Robot Rubik's Cube Solving: Pre-touch Sensing for Sequential Manipulation

Boling Yang

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2018

Committee:

Joshua R. Smith

Blake Hannaford

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Robot Rubik's Cube Solving:
Pre-touch Sensing for Sequential Manipulation

Boling Yang

Chair of the Supervisory Committee:
Professor Joshua R. Smith
Allen School of Computer Science and Engineering,
Department of Electrical Engineering

Robotics manipulation accuracy highly depends on the robot's belief about the relative pose between object and manipulator. While most robots achieve object pose estimation through computer vision via head-mounted cameras, robotics manipulation accuracy is limited by this perception method. A major limitation comes from our inability to perfectly calibrate both the cameras' parameters and the coordinate frames of robot arms. Moreover, when a task requires a robot to perform a long sequence of manipulation, manipulation errors could be accumulated a long time and eventually result in an irreversible failure to the task. The focus of this work is to examine how robots can achieve more robust sequential manipulation through the use of pre-touch sensors. The utility of close-range proximity sensing is evaluated through a robotic system that uses a new optical time-of-flight pre-touch sensor to complete a highly precise and sequential task - solving the Rubik's cube. The techniques used in this task are then extended to a more general framework in which ICP is used to match pre-touch data to a reference model, demonstrating that even simple pre-touch scans can be used to recover the pose of common objects that require sequential manipulation. Two pre-touch sensing region detection algorithms are introduced to detect object regions that contain distinctive geometric features. By focusing pre-touch sensing on these regions, the robot can more efficiently gather the information necessary to adjust its original pose estimate.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

AR TAGS: Fiducial markers that are detectable through computer vision. In this project, they are used to represent object pose and robot end effector pose.

END EFFECTOR: The component at the end of a robotic arm that is designed to interact with the environment.

IMAGE FEATURE: A quantity or vector of quantities that describes a potential region of interest in an image. In this project, it represents geometric variation of an object's surface.

PR2: A general purpose robotic research platform. The hardware and software in this project were designed for this platform.

POINT CLOUD: A collection of 3D points. In this project, point clouds are produced by a Microsoft Kinect camera or our optical time-of-flight pre-touch sensors.

ACKNOWLEDGMENTS

This work was done under the mentorship of Prof. Joshua R. Smith. I also want to note the helpful discussions and support provided by Patrick Lancaster from Allen School of Computer Science and Engineering, University of Washington.

Chapter 1

INTRODUCTION

As robots continue to transition from operating in controlled, carefully designed environments towards human-centric, unstructured ones, they will have to make more sophisticated use of sensing to cope with the inherent uncertainty in our real world. In particular, robust robot manipulation is difficult to achieve because of the uncertainty involved in manipulating an object [16]. For tasks that require sequential manipulations, the robot's belief about the pose of the object at any point in time can be corrupted by a poor characterization of the initial pose, or through the accumulation of error caused by controller noise, previous imperfect manipulations, and perceptual errors. Depending on the precision required, such errors can cause the robot to fail at the task of interest. We propose a novel sensing method – optical time-of-flight pre-touch sensing, which can effectively solve the problem of error accumulation during sequential manipulation. Pre-touch sensing is achieved by mounting proximity sensor on the end effector. This thesis will describe the design details of hardwares, softwares, and experiment for achieving precise object pose estimation via pre-touch sensing.

1.1 Background

In this section, optical time-of-flight pre-touch sensing is compared with perception methods from two major robotics perception categories, computer vision and tactile sensing, as well as with other existing pre-touch sensing methods.

Figure 1.1: This figure visualizes the calibration errors in a robotic system. The left picture shows the reality distance between the robot's end effector and the edge of a bowl, and the right picture shows what the robot believes.

1.1.1 Computer Vision

Computer vision (CV) based object localization has been widely explored from the robot manipulation perspective. Typically, a robot uses image input to localize important objects and evaluate object state, and then generates manipulation plans base on this knowledge. For example, a laundry folding robot is developed by Maitin-Shepard [22], where the robot finds the corner of a piece of laundry with a CV algorithm. In another example, Chang [3] successfully enables a robot to detect and separate objects in a cluttered environment. However, the manipulation accuracy is not emphasized by the previous two examples. Vahrenkamp [41] and Kragic [17] use visual servoing to achieve closed loop robotic arm control, which improves manipulation accuracy. Furthermore, the Eye-In-Hand Sensing methods by Leeper [19] and Kahn [15], improve manipulation by relocating the cameras closer to the wrist of a robot arm. The maneuverability of its arm allows the robot to explore the workspace and get a better observation perspective for its camera. However, all of these methods can not completely avoid the problem of mis-calibration of camera parameters and robot coordinate frames, which our work will address by integrating the sensor into the robot's manipulator.

1.1.2 Tactile Sensing

While computer vision is typically most useful for long range sensing, tactile sensors aid manipulation upon making contact with an object. Tactile sensors are typically attached to the robot's end-effector such that they are maneuverable, potentially allowing them to sense regions of an object that could not be imaged by a camera because of their relatively static nature. Li's GelSight tactile sensor [21] generates a height map, which allows the robot to perform fine-grained manipulation within its grasp, such as insertion of a USB connector. Petrovskaya and Khatib [29] designed the particle-based Scaling Series algorithm and applied it to tactile measurements to estimate object pose, allowing their robot to locate and grasp objects and manipulate a door handle. Furthermore, deep learning has also been applied to tactile sensing for object classification. Schmitz [37] uses a robot hand covered with a 241 element tactile sensor array to perform high reliability objects recognition with a deep neural network. Yet, contact based methods could lead to object displacement and even the change of object state such as object shape.

1.1.3 Pre-touch Sensing

Pre-touch sensors typically operate at a range intermediate to that of short-range tactile sensors and long-range vision-based sensors, endowing them with some of the same benefits that are achieved by both short range and long range sensing. Similar to tactile sensors, pre-touch sensors are mounted to the robot's end-effector, giving them robustness to the difficulties of sensing occluded parts of the environment. Compared to long range sensing, they operate in a shorter range but have the potential to measure more precisely. Moreover, pre-touch sensors do not have to make contact with a surface or object during the sensing process. This is an advantage over tactile sensors, because contact could unintentionally cause a change in the pose or state of the object.

Electric Field Pre-touch

Electric field sensing has been widely explored in the context of pre-touch sensing. Electric field sensing is typically achieved by transmitting an AC signal from one electrode to another. Objects near the sensor will alter the displacement current between the two electrodes, inducing a deviation from the baseline measurement (in which no objects are near the sensor) [38]. Electric field sensors are most adept at measuring conductive objects, but are unable to detect plastics, foams, or other objects with a dielectric constant similar to that of air. In [42], electric field sensing is used to localize an object and preshape the robot's fingers in order to achieve a stable grasp. Mayton et al. [25] extend electric field sensing to co-manipulation of objects between humans and robots. Finally, Mühlbacher-Karre [26] integrate electric field sensing into a robotic bar-tending system in order to determine the fill level of beverages.

Acoustic Pre-touch

More recently, acoustic sensing has been developed for pre-touch sensing. In [13], Jiang and Smith create a "seashell effect" sensor to localize objects that are typically difficult to sense with RGB-D cameras and other optical sensors, and then apply those measurements to grasping. The sensor consists primarily of a miniature metal pipe that has a microphone attached to one end. As objects approach the open end of the pipe, the effective resonant frequency of the pipe changes, which the microphone measures.

Optical Pre-touch

Optical pre-touch sensors emit infrared beams and estimate distance by measuring the infrared signal reflected by the object. Previously, a probabilistic optical pre-touch sensor model is created by Hsaio [12] and successfully achieves reactive object grasping. Guo [8] uses a break-beam proximity sensor to determine whether a target object is within the robot's griper. This method is robust to specular objects that present challenges for reflective optical sensors. Maldonado [23] applies the same laser sensing technology used in computer mice

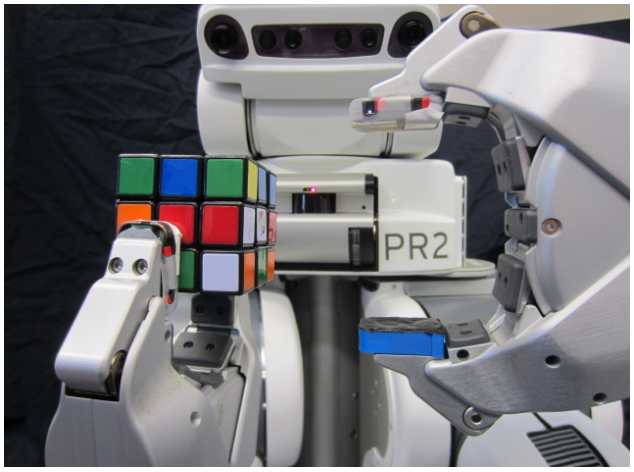

Figure 1.2: The robot is able to precisely manipulate the Rubik's cube using the equipped pre-touch sensors.

to reconstruct object surface shape, classify object surface types, and detect slip. All three of these works measure the amount of light reflected off of an object to either detect it or infer distance, whereas the sensor we developed uses time-of-flight technology to measure distance. Compared to [12], our sensor has a longer range with comparable accuracy, and does not need to be calibrated for the surface reflectivity of the object.

While previous work on pre-touch sensing focuses on estimating the pose of the surface local to the pre-touch sensors for the purpose of single-shot grasping, we are interested in full pose estimation of an object. Knowing the full pose becomes increasingly important when considering more complex manipulations, such as those that involve opening, twisting, or re-grasping an object. To the best of the authors' knowledge, our work is the first to estimate the full pose of an object by applying ICP (or any other similar algorithm) to a single, sparse pre-touch scan.

1.2 Project Overview

In this work, we show that manipulation errors can accumulate through out long sequential manipulation tasks, and how these errors can be effectively controlled by the type of pre-touch

sensor we developed. In order to show that robot manipulation accuracy can be improved by pre-touch sensing, we first demonstrate how pre-touch sensing enables a PR2 robot to solve a Rubik's cube more robustly compared to the baseline method. We then examine whether pre-touch sensing can be applied to other objects that may require sequential manipulation. We find that employing even a simple pre-touch scanning strategy allows the robot to infer the pose of a number of common objects. A good pose estimation can improve manipulation robustness, and the last problem we solve is how to get high quality pose estimation for common objects from pre-touch sensing. We develop and compare two algorithms that can be generalized to any pre-touch sensing scenarios. These algorithms detect object regions that contain rich and distinct geometric features, such that the robot can accurately inferred the object pose by scanning these regions with pre-touch sensor.

We chose to first tackle the task of solving the Rubik's cube because the challenges that general purpose robots face when attempting to achieve sequential manipulation are well represented by this task. Solving a Rubik's cube can require up to 20 rotations [35]. For each rotation, the robot may have to alter its grasp on the cube by transferring it from one hand to the other and/or shifting the grasp point(s). Because each face of a Rubik's cube contains sub-cubes of a dimension of only 1.9 cm, each manipulation may only be able to tolerate error on the order of millimeters. Even if the robot is able to complete a rotation at a given point in time, an imprecise grasp could cause future rotations to fail by invalidating the robot's belief of where the cube is with respect to the robot's gripper.

Ultimately, the authors aim to show that pre-touch sensing enables robots to accomplish sequential manipulations more effectively. We hypothesize that pre-touch scanning enables a robot to gain the geometric information necessary to estimate the pose of an object of interest, and thereby perform actions that are common in sequential manipulation, such as re-grasping. The specific contributions of this work are:

1. Introduction of a new type of pre-touch sensing based on optical time-of-flight measurements, and full integration of this sensor with the PR2 system.

2. The application and evaluation of pre-touch sensing for robot manipulation in the context of solving the Rubik's cube. To the best of the authors' knowledge, pre-touch has never previously been applied to solving the Rubik's cube. Furthermore, the authors argue that solving the Rubik's cube is the most complex, sequential task to which pre-touch sensing has ever been applied.

3. The use of an Iterative Closest Point (ICP) algorithm to align a 1D pre-touch scan to a pre-existing reference point cloud in order to estimate object pose.

4. A comparison of the performance of optical time-of-flight pre-touch (mounted in one finger) with our prior electric field pre-touch sensor (mounted in another finger), and the finding that the two sensing modalities are complementary.

5. Introduction of two pre-touch area detection algorithms that use deep neural network and Normal Aligned Radial Feature. These algorithms allows robot to use pre-touch sensing to estimate high quality pose for any objects.

1.3 Associated Publications

Some material from this thesis has previously been published in two earlier conference papers[45][18]. In addition to me, the author list for both papers included Patrick Lancaster and Joshua Smith. I was the first author of [45] and Patrick was the first author of [18]. Chapter 2, Chapter 3, and Chapter 4 in this thesis focus on content from [45] – "Pre-touch Sensing for Sequential Manipulation", which was published in the 2017 International Conference on Robotics and Automation. In that paper, my main contribution was developing the software that enables a PR2 robot to perform robust sequential manipulation of a Rubik's cube and rough object pose estimation using pre-touch sensors. The sensor hardware was designed by Patrick Lancaster. The second paper[18] – "Improved Object Pose Estimation via Deep Pre-touch Sensing" was published in the 2017 International Conference on Intelligent Robots. That paper presented a self-supervised training framework

to train a pre-touch region proposal neural network. However, that training framework is not the focus of this thesis. Instead, since I mainly contributed by designing an alternative method (NARF pre-touch region detection), this thesis focuses on explaining that method in Chapter 5.

Chapter 2

SENSOR HARDWARE

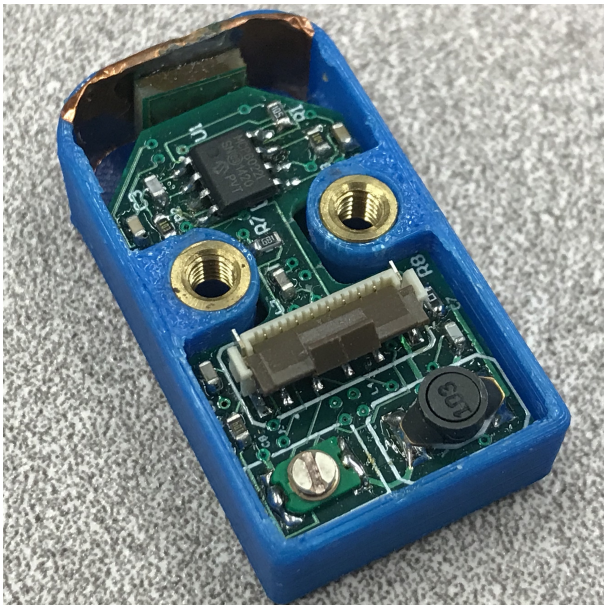

Figure 2.1: Electric field pre-touch sensor for PR2 robot.

This work explores the use of two different pre-touch sensors in aiding sequential manipulation. One is an electric field sensor that was adapted from the work of [25]. The other is a new optical time-of-flight sensor, which the following paragraphs of this section will describe. Both sensors' casings were designed to match the form factor of the PR2's fingertip. On each of the robot's parallel jaw grippers, an optical sensor was attached to the left fingertip, while an electric field sensor was mounted to the right fingertip. Only the optical sensor was used when attempting to solve the Rubik's cube because the electric field sensor is not able to sense the plastic Rubik's cube effectively.

The optical pre-touch sensor measures the distance to a surface using the VL6180x optical time-of-flight sensor [39] module created by ST MicroElectronics. It is capable of measuring the distance to an object with millimeter accuracy at a range of 1cm to 10cm. Furthermore, some objects (particularly those with light colors) can be detected out to a range of 25.5cm. Fig. 2.2 demonstrates the sensor's performance over various colored targets.

The sensor supports up to six VL6180x sensing modules, where one can be placed at

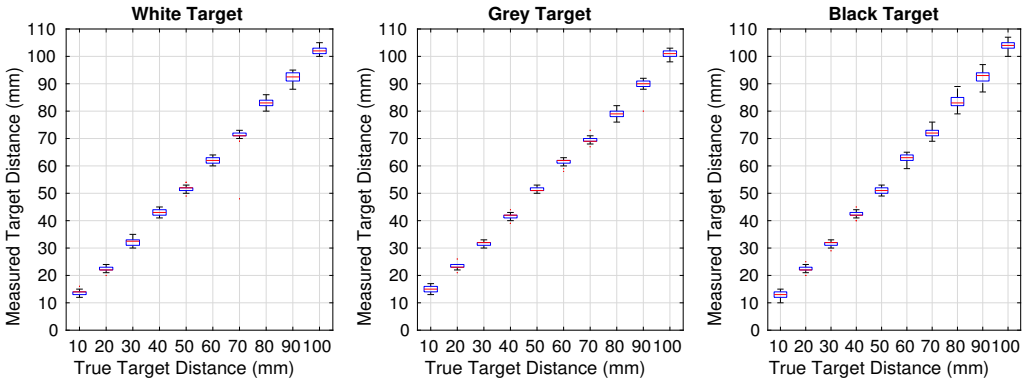

Figure 2.2: Boxplots of sensor measurements over the specified range for white, grey, and black target objects. Each box consists of 30 measurements.

the tip of the finger, two on each side, and one in the pad of the finger. The location of each sensing module is labeled in Fig. 2.3, and will be referenced as such when referring to a specific sensor module throughout the rest of this work. The sensor's design consists of a main board (29x16.5mm) and a secondary board (8.25x5.75mm). The main board hosts an ATMega168PA microcontroller that communicates with each of the VL6180x sensing modules over I2C. The sensor module on the pad of the fingertip sensor is directly soldered to the main board. Every other module is connected to the main board through 1mm pitch headers that interface between the main board and secondary boards.

The robot is able to collect data from each of the six sensing modules at a rate of 30hz. Measurements are passed from the sensor's microcontroller to the robot through an SPI interface built into the gripper and then published to a ROS topic. Two metal screw inserts are pressed into the sensor's casing so that it can be fastened to the robot's fingertip.

The components required to build one of these optical sensors cost less than $100. The circuit schematics, PCB files, sensor casing CAD files, and firmware are publicly available at https://bitbucket.org/planc509/optical-distance.

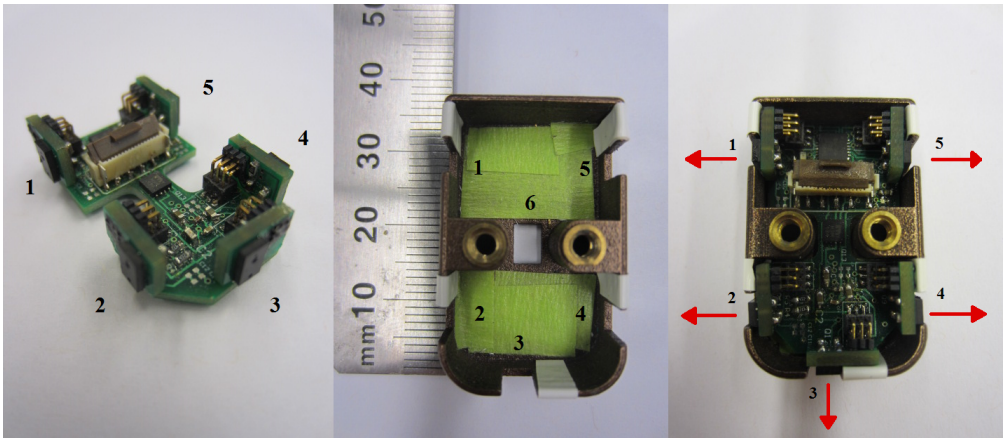

Figure 2.3: Left: The printed circuit boards that compose the sensor. Middle: The 3D printed sensor casing. The hole in the middle of the case is for a sensing module soldered to the bottom of the main PCB. Right: The assembled sensor with arrows denoting the directions of five out of six sensor modules' infrared beams.

Chapter 3

SOLVING THE RUBIK'S CUBE

The implementation of the software that enables a PR2 robot to solve a Rubik's cube is discussed in this chapter. Throughout this work, the robot uses simple scanning strategies to estimate the pose of a Rubik's cube. The simplicity of these scans suggests that they are applicable to a wide range of objects, detail generalization methods are discussed in Chapter 4 and Chapter 5. The following sections will detail the scanning strategies used and their effectiveness. In the remainder of this text, unless explicitly said otherwise, any coordinate references are with respect to the coordinate frame of whichever gripper is currently holding the object. That is, the y-axis extends along the direction that the gripper opens and closes, the x-axis extends out along the direction that the robot's fingertips point, and the z-axis is orthogonal to both the x and y-axes using a right-handed coordinate system.

Solving the Rubik's cube has been previously suggested as a benchmark for robot manipulation [2], [48]. Beyond this suggestion, [48] and [34] develop robot systems for solving the Rubik's cube. In [48], the robot's grippers are somewhat specialized; their trough shape is used to grasp the cube at the corners and thereby reduce the uncertainty in the cube's state. The system in [34] was based on a PR2 robot without task specific modification of the grippers. They demonstrated that their system can successfully solve a basic Rubik's cube puzzle that required six rotations, which matches the performance of our baseline system that does not use pre-touch sensing for manipulation. However, there is no further information to indicate that this system is capable of robustly solving Rubik's cubes that require a larger number of rotations.

Our system enables a PR2 robot to solve the Rubik's cube from any initial configuration. A head-mounted camera is used to detect the initial color state of the cube. Note that the

only modification to the robot is the addition of the pre-touch sensors; no modifications were made to physically constrain the cube to a certain position within the robot's gripper. The goal of the puzzle is to rotate the faces of the cube so that the nine squares of each face are all of the same color. The robot picks up the cube and inspects the faces using the Kinect camera to estimate the initial cube condition (i.e the color of each square).

3.1 Representation in Robotics Motion Planning

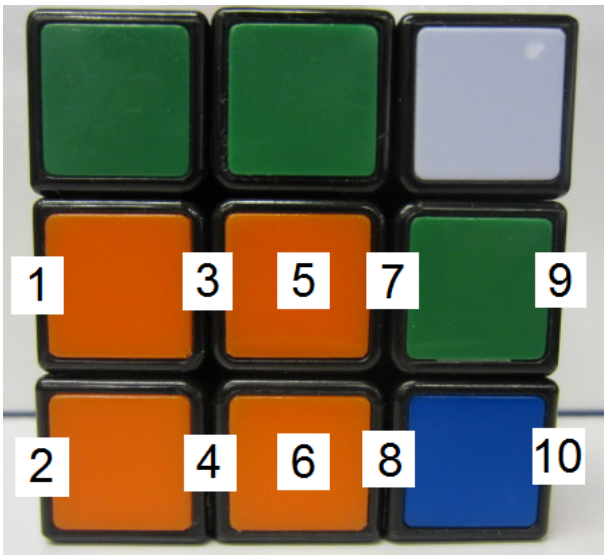

Figure 3.1: The proposed system will use ten unique grasp points. Assuming the gripper is approaching from the bottom of the cube, each white box represents a possible grasp point.

The space of the general grasp planning problem is surprisingly large. There are up to 120 feasible grasp points for one hand on a cube: each hand can be placed in one of 6 headings, and in one of two wrist roll states, which yields 6x2 = 12 hand approach orientations. For each of these, there are 10 valid grasp points, as shown in Fig. 3.1. This yields a total of 12x10 = 120 single handed grasps. When the second hand is considered, there are no more than 5x2x10 = 100 grasp points. (The second hand must have a different approach orientation than the first hand, so there are 5x2 rather than 6x2 hand approach orientations.) Thus the total number of two handed grasps is no more than 120x100 = 12,000. (In actuality, kinematic constraints and additional hand collisions reduce the set of feasible two-handed grasps further.)

Most generally, one can imagine wanting to transition from any one of the initial two-handed grasp states to any other two-handed grasp state. Making such a transition may

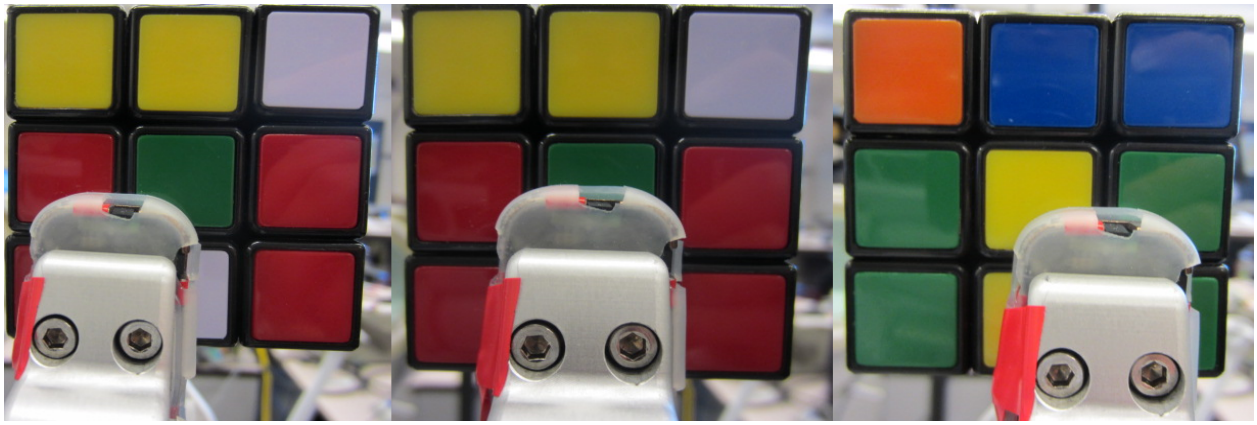

Figure 3.2: Three grasp points used by the current system. These correspond to the grasp points labeled 3, 5, and 7 in 3.1.

require a motion plan such as: open gripper 1, re-position it, close gripper 1 at a new location, open gripper 2, reposition it, close gripper 2.

In our initial working system, we introduce a number of constraints to simplify the motion planning. We use one gripper to fix the Rubik's cube while the other rotates one of the faces. In some cases, the cube must be transferred from one gripper to another in order to rotate a certain face. Re-grasping can also be necessary in order to shift the grip position with respect to the cube. Two 'home' poses were defined to allow the robot to return to a kinematically unconstrained position before attempting each rotation. All the arm motions and manipulation on the cube comply with the following constraints:

1. After performing a cube face rotation, the grippers return to the home pose.

2. During a cube rotation, one of the grippers holds two rows of the cube stationary and the other rotates one face of the cube. The cube holding gripper always constrains two layers of the cube and the manipulating gripper rotates only one layer. Each finger of the holding gripper has three feasible grasp points as shown in Fig. 3.2. There is only one allowed grasp point for the rotating gripper: the mid-point of the face, which allows the face to rotate with a very simple wrist rotation.

3. The cube's orientation is kept consistent with respect to the holding gripper.

4. The front, back, and right faces can only be rotated when the left gripper is holding the cube. The up, down, and left faces can only be rotated when the right gripper is holding the cube.

These constraints effectively simplify the motion planning problem enough that it can be handled by a finite state machine. The system only considers a subset of the grasp points shown in Fig. 3.1. A robot pose is defined by which gripper holds the cube, either left or right gripper, and the grasp point where the holding gripper grasps on. A Rubik's cube has 6 faces, and each of them can be turned by either 90 degrees clockwise, 90 degrees counter-clockwise, or 180 degrees. Therefore, the action space of a Rubik's cube contain 18 actions. The finite state machine takes the current robot pose and the next action as inputs, and output the next pose for robot.

3.2 The Two-Phase Algorithm

Given the initial state of the Rubik's cube, the robot generates a solution using Kociemba's Two Phase algorithm[35][34]. To solve the Rubik's cube, the algorithm first uses an iterative deepening $A*$ algorithm to search for maneuvers that transform the scrambled cube to a state that is an element of a particular group G with special properties that correspond to a partial solution with corners and edges in the right locations but possibly with incorrect orientation. In phase 2, the 8 corners and all edges will be set to the correct orientations. The algorithm searches to find a solution that requires a minimal number of cube face rotations. It is known that any cube can be solved with at most 20 moves.[35]

3.3 Frames and Transformation

In our robotic system, as shown in figure 3.3, a frame represents a right-handed coordinate system in three dimensions. The geometric relation between two frames can be represented by a transformation matrix. Any object pose can be transformed from one reference frame

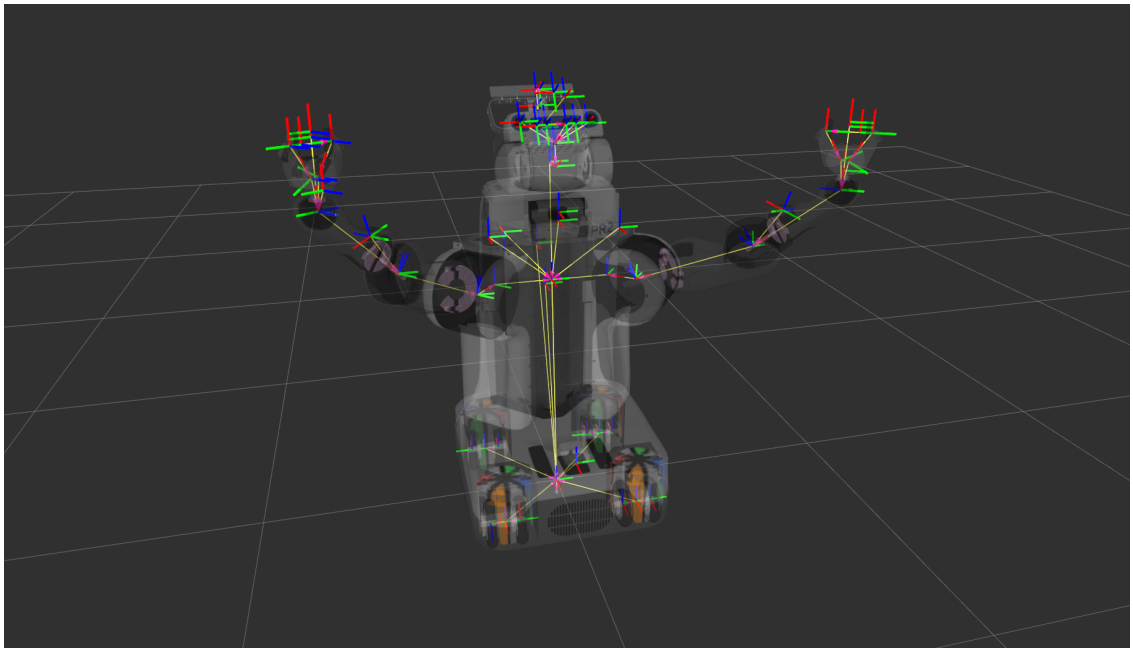

Figure 3.3: Visualization of PR2 robot's body frames and links.

to another using a transformation matrix. As the robot manipulates the Rubik's cube, its gripper frames are constantly changing with respect to the robot's base link. The estimated cube pose is solely determine by the coordinate frames of the robot's arms and hands. As the gripper moves, the cube's pose is updated under the assumption that the gripper and cube are rigidly connected. All of the transformation between coordinate frames for our PR2 robot are done by the tf library by Foote [6].

3.4 Implementation On PR2 Robot

In this work, pre-touch sensing is used to estimate the pose of the cube not only to minimize positional error in the next grasp, but also to correct any error that does occur from one manipulation to the next. As the robot solves the Rubik's cube, it will need to transfer it from one hand to the other, as well as change how it is grasping it. Before each re-grasp, the robot uses pre-touch scanning to refine its estimate of the pose of the cube with respect to the coordinate frame of the gripper that is currently holding it. The robot assumes that the cube

is oriented such that its upper and lower faces are approximately parallel to the ground. This assumption is not always true, but works well in practice. Furthermore, the robot already has a good approximation of the center of the cube's position in the y direction and its rotation around the y-axis because the cube is held between the robot's fingers. However, due to errors in previous re-grasps, the cube could have shifted unexpectedly along the x and/or z directions. There are many pre-touch scanning strategies that could be used to estimate the position along these two directions, but our method used the following strategy in order to minimize the amount of actuation required:

1. The gripper that is not holding the cube is opened if it is not already open.

2. The optical pre-touch sensor on that gripper's fingertip is then oriented such that the beam of sensing module 3 is normal to one of the faces of the cube that is normal to the xz-plane, such as in Fig. 1.2. This gripper is positioned such that the beam is not yet broken by the cube.

3. The gripper begins to close, causing the sensor to move in the y-direction. As the gripper closes, any significant change in the sensed distance indicates the position of the edge of the cube, allowing the robot to infer the cube's position along one of the uncertain axes.

4. Once the gripper has finished closing, the robot uses the sensor's distance measurements at the current position to estimate the position of the cube along the remaining uncertain axis.

This pre-touch scanning strategy was integrated into a baseline system that uses a computer vision module to recognize the colors of the cube faces, the two phase algorithm [35] [34] to determine the necessary cube rotations, and a finite-state machine based motion planner to execute the trajectories necessary to solve the cube.

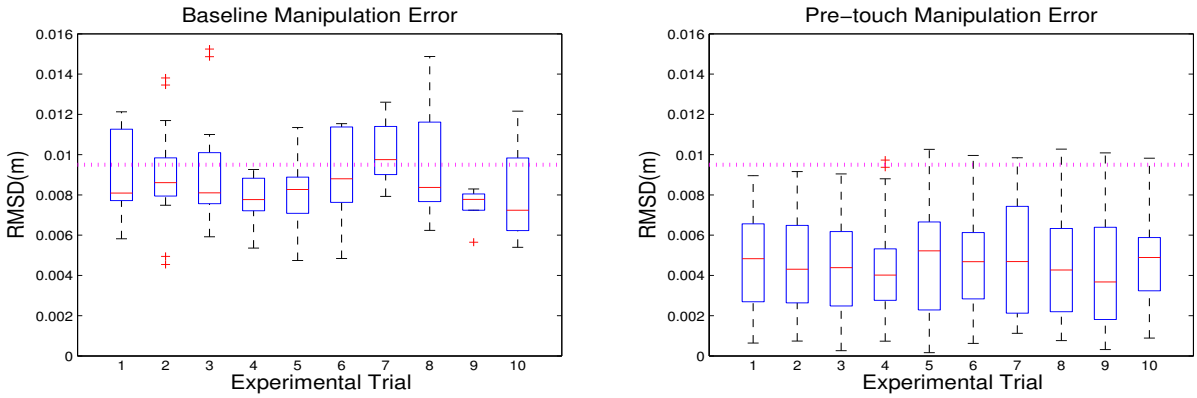

Figure 3.4: Box plots plots of positional error for the baseline (left), and corrected pre-touch (right) methods. Each box corresponds to one of the 10 trials and consists of all cube pose RMSD errors observed during that trial. The RMSD error is recorded prior to each re-grasp. The horizontal line across each plot denotes half of the dimension of a sub-cube, demonstrating that the increased dexterity provided by pre-touch sensing is significant for this task.

3.5 Experiment and Result

In order to determine the effectiveness of pre-touch scanning in the context of Rubik's cube solving, a system (as briefly described at the end of Section 2.2) for manipulating the cube was created in which pre-touch sensing could be enabled or disabled. When pre-touch is disabled, the system serves as a baseline for what is achievable without pre-touch sensing. Instead of scanning the cube after each re-grasp, the baseline system just assumes that the robot re-grasped the cube in the exact desired location.

3.5.1 Experimental Setup

We generated 10 random cube configurations that required between 20 to 23 rotations for the system to solve, and had both the baseline and pre-touch enabled versions of the system attempt them. In addition to reporting the success/failure rate, the robot's estimate of the cube's position throughout each trial is examined for both methods. Prior to each re-grasp, the robot's estimate of the cube position was recorded. All pose estimates were transformed

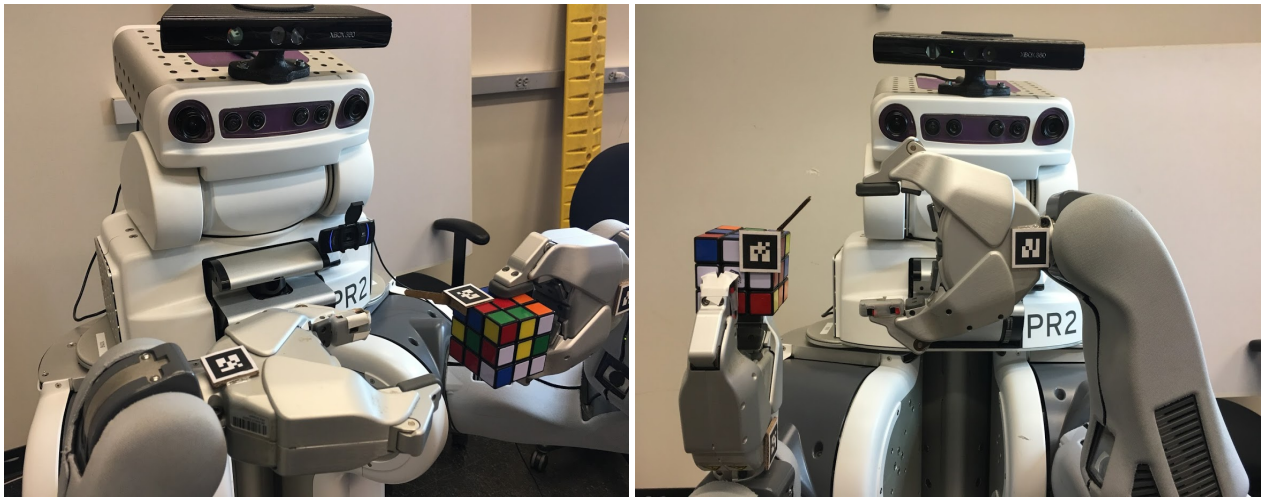

Figure 3.5: Two example instances when the ground truth data are being measured. The AR tag on the Rubik's cube has a frame which is designed to guarantee this AR tag represents the center of a corner cubelet. This AR tag is removed before the robot executes an action, and is reattached to the cube after the motion is finished for ground truth cube pose measurement.

into the frame of the gripper currently grasping the cube. In order to get ground truth measurements of the cube's position, an AR tag was attached to each face of the cube. We then use cameras external to the robot to detect and estimate the position of the cube when appropriate. Note that these cameras were calibrated, in order to provide the best possible ground truth. An AR tag was also added to each of the robot's grippers at a fixed distance away from that gripper's coordinate frame. This allowed us to compute the pre-touch enabled pose estimate (and corresponding ground-truth) of the cube without using the robot's coordinate transforms, as show in figure 3.5. Although our robot was re-calibrated prior to beginning this work, there was still significant error in the coordinate transforms (as there would be for any calibration of a high degree of freedom robot). Despite the use of AR tag detection as a 'ground truth' estimate of the pose of the cube, we are not implying that this method is better than pre-touch sensing for pose estimation. This method will have its own errors depending on how well the tag is detected, and has the disadvantage of requiring one or more tags to be placed on any object whose pose is to be estimated.

3.5.2 End-to-end Results

The experiment demonstrated that the robot's ability to solve the Rubik's cube was significantly enhanced by pre-touch sensing, as shown in Table 3.1. Using the pre-touch enabled method, the robot successfully solved 8 out of 10 cube configurations. As for the two failure trials, the robot finished 14 and 19 rotations out of 21 and 20 total required rotations before failing to complete a rotation. On the other hand, the baseline method did not successfully solve any of the puzzles. The maximum number of successful rotations for any of the 10 trials was 17, the minimum was 3, and the average was 9.6. All of the unsuccessful rotations occurred when the robot failed to re-grasp the cube; either as it tried to transfer the cube from one gripper to the other, or as it attempted to grasp the cube in order to rotate a face. These results demonstrate that although re-grasping motions are very sensitive to positional error, pre-touch sensing allows the robot to effectively compensate for them.

Table 3.1: End-to-end Rubik's Cube Solving

Method \ Result	Success	Fail	Avg. Rotations Completed
Baseline	0	10	9.6
Pre-touch	8	2	20.1

3.5.3 Intermediate Pose Estimation

Comparing the robot's positional estimate of the Rubik's cube for both methods to the ground truth values also yields interesting results. The robot re-estimated the pose of the cube prior to each re-grasp. For each pose estimate, the error was calculated as the root-mean-square deviation (RMSD) between the x and z axes of the estimate and the recorded ground-truth. The RMSD throughout each trial for both methods are summarized as box plots in Fig. 3.4.

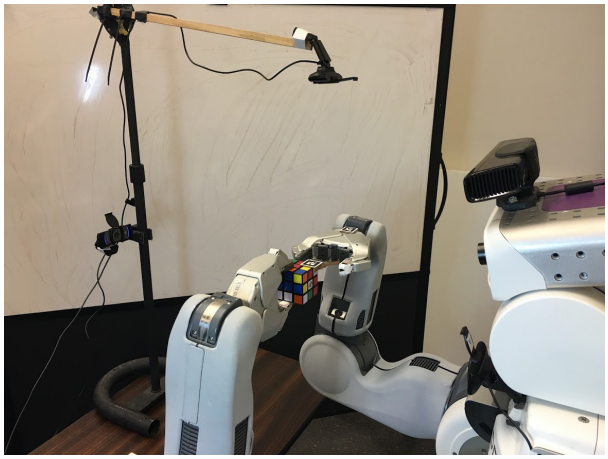

Figure 3.6: Cameras setup for ground truth data measurements. Three carefully calibrated cameras are used to detect the pose of AR tags from various perspectives. The first camera is attached on the left side of the robot's chest facing forward. The second camera is placed in front of the robot, facing toward the robot. And the last camera is mounted on a pole on top of the robot, facing downward.

The observed errors demonstrate that pre-touch sensing allowed the robot to significantly reduce the amount of error in the robot's estimate of the pose of the Rubik's cube. The robot has a finger width that is approximately equal to the size of a sub-cube of the Rubik's cube. Given that the desired grasp point is exactly in-between two sub-cubes, the margin of acceptable error is approximately half the width of a sub-cube. More error than this could cause the robot to unintentionally constrain one of the faces, causing a future manipulation to fail. Alternatively, error above this threshold could cause the robot to fail to even grasp the cube. The left plot of Fig. 3.4 demonstrates that for most trials, a large portion of the baseline method's pose estimates had an RMSD significantly larger than half of the size of a sub-cube. In contrast, the right plot shows that the RMSD of pre-touch aided estimates very rarely went above this threshold. In fact, for the pre-touch enabled method, most pose estimates had an RMSD of less than 0.8cm. Thus, through the use of pre-touch scanning, the robot has been able to limit the error in its estimate of the cube's pose and thereby solve it more robustly.

Chapter 4

PRE-TOUCH SCANNING FOR COMMON OBJECTS

The purpose of this chapter is to examine whether pre-touch scanning can be applied to objects with more complex geometry. We aim to demonstrate that a simple 1D scan of an arbitrary object can contain enough distinctive features to estimate its pose when matched to a reference model. This estimate could be useful when initially picking up an object, or before performing a re-grasp. After showing that the concept of using pre-touch scans to estimate object pose is sufficient, we automate the scanning areas detection in Chapter 4.

In this section, a single trajectory was chosen by the experimenter for each object that was likely to capture distinctive features. Each of the chosen trajectories consisted of the scanning gripper moving in a straight line with a fixed orientation. While executing each trajectory, the robot sampled the object at discrete points along it, where the interval between the sampling points was determined by the size of the object such that approximately 50 samples were collected. The process of obtaining a sample was slightly different depending on which pre-touch sensor was being utilized.

4.1 Data Points Sampling Strategies

When using the electric field pre-touch sensor to scan the object, the robot oriented the front of its fingertip orthogonal to the trajectory and towards the object. The object affects the sensor's measurements by shunting displacement current away from the electrode located at the front of the fingertip sensor. At each sample point, the robot moves its gripper towards or away from the object, causing the amount of current shunted away, and therefore the change from the baseline measurement (i.e. the measurement when the object is far way from the sensor), to change. The robot obtains a distance measurement by servoing its

gripper in this fashion until the change from the baseline measurement is sufficiently close to a pre-determined threshold. This threshold, which was recorded before beginning the scan and is different for each object, indicates when the fingertip is 1.5cm away from the object. This technique makes the assumption that the volume of the object local to the fingertip is uniform throughout the trajectory. Although this assumption is usually violated, in practice a satisfactory point cloud can often still be obtained with this method, as will be shown in Chapter 3.

When the robot uses the optical pre-touch sensor to scan, it again orients the front of its fingertip orthogonal to the trajectory and towards the object. At each sample point along the trajectory, the robot uses the sensor module at the tip of its finger to measure the distance to the object. Unlike the electric field sensor, the only actuation required is movement along the trajectory because the distance to the object at each sample point is directly reported by the optical sensor.

4.2 Iterative Closest Point Algorithm

After a point cloud model is sampled from a pre-touch sensor, object pose is estimated by matching the sampled point cloud with a reference object point cloud using the iterative closest point algorithm(ICP)[5]. Compare to other rigid 3D point cloud registration algorithms, like Singular Value Decomposition (SVD) [24] and Principal Component Analysis (PCA) based registration [44], ICP usually yields smaller matching errors after enough iterations[1]. The ICP algorithm is designed to minimize the distance between two point clouds by iteratively applying translations and rotations to the first point cloud. Typically the first(source) point cloud is obtained from real world observations and the second point cloud is a reference point cloud. The algorithm manipulates the source point cloud with the following steps:

1. Find the closest point from the reference point cloud for every point in the source point cloud.

2. Align the two point clouds by minimizing the root mean square point-to-point distance

across each matched point pairs.

3. Repeat steps 1 and 2 until the alignment error reaches a local minimum.

In our system, this process is done by using the *IterativeClosestPoint* function in Point Cloud Library[36]. This function takes in the source and reference cloud as inputs. The function will output the transformation matrix that best aligns the two point clouds and a fitness score to describe how well the two point clouds are aligned. Beyond PCL's default implementation of ICP, we also use a correspondence rejector that limits the influence of outliers [30]. In this problem, ICP registers a sampled partial point cloud of the target object with a full reference object point cloud. Similar scenarios have been discussed on previous ICP research, [47] [46] [43], and we will experiment these methods on our problem in the future.

4.3 Experiment and Result

While pre-touch sensing was found to be very applicable to manipulation of the Rubik's cube, the following experiment examines how simple pre-touch scanning can be used in more general manipulation tasks. Here, we estimate the pose of an object by using ICP to match a single, simple pre-touch scan to a corresponding reference model. A future manipulation system could then utilize the transform between the pre-touch scan and reference model to compute the object's pose. Such an estimation may be useful for manipulation when the object is initially grasped, and/or as the object is re-grasped throughout the manipulation task.

Experimental Setup

We evaluate the use of pre-touch scanning on seven common objects that could require sequential manipulation: a metal bowl, banana, lemon, coffee can, hammer, bell pepper, and a glass soda bottle. Each object is separately scanned by an electric field pre-touch

sensor and an optical pre-touch sensor as described in Chapter 2, section 3. Each sensor has an operating regime in which it is most effective. In particular, the optical pre-touch sensor can sense non-transparent objects well, while the electric field sensor is limited to sensing objects with a dielectric constant significantly different from that of air. Furthermore, the optical sensor has a long, narrow sensing region while the electric field has a short, wide sensing region. It is conceivable that a future system could combine measurements from both sensors to obtain more accurate estimates over a wider set of objects than either sensor alone. For now, we explore each sensor's individual ability to provide geometrically discriminative features for pose estimation through pre-touch scanning.

ICP Matching Results

After obtaining a pre-touch scan of an object, we estimate its pose by matching the scan to a point cloud reference model. We used Kinect Fusion [28] to create each reference model, and the PCL library's ICP algorithm [11] to do the matching. The results of each scan are shown in Fig.4.1, the first column shows each of the seven objects and the region that was scanned in green. The second column shows the raw point cloud obtained by the electric field sensor for each object, and the third column shows how the ICP algorithm matched the raw point cloud to the reference model. The raw point cloud is colored green if the ICP algorithm found a correct partial or full match, and red if it failed. The fourth and fifth columns display the analogous results for the optical pre-touch scans. The fitness scores (where lower scores indicate better matches) are given in Table 4.1. However, fitness score is not always a clear indicator of successful matching. Ultimately, we qualitatively specify the result of the matching by examining if the pre-touch scan matches to the correct region of the object.

The electric field sensor performed particularly well on two objects - the bowl and the hammer - most likely because they are made of metal. In fact, for these objects, the electric field sensor outperformed the optical pre-touch sensor, which failed to even find a match for the hammer. Although the optical sensor's raw point cloud of the hammer captures its

Table 4.1: ICP Fitness Score

Sensor \ Object	1	2	3	4	5	6	7
E-field(10^{-6})	6	89	7	17	9	15	24
Optical(10^{-6})	23	24	6	24	286	33	210

general shape, we believe that a number of poor samples at the claw end of the hammer caused the matching to fail. These bad readings could have been caused by the reflectivity of the hammer head, the more complex geometry at the claw end, or a combination of the two. Although the coffee can is also made of metal, the electric field sensor did not perform as well, potentially due to the internal coffee powder not being uniformly distributed throughout the can. Visually, the raw point clouds of the fruits created by the optical pre-touch sensor capture the profile of the corresponding fruit. However, the scan of the banana did not correctly match its reference model, most likely due to a failure to capture distinctive features in the pre-touch scan. The electric field sensor failed to get a distinguishable outline of the bell pepper, potentially due to the bell pepper being mostly hollow inside. Its raw scans of the other fruits capture the general shape of both the lemon and the banana, albeit with less fidelity than the optical sensor.

When scanning the soda bottle, the electric field sensor captured the shape of the body and the gradual transition from the body to the neck well. The matching gives a very rough estimate of the pose of the bottle. Note that the reference model was obtained by wrapping the bottle in opaque tape because transparent objects are difficult for the Kinect to sense. Accordingly, the optical pre-touch sensor did very poorly when scanning the uncovered bottle.

This experiment has demonstrated that for many objects, a single, simple scan is sufficient to get a good estimate of the pose of the object. Furthermore, only the electric field scan of the bell pepper and the optical scan of the bottle completely failed to retrieve any geometric

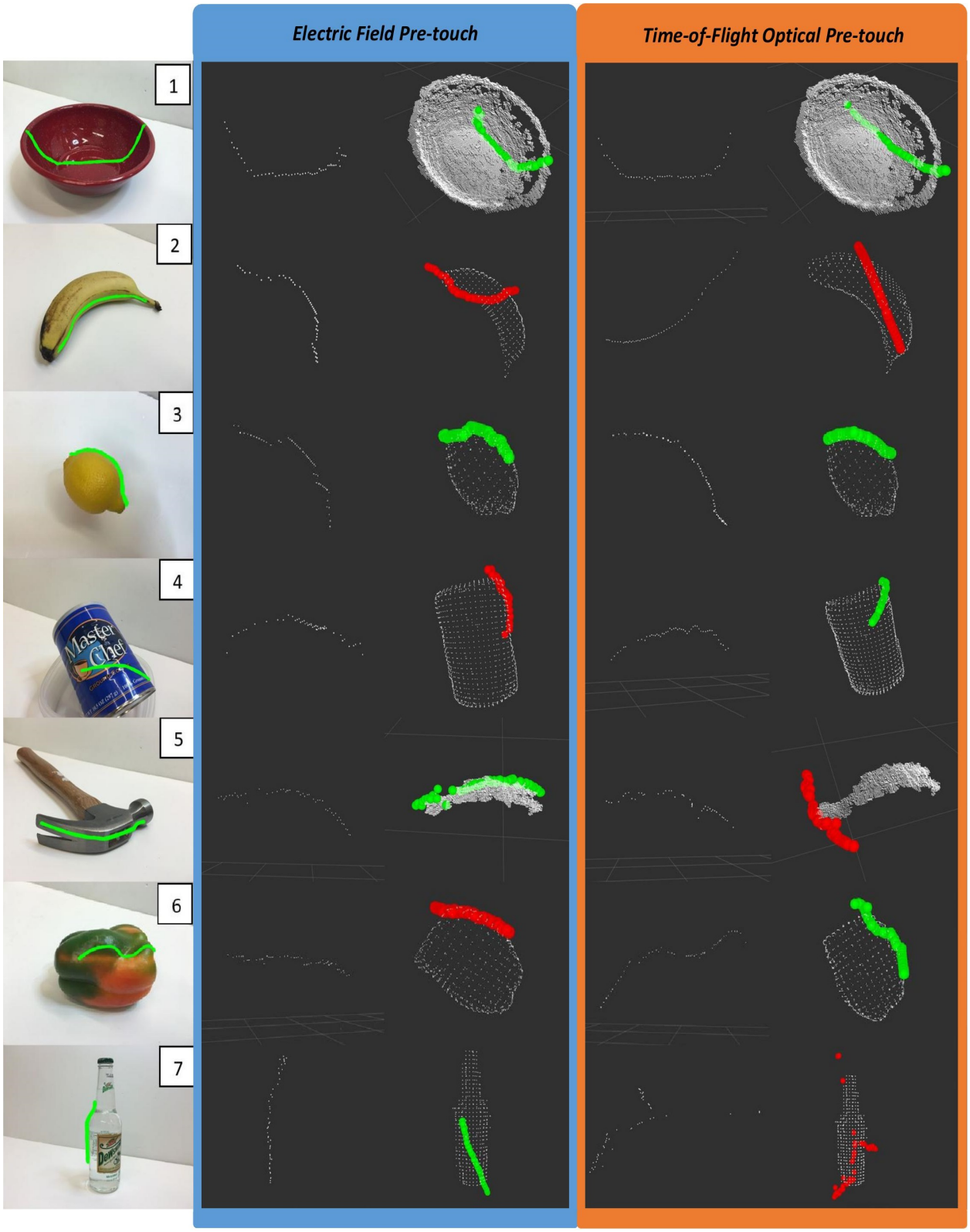

Figure 4.1: The results of applying pre-touch scanning and ICP to seven common objects.

information. This is encouraging because when one sensor completely failed, the other was able to capture at least some geometric information. Furthermore, it is possible that a method specifically designed to match these types of 1D scans to reference models would produce even better results than off-the-shelf ICP. It is therefore feasible that a system employing multiple scans with both of the sensors could have a robust ability to recover the pose of a wide range of objects.

Chapter 5

PRE-TOUCH SCAN REGION DETECTION

In the previous chapters, we demonstrated that through the use of optical time-of-flight pre-touch sensing, robot manipulation robustness is significantly improved in the context of a PR2 robot solving a Rubik's cube. Then we examined the application of pre-touch sensing to manipulation in the context of objects beyond just the Rubik's cube, that leading to the conclusion that this strategy is viable for a number of common objects. By focusing pre-touch sensing on regions that contain distinct geometric information, the robot can reestimate the object pose without scanning the whole object. However, in the experiment on Chapter 3, Section 2, these regions were manually selected based on human intuition. Such method becomes insufficient when human labels are not available. Furthermore, humans' intuition on these regions is neither alway correct nor necessarily optimal from an ICP perspective. Up to this point, there is still one unsolved question toward generalizing pre-touch sensing for object pose estimation. Namely, how can we autonomously generate pre-touch scanning trajectories for unseen objects.

In this chapter, we approach this problem by developing two algorithms that propose pre-touch sensing areas. In Chapter 3 Section 2, the experimenter determined the trajectory that captures the most diverse surface geometry on each object. The first algorithm aims to imitate such selection policy using Normal Aligned Radial Features(NARF)[40]. The second algorithm trains a deep neural network to detect object regions that contain distinctive geometric features. In both approaches, the pre-touch sensing regions are parameterized as rectangles in the image plane, and the robot collects pre-touch data by scanning along the perimeter of these rectangles. Note that the author's main effort was on developing the

first algorithm, and the second approach was primarily developed by Patrick Lancaster [1]. Therefore, the design details of the second approach will not be emphasized in this chapter.

5.1 Region Specification

Rotated rectangles represent a flexible, yet simple way to specify regions. Jiang et al. [14] first proposed using this representation for robot grasping, and many works have adopted it [20][31][32]. In this work, a specified rectangle represents a region at which to apply pre-touch sensing. Specifically, we perform the pre-touch scan along the perimeter of the region. Each rectangle can be parameterized as follows:

$$[x, y, w, h, \theta]$$

where x and y are the center of the rectangle in pixels, w and h are the width and height of the rectangle, and θ is the rotation around the center of the rectangle in the image plane. Following [32], in practice we use *cosine* and *sine* of twice the angle to represent the rotation due to the symmetry of the rectangle.

5.2 NARF-Based Region Proposal

In object recognition and mapping tasks, it is important to extract areas that can be recognized and matched from multiple sets of sensor readings. Therefore, detecting areas that have distinctive features is one of the most important challenges of these problems. The NARF algorithm is designed to extract interest points from a single range image and calculate feature descriptors. Each interest point corresponds to a feature descriptor, which is calculated according to the geometric information of the surface area around the interest point. The descriptor of an interest point is stable and unique even when being observed from multiple different perspectives.

---

[1] Ph.D. Candidate of Allen School of Computer Science & Engineering, University of Washington

5.2.1 Normal Aligned Radial Feature

The NARF algorithm[40] has three major steps: border extraction, interest point extraction, and descriptor calculation. The goal of the border extraction process is to extract points that belong to the object borders while simultaneously eliminating the veil points and points that belongs to shadow borders. Shadow borders are the points that on the background instead of the object, and veil points are interpolated points between the background borders and the object borders. Since we are only interested in identifying object borders, the algorithm operates in the following manner to extract object border points from a range image:

1. Calculate the euclidean distances between each point and it's neighboring points.

2. Determine if a point is a border point by checking whether it's distances from neighbors exceed some threshold.

3. Discard border points that belong to the background, which is determined by its corresponding range value in the range image.

4. Apply non-maximum suppression to get "thinner" object borders.

The algorithm then determines interest points using the border information resulting from the previous operations. An interest point aims to represent an area that has significant geometric variation in the immediate vicinity. The size of an area is defined by a parameter – support size, which is the diameter of the sphere around the interest point. An area that contains these characteristics should have substantially different dominant directions of surface changes in a local neighborhood. Such areas can be reliably detected from multiple perspectives. The follow steps identify and extract interest points from a range image:

1. For every point in an image, calculate how much the surfaces changes in its neighborhood and determine the dominant direction of change, incorporating the border information.

2. Score each point according to how much the surrounding dominant directions differ from each other and how stable such point is.

3. Apply smoothing and non-maximum suppression to these scores to find final interest points.

Finally, a NARF feature descriptor is calculated for each of the extracted interest points. These feature descriptors summarize the surface and outer shape of an object. The following steps calculate a NARF feature descriptor for an interest point:

1. Calculate a normal aligned range image for the area that corresponds to the interest point.

2. Overlay a 20 edges asterisk pattern on the normal aligned range image.

3. Calculate how much the pixels changes under each of the beams in the asterisk pattern.

4. Construct a descriptor using these calculated values, such that each element of the descriptor corresponds to the amount of pixels changes under a beam.

5.2.2 Pre-touch Region Proposal

From the NARF algorithm as described above, interest points and NARF feature descriptors are calculated from a range image. These interest points represent the areas that contain the most significant surface changes on the object, and the corresponding feature descriptor provides detailed information of the geometric variation in the area. We introduce a pre-touch region proposal algorithm that utilizes the NARF interest points and feature descriptors to create pre-touch sensing areas (rectangles in the image plane). This algorithm allows the robot to re-estimate object pose based on the object's most diverse surface geometries. The algorithm details are as follows, note that we will expend point 2, 5, and 6 in the subsequence paragraph:

1. Obtain a pointcloud of the object from the robot's head-mounted rgb-d camera.

2. Calculate the object size and set the NARF parameter [support size(σ)] using the follow equation:
$$\sigma = 2 \times \sqrt{\frac{object\_size \times 0.1}{\pi}}$$

3. Convert the input pointcloud to a range image, and extract NARF interest points.

4. Remove the interest points that do not correspond to the object and calculate NARF descriptors for those that remain.

5. For each interest point, find one or more dominant directions of surface change and define a rectangle for each dominant direction.

6. Calculate a confidence score for each of the rectangles.

7. Return a list of rectangles and a list of corresponding confident scores to the robot.

A rectangle is defined by two perpendicular vectors whose origins are at the interest point. The direction of the first vector corresponds to the element of the descriptor that has the greatest value, which indicates the dominant direction of surface change. And the direction of the second is chosen according to which of the two elements perpendicular to the first has the greater descriptor value. The length of these vectors is chosen so that the rectangles area is half that of the rectangle bounding the object. The rectangles confidence score is the sum of the two elements that correspond to the chosen directions.

5.3 Deep Pre-touch Neural Network

The second method is to train a deep neural network to propose pretouch sensing areas. We re-purpose a state-of-the-art object detection neural network for our task. In order to generate examples from which the network can learn, we form a large set of random candidate

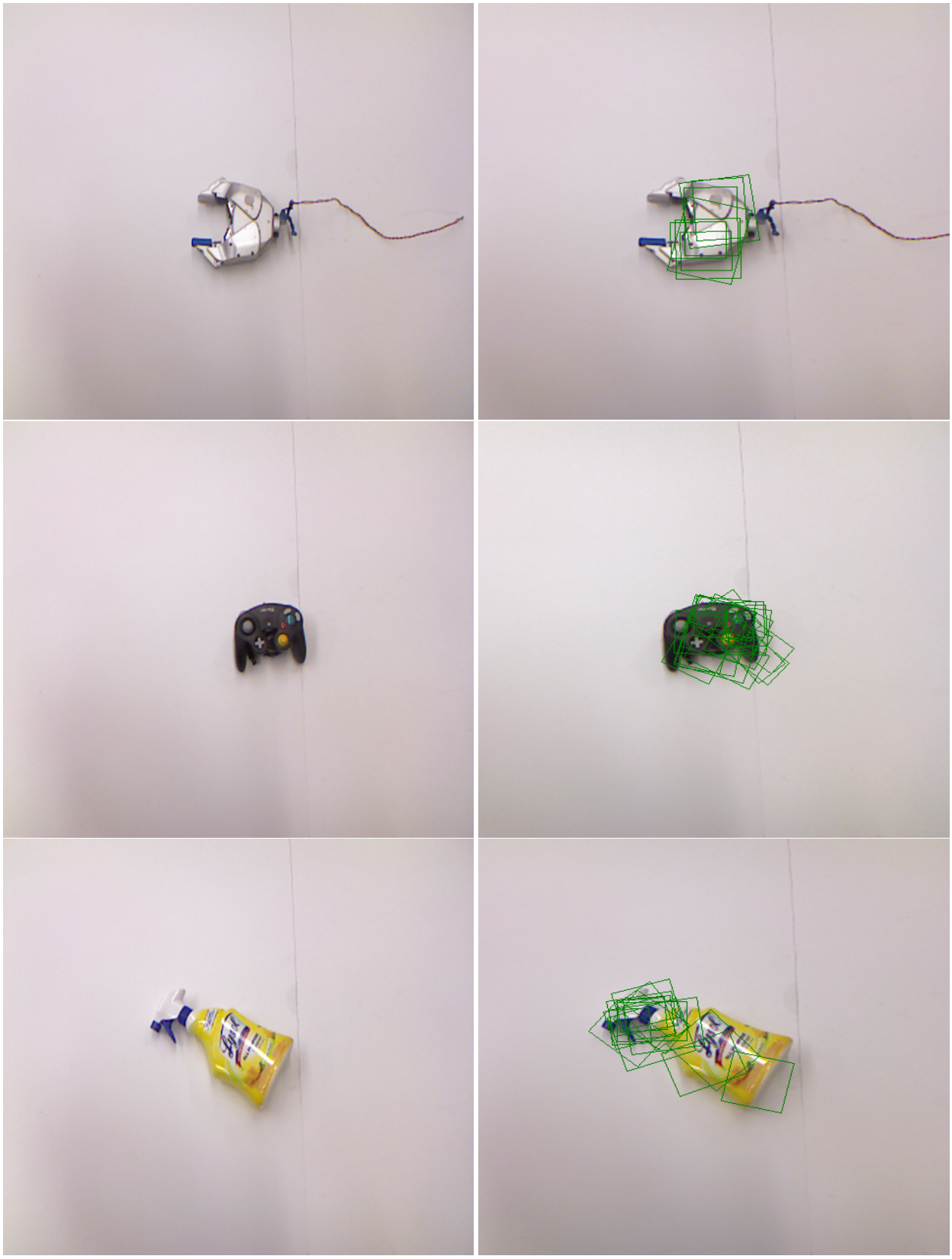

Figure 5.1: Examples of detected pre-touch scan regions.

regions, simulate pre-touch measurement in those regions, and retain the regions that are most robust to random noise and offsets when being matched to the corresponding region of the Kinect cloud.

5.3.1 Neural Network Architecture

The Faster R-CNN framework [33] is used in this method, which uses a Region Proposal sub-network to indicate to the detection layers which sub-patches are likely to contain a region of interest. While the original Faster R-CNN implementation [7] was designed for object detection, we expanded it to be able to predict rotated rectangles (this required us to efficiently calculate the overlap between two rotated rectangles, which was achieved with the General Polygon Clipper library [27]). We initialized the convolutional layers of our network with the 'CNN_M' pre-trained network from [4]. Given a depth image, our network outputs a list of detected regions, and each region is accompanied by a confidence score.

5.3.2 Data Collection

We automate the data collection process and generate a large number of random candidate regions, and then build a set of labels based on which regions yield the most robust ICP matching results. We evaluate a large number of candidates by simulating pre-touch measurement of each region. The label generation algorithm is shown in Algorithm 1. A candidate set of of one thousand rectangles ($n_R = 1000$) is first randomly generated, where each rectangle has an intersection-over-union of at least fifteen percent with the rectangle bounding the object, and has an area that is ten to fifty percent of that of the bounding rectangle. A target cloud is generated for each rectangle. The algorithm then simulates a noiseless pre-touch scan along the perimeter of the region that corresponds to the rectangle by collecting points from an object pointcloud with random offset. The source clouds are generated by adding Gaussian noise ($\sigma = 0.15$cm) to the simulated pre-touch data. The pair is scored by calculating the average distance between the predicted aligned cloud's and the ground truth aligned cloud's points. More detailed explanations of the algorithm can be

Algorithm 1 Label Generation

procedure LABELGENERATION(K) ▷ Kinect cloud K
  $R \leftarrow n_R$ random rectangles
  $K2D \leftarrow$ projection of K into image plane
  $T \leftarrow n_R$ empty target clouds
  for `r in 1:`n_R **do**
    for `i in 1:`n_{K2D} **do**
      if $R^r.nearPerimeter(K2D_i)$ **then**
        $T^r.addPoint(K_i)$
  $\Delta K \leftarrow n_{trials}$ random translations & rotations
  for `j in 1:`n_{trials} **do**
    $K' = K + \Delta K^j$ ▷ Offset cloud K'
    $K'2D \leftarrow$ projection of K' into image plane
    $S \leftarrow n_R$ empty source clouds
    for `r in 1:`n_R **do**
      for `i in 1:`$n_{K'2D}$ **do**
        if $R^r.onPerimeter(K'2D_i)$ **then**
          $S^r.addPoint(K'_i)$
      $trialScores^{j,r} = Score(S^r + N(0, \sigma), T^r)$
  for `r in 1:`n_R **do**
    $scores^r = \max_j(trialScores^{j,r})$
  `return` $FilterRecs(R, scores)$
procedure SCORE(P, Q)
  $PAlign = icp(P, Q)$
  `return` $\frac{\sum |PAlign_i - PAlign_i^*|_2}{n_P}$

found in our publication "Improved Object Pose Estimation via Deep Pre-touch Sensing"[18] from the IEEE International Conference on Intelligent Robots.

5.4 Experimental Result

In order to evaluate the proposed framework, we apply it to pose estimation of eight objects that are not in the dataset used to train the deep neural network. For each object, the PR2 robot receives an initial point cloud from the Kinect, which our detection network uses to propose a set of pre-touch scanning regions. For each region, the robot scans along the path that corresponds to the perimeter of the rectangle. After performing each scan with the pre-touch sensor, we match it with the corresponding region of the Kinect cloud, where the corresponding region is computed in a fashion similar to that of lines 6-8 of Algorithm 1. We obtain a ground-truth estimate of the pose by matching a separate pre-touch scan of the whole object with the Kinect cloud. We argue that this is reasonable because, again, the data from the pre-touch sensor is not affected by mis-calibration of the coordinate frames of the robot's arms. We have also observed, for example as in Fig.5.2, that data from the pre-touch sensor matches reality better than data from the Kinect. All scans were performed such that the sensing beam was normal to the supporting table for simplicity, but future systems could use more complex scanning strategies.

5.4.1 Individual Scan Comparison

We evaluated both methods, as well as an additional region proposal strategy that generates random rectangles. These rectangles are constrained in a fashion similar to those produced by the label generation process (Section 5.3.2). For each of the three methods, we collected scans corresponding to the first ten proposed rectangles for each of the objects, individually matched them to the corresponding portions of the Kinect cloud as previously described, and computed the average distance across corresponding points of the estimated aligned cloud and the ground truth aligned cloud. These pose estimate errors' means and standard deviations are shown in Table 5.1. The standard deviations can be interpreted as a measure

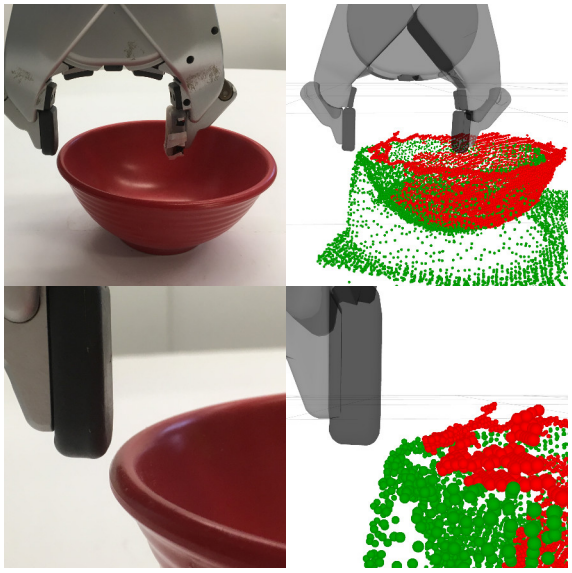

Figure 5.2: Left: A far and close view of the surface of the robot's fingertip aligned with the edge of the bowl. Right: Pre-touch measurements (green) and Kinect measurements (red) with respect to the fingertip.

of each method's consistency, where a lower value indicates greater consistency.

For all but one of the items, our region detection network had the lowest mean, typically by a significant margin. Also, the network had the lowest standard deviation for most of the items, while being outperformed by a small margin on two of the items. The NARF region proposal algorithm out performed the other two methods on the cleanser. However, it has similar performance as the random region generator for all of the other objects. Moreover, the time needed for the region detection process is much longer for NARF region proposal compared to the region detection network. It is clear that the detected regions from our neural network are more optimal for this task.

	Mean Error (cm)			Std. Dev. of Error (cm)		
	Random	NARF	Deep Pre-touch	Random	NARF	Deep Pre-touch
Air Freshener	1.31	2.21	**1.26**	0.52	1.50	**0.50**
Clock	2.21	2.97	**0.80**	1.40	2.16	**0.32**
Cleanser	1.93	**0.71**	1.16	1.46	**0.58**	0.60
Controller	2.77	2.91	**1.49**	1.33	1.96	**0.76**
Fruit Bowl	1.69	1.87	**0.83**	1.50	1.46	**0.29**
Gripper	2.24	1.48	**1.11**	1.39	0.95	**0.41**
Toy	2.55	1.94	**1.72**	1.13	0.66	**0.43**
Wallet	1.16	1.70	**0.89**	**0.63**	0.83	0.66

Table 5.1: Mean and standard deviation pose estimate error across ten individual scans for each of the eight different objects and each of the three methods. All values have centimeter units, and the best value across each of the three methods is bolded.

Chapter 6

CONCLUSION

This work focuses on the development of a novel sensing method – optical time-of-flight pre-touch sensing. By mounting an optical time-of-flight proximity sensor to the end effector of the robot manipulator, pre-touch sensing has the ability to detect and compensate for the errors that are generated by the mis-calibration of the robot's coordinate frames and camera parameters. As a result, the robot can gain significantly higher precision in manipulation and object pose estimation. There are three major contributions in this work. We first developed a new optical time-of-flight pre-touch sensor that is composed of inexpensive components, and showed that this pre-touch sensor allows a PR2 robot to accurately re-localize the Rubik's cube. Our pre-touch sensing method endows the robot with the dexterity necessary to robustly solve the cube. In contrast, without using pre-touch sensors, manipulation errors rapidly accumulates and eventually resulted in a critical failure after approximately nine turns of the cube. We then show that through the use of ICP, pre-touch scanning can be generalized to the pose estimation for common objects. When the scanning region contain enough geometric information, even a single, simple scan can perform the estimation well. Finally, we presented two methods that allow a robot to autonomously propose pre-touch scanning regions. These methods detect object regions that are rich in geometric features, such that the robot can accurately reestimate object pose by only perform pre-touch scanning on these regions, instead of scanning the whole object. The first method proposes pre-touch regions using the NARF interest points and feature descriptors. The second method trains a deep real-time region detection neural network to detect pre-touch regions. Both methods result in object pose estimation that is reasonably accurate, yet, deep neural network requires significantly less time to propose regions and has smaller estimation errors on most of the

experimental items.

BIBLIOGRAPHY

[1] Ben Bellekens, Vincent Spruyt, Rafael Berkvens, and Maarten Weyn. A survey of rigid 3d pointcloud registration algorithms. In *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies, August 24-28, 2014, Rome, Italy*, pages 8–13, 2014.

[2] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *Robotics & Automation Magazine, IEEE*, 22(3):36–52, 2015.

[3] Ly-Yu Chang, Joshua R Smith, and Dieter Fox. Interactive singulation of objects from a pile. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3875–3882. IEEE, 2012.

[4] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[5] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[6] Tully Foote. tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pages 1–6, April 2013.

[7] Ross Girshick. py-faster-rcnn. `https://github.com/rbgirshick/py-faster-rcnn`, 2015.

[8] Di Guo, Patrick Lancaster, Liang-Ting Jiang, Fuchun Sun, and Joshua R Smith. Transmissive optical pretouch sensing for robotic grasping. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5891–5897. IEEE, 2015.

[9] Ying He, Bin Liang, Jun Yang, Shunzhi Li, and Jin He. An iterative closest points algorithm for registration of 3d laser scanner point clouds with geometric features. *Sensors*, 17(8):1862, 2017.

[10] E Farrell Helbling, Sawyer B Fuller, and Robert J Wood. Altitude estimation and control of an insect-scale robot with an onboard proximity sensor.

[11] Dirk Holz, Alexandru E Ichim, Federico Tombari, Radu B Rusu, and Sven Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics & Automation Magazine*, 22(4):110–124, 2015.

[12] Kaijen Hsiao, Paul Nangeroni, Manfred Huber, Ashutosh Saxena, and Andrew Y Ng. Reactive grasping using optical proximity sensors. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2098–2105. IEEE, 2009.

[13] Liang-Ting Jiang and Joshua R Smith. Seashell effect pretouch sensing for robotic grasping. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2851–2858. IEEE, 2012.

[14] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3304–3311. IEEE, 2011.

[15] Gregory Kahn, Peter Sujan, Sachin Patil, Shaunak Bopardikar, Julian Ryde, Ken Goldberg, and Pieter Abbeel. Active exploration using trajectory optimization for robotic grasping in the presence of occlusions. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4783–4790. IEEE, 2015.

[16] Charles C Kemp, Aaron Edsinger, and Eduardo Torres-Jara. Challenges for robot manipulation in human environments. *IEEE Robotics and Automation Magazine*, 14(1):20, 2007.

[17] Danica Kragic, Henrik I Christensen, et al. Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, 15, 2002.

[18] Patrick Lancaster, Boling Yang, and Joshua R Smith. Improved object pose estimation via deep pre-touch sensing. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2448–2455. IEEE, 2017.

[19] Adam Leeper, Kaijen Hsiao, Eric Chu, and J Kenneth Salisbury. Using near-field stereo vision for robotic grasping in cluttered environments. In *Experimental Robotics*, pages 253–267. Springer, 2014.

[20] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[21] Rui Li, Robert Platt, Wenzhen Yuan, Andreas ten Pas, Nathan Roscup, Mandayam A Srinivasan, and Edward Adelson. Localization and manipulation of small parts using gelsight tactile sensing. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3988–3993. IEEE, 2014.

[22] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2308–2315. IEEE, 2010.

[23] Alexis Maldonado, Humberto Alvarez, and Michael Beetz. Improving robot manipulation through fingertip perception. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2947–2954. IEEE, 2012.

[24] Sam Marden and Jose Guivant. Improving the performance of icp for real-time applications using an approximate nearest neighbour search. In *Proceedings of the Australasian Conference on Robotics and Automation, Wellington, New Zealand*, pages 3–5, 2012.

[25] Brian Mayton, Louis LeGrand, and Joshua R Smith. An electric field pretouch system for grasping and co-manipulation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 831–838. IEEE, 2010.

[26] Stephan Muhlbacher-Karrer, Andre Gaschler, and Hubert Zangl. Responsive fingerscapacitive sensing during object manipulation. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4394–4401. IEEE, 2015.

[27] Alan Murta. A general polygon clipping library. *Advanced Interfaces Group, Department of Computer Science, University of Manchester, Manchester, UK*, 2000.

[28] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[29] Anna Petrovskaya and Oussama Khatib. Global localization of objects via touch. *Robotics, IEEE Transactions on*, 27(3):569–585, 2011.

[30] Jeff M Phillips, Ran Liu, and Carlo Tomasi. Outlier robust icp for minimizing fractional rmsd. In *3-D Digital Imaging and Modeling, 2007. 3DIM'07. Sixth International Conference on*, pages 427–434. IEEE, 2007.

[31] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016.

[32] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1316–1322. IEEE, 2015.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[34] Lorenzo Riano. pr2 rubiks solver, 2011.

[35] Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. The diameter of the rubik's cube group is twenty. *SIAM Review*, 56(4):645–670, 2014.

[36] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[37] Alexander Schmitz, Yusuke Bansho, Kuniaki Noda, Hiroyasu Iwata, Tetsuya Ogata, and Shigeki Sugano. Tactile object recognition using deep learning and dropout. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 1044–1050. IEEE, 2014.

[38] Joshua R Smith, Eric Garcia, Ryan Wistort, and Ganesh Krishnamoorthy. Electric field imaging pretouch for robotic graspers. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 676–683. IEEE, 2007.

[39] ST Microelectronics. *Proximity and ambient light sensing (ALS) module*, 8 2014. Rev. 6.

[40] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, and Wolfram Burgard. Point feature extraction on 3d range scans taking into account object boundaries. In *Robotics and automation (icra), 2011 ieee international conference on*, pages 2601–2608. IEEE, 2011.

[41] Nikolaus Vahrenkamp, Steven Wieland, Pedram Azad, David Gonzalez, Tamim Asfour, and Riidiger Dillmann. Visual servoing for humanoid grasping and manipulation tasks. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 406–412. IEEE, 2008.

[42] Ryan Wistort and Joshua R Smith. Electric field servoing for robotic manipulation. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 494–499. IEEE, 2008.

[43] YF Wu, W Wang, KQ Lu, YD Wei, and ZC Chen. A new method for registration of 3d point sets with low overlapping ratios. *Procedia Cirp*, 27:202–206, 2015.

[44] Wendy S Yambor, Bruce A Draper, and J Ross Beveridge. Analyzing pca-based face recognition algorithms: Eigenvector selection and distance measures. In *Empirical evaluation methods in computer vision*, pages 39–60. World Scientific, 2002.

[45] Boling Yang, Patrick Lancaster, and Joshua R Smith. Pre-touch sensing for sequential manipulation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5088–5095. IEEE, 2017.

[46] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: a globally optimal solution to 3d icp point-set registration. *arXiv preprint arXiv:1605.03344*, 2016.

[47] Xuetao Zhang, Libo Jian, and Meifeng Xu. Robust 3d point cloud registration based on bidirectional maximum correntropy criterion. *PloS one*, 13(5):e0197542, 2018.

[48] Cezary Zieliski, Tomasz Winiarski, Wojciech Szynkiewicz, Maciej Staniak, Witold Czajewski, and Tomasz Kornuta. Mrroc++ based controller of a dual arm robot system manipulating a rubiks cube. Technical report, Citeseer, 2007.