. *International Journal of Computer Vision*, 84(1):80–96, Aug. 2009.

[54] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, July 2000.

[55] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, Feb. 1994.

[56] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[57] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *SIGGRAPH*, pages 367–374, 2000.

[58] N. Max. Hierarchical rendering of trees from precomputed multi-layer z-buffers. In *Proceedings of the Eurographics Rendering Workshop*, pages 165–174, 1996.

[59] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. Gerkey, and E. Berger. Autonomous door opening and plugging in with a personal robot. In *Proceedings of the International Conference on Robotics and Automation*, 2010.

[60] S. Miller, J. van den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *International Journal of Robotics Research (IJRR)*, 31(2):249–267, Feb. 2012.

[61] J. M. Morel and G. Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, Apr. 2009.

[62] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative O(n) solution to the PnP problem. In *Proceedings of the International Conference on Computer Vision*, Oct. 2007.

[63] J. F. O'Brien, J. Robert E. Bodenheimer, G. J. Brostow, and J. K. Hodgins. Automatic joint parameter estimation from magnetic motion capture data. In *Proceedings of Graphics Interface*, 2000.

[64] M. Paladini, A. D. Bue, M. Stošić, M. Dodig, J. Xavier, and L. Agapito. Factorization for non-rigid and articulated structure using metric projections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2898–2905, June 2009.

[65] C. Papazov, S. Haddadin, S. Parusel, K. Krieger, and D. Burschka. Rigid 3D geometry matching for grasping of known objects in cluttered scenes. *International Journal of Robotics Research*, 2012 (forthcoming).

[66] F. C. Park and B. J. Martin. Robot sensor calibration: solving ax=xb on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, Oct. 1994.

[67] H. S. Park and Y. Sheikh. 3D reconstruction of a smooth articulated trajectory from a monocular image sequence. In *Proceedings of the International Conference on Computer Vision*, pages 201–208, Nov. 2011.

[68] A. M. Petrina. Advances in robotics (review). *Automatic Documentation and Mathematical Linguistics*, 45(2):43–57, 2011.

[69] R. Pfeifer, M. Lungarella, , and F. Iida. The challenges ahead for bio-inspired 'soft' robotics. *Communications of the ACM*, 55(11):76–87, Nov. 2012.

[70] J. Pilet, V. Lepetit, and P. Fua. Fast non-rigid surface detection, registration and realistic augmentation. *International Journal of Computer Vision*, 76(2):109–122, Feb. 2008.

[71] C. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–218, 1997.

[72] J.-P. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 822–827, June 2005.

[73] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, 40(1):1–29, Oct. 1987.

[74] D. A. Ross, D. Tarlow, and R. S. Zemel. Learning articulated structure and motion. *International Journal of Computer Vision*, 88(2):214–237, Mar. 2010.

[75] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27:157–173, Feb. 2008.

[76] S. Seitz and C. Dyer. Complete scene structure from four point correspondences. In *Proceedings of the International Conference on Computer Vision*, pages 330–337, June 1995.

[77] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 519–528, June 2006.

[78] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1067–1073, June 1997.

[79] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Proceedings of A.C.M. SIGGRAPH*, pages 231–242, 1998.

[80] D. Sinclair, L. Paletta, and A. Pinz. Euclidean structure recovery through articulated motion. In *In Proc. 10th Scandinavian Conference on Image Analysis*, 1997.

[81] S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *Proceedings of the International Conference on Computer Vision*, pages 349–356, 2005.

[82] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *Workshop on Volume Graphics*, pages 81–101, 2001.

[83] G. G. Slabaugh, W. B. Culbertson, T. Malzbender, M. R. Stevens, and R. W. Schafer. Methods for volumetric reconstruction of visual scenes. *International Journal of Computer Vision*, 57(3):179–199, 2004.

[84] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 22(6):371–393, 2003.

[85] N. Snavely, 2006. Bundler: SfM for unordered image collections, http://phototour.cs.washington.edu/bundler.

[86] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3D. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 25(3):835–846, 2006.

[87] S. Srivastava and N. Ahuja. Octree generation from object silhouettes in perspective views. *Computer Vision, Graphics and Image Processing*, 49(1):68–84, Jan. 1990.

[88] K. H. Strobl and G. Hirzinger. Optimal hand-eye calibration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4647–4653, Oct. 2006.

[89] J. Sturm, A. Jain, C. Stachniss, C. Kemp, and W. Burgard. Operating articulated objects based on experience. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[90] J. Sturm, K. Konolige, C. Stachniss, and W. Burgard. Vision-based detection for learning articulation models of cabinet doors and drawers in household environments. In *Proceedings of the International Conference on Robotics and Automation*, 2010.

[91] J. Sturm, C. Stachniss, V. Pradeep, C. Plagemann, K. Konolige, and W. Burgard. Learning kinematic models for articulated objects. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2009.

[92] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[93] L. Torresani and C. Bregler. Space-time tracking. In *Proceedings of the European Conference on Computer Vision*, 2002.

[94] L. Torresani, A. Hertzmann, and C. Bregler. Learning non-rigid 3D shape from 2D motion. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[95] L. Torresani, D. Yang, G. Alexander, and C. Bregler. Tracking and modelling non-rigid objects with rank constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[96] S. Tran and L. Davis. 3D surface reconstruction using graph cuts with surface constraints. In *Proceedings of the European Conference on Computer Vision*, pages 219–231, May 2006.

[97] P. Tresadern and I. Reid. Articulated structure from motion by factorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1110–1115, 2005.

[98] E. Trucco and A. Verri. *Introductory Techniques for 3D Computer Vision*. Upper Saddle River, NJ: Prentice Hall, 1998.

[99] R. Y. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345–358, June 1989.

[100] C. A. Vanegas, D. G. Aliaga, and B. Beneš. Building reconstruction using Manhattan-world grammars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[101] G. Vogiatzis, C. Hernández, P. H. Torr, and R. Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246, Dec. 2007.

[102] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 391–398, June 2005.

[103] G. Walck and M. Drouin. Automatic observation for 3d reconstruction of unknown objects using visual servoing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010.

[104] C. Wengert. Camera Calibration and Hand to Eye Calibration Toolbox, http://www.vision.ee.ethz.ch/software/calibration_toolbox//calibration_toolbox.php.

[105] B. Willimon, S. Birchfield, and I. Walker. Rigid and non-rigid classification using interactive perception. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1728–1733, 2010.

[106] B. Willimon, S. Birchfield, and I. Walker. Classification of clothing using interactive perception. In *International Conf. on Robotics and Automation (ICRA)*, pages 1862–1868, 2011.

[107] B. Willimon, S. Birchfield, and I. Walker. Model for unfolding laundry using interactive perception. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[108] J. Xiao, J.-X. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. *International Journal of Computer Vision*, 67(2), 2006.

[109] J. Yan and M. Pollefeys. Automatic kinematic chain building from feature trajectories of articulated objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[110] J. Yan and M. Pollefeys. A factorization-based approach for articulated nonrigid shape, motion, and kinematic chain recovery from video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):865–877, May 2008.

[111] X. Zhang, Y. Liu, and T. S. Huang. Motion analysis of articulated objects from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):625–636, Apr. 2006.

[112] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.