Construction of a 3D Object Recognition and Manipulation Database from Grasp Demonstrations

by

David Kent - davidkent@wpi.edu

A Thesis

Submitted to the Faculty

of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Robotics Engineering

April 2014

Approved

Professor Sonia Chernova, Thesis Advisor

Professor Dmitry Berenson, Thesis Committee Member

Professor Michael Gennert, Thesis Committee Member

Abstract

Object recognition and manipulation are critical for enabling robots to operate within a household environment. There are many grasp planners that can estimate grasps based on object shape, but these approaches often perform poorly because they miss key information about non-visual object characteristics, such as weight distribution, fragility of materials, and usability characteristics. Object model databases can account for this information, but existing methods for constructing 3D object recognition databases are time and resource intensive, often requiring specialized equipment, and are therefore difficult to apply to robots in the field. We present an easy-to-use system for constructing object models for 3D object recognition and manipulation made possible by advances in web robotics. The database consists of point clouds generated using a novel iterative point cloud registration algorithm, which includes the encoding of manipulation data and usability characteristics. The system requires no additional equipment other than the robot itself, and non-expert users can demonstrate grasps through an intuitive web interface with virtually no training required. We validate the system with data collected from both a crowdsourcing user study and a set of grasps demonstrated by an expert user. We show that the crowdsourced grasps can produce successful autonomous grasps, and furthermore the demonstration approach outperforms purely vision-based grasp planning approaches for a wide variety of object classes.

Acknowledgements

I would like to thank Professor Odest Chadwicke Jenkins of Brown University for allowing the use of the PR2 robot throughout the user study and validation experiments. Furthermore, I would like to thank fellow graduate students Russell Toris for the development of the Robot Management System, and Morteza Behrooz for his help in conducting the user study. This work was supported by the NSF award number 1149876, CAREER: Towards Robots that Learn from Everyday People, PI Sonia Chernova, and ONR Grant N00014-08-1-0910 PECASE: Tracking Human Movement Using Models of Physical Dynamics and Neurobiomechanics with Probabilistic Inference, PI Odest Chadwicke Jenkins.

Contents

List of Figures

List of Tables

1 Introduction

The abilities to recognize and manipulate a wide variety of objects are critical for robots operating autonomously in household environments. These environments often contain a large variety of objects with which a robot may need to interact. As such, a robot must automatically determine useful properties of encountered objects, or make use of a database of known objects. Automatic grasp detection can lead to difficulties in determining usability characteristics for arbitrary objects, e.g. a bottle of water should not be grasped in such a way that the opening is obstructed, preventing pouring. A database of known objects enables this type of information to be encoded into the object models.

A major disadvantage of object recognition databases is that they require a significant amount of work to construct. Large object recognition databases do exist, but researchers are then limited to only the objects included in the database, which in turn limits the environments in which a robot can operate. Adding new objects to a database requires obtaining a detailed model of each new object. The current state of the art is to use a turntable with sensors mounted around it in a specific setup [19, 18]. This is a time consuming approach, and it cannot be used for learning in a real-world environment, where such a detailed setup is impractical. Learning object models based on real-world observations encountered during operation in the target environment is a more challenging problem since the object angles and relationships between multiple views are unknown. Additionally, the challenges of collecting sufficient training data and object labels must also be overcome.

It is difficult to determine usability information for a set of objects using an autonomous process without any prior knowledge of the object, as environments can have a large number of objects that must all be grasped with different criteria in mind. Some usability information can be generalized, such as the pouring metric mentioned above, but this metric only applies to certain sets of objects. Complicating the grasping issue further, certain difficult-to-observe physical characteristics of objects, such as weight distribution, fragility of materials, or other material properties, need to be accounted for when performing a successful grasp. Grasp demonstration by human users provides a natural way to convey hard-to-observe information to robots, although it can be repetitive and time consuming.

The developing field of web robotics provides new solutions for the collection of both training data for object recognition and demonstrated grasps. Crowdsourcing helps with these repetitive tasks by reducing the time and effort required from each individual user, while still providing large amounts of data. One successful method of taking advantage of the Internet for object recognition is in using large annotated datasets from online image search engines, such as Flickr or Google [6, 33]. We present a different method of incorporating crowdsourcing into an object recognition and grasping system, in which participants connect remotely to a robot and label objects while demonstrating grasps through teleoperation. This allows for robot-specific data beyond object labels, such as successful grasp examples, to be collected via the web. With participants demonstrating both object recognition and manipulation techniques, the data can automatically reflect usability characteristics. We present a method of incorporating data collected from crowdsourcing into an object recognition and grasping database, resulting in an automated grasping technique that inherently accounts for usability constraints.

This thesis makes three contributions. First, we present a web-based system for the collection of point cloud data and demonstrated grasps for a desired set of objects. We designed this system specifically to allow inexperienced participants to remotely demonstrate grasps and collect object data while controlling a sophisticated robotic manipulation platform through a web browser. Furthermore, the system makes it easy to add new objects, and requires no further equipment than the robot itself. We present a data collection user study in which we explore three study conditions to determine how to most effectively engage participants and improve their success rate, thereby providing us with greater amounts of usable data in a shorter period of time.

The second contribution is a system for constructing a functional object recognition system using only point cloud data gathered through crowdsourcing. The system constructs 3D object models given a set of point cloud representations of objects collected from various object views, by determining disparate sets of overlapping point clouds and merging each set of point clouds with common features. We demonstrate that when combined with object labels provided by either a researcher or through crowdsourcing, these 3D object models can be used for object recognition. We present results of using this object recognition system with the object database constructed from a crowdsourcing user study. We also include further analysis using supplemental data collected from a larger and more varied set of objects.

The final contribution is a system for incorporating grasping data into the object recognition database, while still meeting the original goals of the recognition database, i.e. that it should be easy to add new objects, and no extra equipment other than the robot and the objects themselves would be required. Both this ease of use and lack of setup allows the process to be crowdsourced, which is appealing to researchers who want to quickly expand the capabilities of their system. We also compare the effectiveness of the crowdsourced grasps to expert demonstrated grasps, and we show the advantages that this grasping system has over a grasp planner based solely on the geometry of the object.

2 Related Work

This section begins with an overview of crowdsourcing for user studies involving robots, followed by a discussion of existing methods for constructing object recognition databases, and concludes with a survey of approaches for grasp planning.

2.1 Crowdsourcing

Crowdsourcing, or obtaining information from large amounts of people through the Internet, is a technique with growing relevance for robotics. In particular, it has the potential to change the way researchers conduct user studies. These user studies require a remote lab setting, in which users can control a robot through the web. One such example of a remote lab is the PR2 remote lab, which provided a framework for allowing trained and expert users to interact with a shared PR2 over the Internet [8, 23]. This remote lab did was not designed with untrained users in mind, and as such it was not suited to large-scale crowdsourced user studies. Addressing this problem led to the development of the Robot Management System (RMS) [32], which provides tools for researchers to create, manage, and deploy user study interfaces that allow naive users to directly control robots through a web browser. RMS allows anyone with an Internet connection to remotely participate in a user study from the comfort of their own home. Along with its ability to reach large groups of people, RMS includes automatic user scheduling, which allows user study sessions to be run with little to no break in between. These properties make RMS an excellent system for implementing a crowdsourcing experiment.

Sorokin et al. conducted previous work using crowdsourcing for novel object grasping [29]. In this work, the researchers divided object modeling into subproblems of object labeling, human-provided object segmentation, and final model verification. Workers on Amazon Mechanical Turk (AMT), a microtask crowdsourcing marketplace, completed these simpler tasks. Sorokin demonstrated successful object recognition and grasping using the results obtained from AMT workers. Microtask crowdsourcing marketplaces do have some limitations, however. The system does not allow for direct control of a robot, and researchers must design tasks to be simple and quick to ensure for high quality data. With new web robotics systems such as RMS [32], we can perform more complicated tasks, including the grasp pose demonstrations used in this thesis.

Crowdsourcing has been successfully applied to other aspects of robotics, as well. Tellex et al. used crowdsourcing to learn natural language commands for controlling robot navigation and manipulation [31]. Crick et al. showed that crowdsourcing could be used to test interfaces for teaching a robot to navigate a maze, showing that remote users perform better when limited only to information that the robot has [9]. Other studies have shown that crowdsourcing can be used to learn human-robot interaction behaviors, and that gamification of the data collection process helped to motivate users to provide useful data [7, 4].

Quality of data is an important concern in crowdsourcing. The relative anonymity of remote participants as compared to traditional user study participants decreases motivation to provide high quality data. Harris explores incentivization models for improving the quality of crowdsourced data by providing numerical rewards based on the success of participants. Furthermore, the study has shown that providing a perceived negative numerical reward on participant failure can further improve the quality of data [15]. We built on these ideas by designing a scoring system involving both positive and negative rewards to motivate remote participants to provide accurate data in a timely manner.

2.2 Object Recognition Databases

One method for crowdsourcing object recognition involves the use of online image search databases. Chatzilari et al. demonstrate that a database consisting of annotated images downloaded from Flickr is sufficient to recognize objects [6]. A downside to this approach is that it is limited to the set of keywords used during download. Generalizing this type of approach to work for any potential object, Torralba et al. constructed a dataset by downloading 760 Gigabytes of annotated images using every non-abstract noun in the English language over the course of 8 months [33]. Because of the limitations that arise from keyword selection, we avoided the use of online databases for the object recognition system presented in this thesis. Furthermore, there are no large databases comparable to online image searches that also contain object manipulation data (e.g. grasp points).

There are many alternative methods to creating object recognition databases that do not use the Internet. Lai et al. provide an example of a common approach to building 3D models [19]. Using a constantly rotating turntable and a set of cameras mounted at multiple angles from the table horizon, accurate 3D object models can be constructed for use in object recognition. Kasper et al. also used a turntable to construct 3D object models for the KIT object models database, although they required a larger and more expensive array of sensors [18]. A disadvantage of these approaches is that they require an accurate setup of sensors and turntables to produce an accurate model, and each new object must be scanned individually using such a process. As with using online databases, these techniques also do not allow for the collection of grasp information.

While the turntable method allows 3D data to be easily reconstructed into a complete object model, there are also methods to combine unordered point cloud data. Automatic registration of overlapping point clouds can be accomplished through feature matching [25, 28], correlation of extended Gaussian images [20], and geometric optimization [21].

Computer vision researchers have demonstrated many approaches that construct 3D object models from ordered [2, 11] and unordered [5] sets of 2D images as well. These approaches are effective for generating 3D models, but similar problems arise as with the turntable method, as they do not allow for the collection of grasping information about the objects.

2.3 Grasp Planning

Many methods already exist for vision-based grasp planners that use data from depth images. Many of these approaches forgo object recognition and rely solely on geometric analysis of partial views of novel objects to compute effective grasp points [30, 17, 27]. Vision-based grasp planners can be effective for grasping many objects, but they do not account for any object information beyond an object's visible shape.

In this work we use Willow Garage's PR2 robot. This platform has two common off-the-shelf grasp planners. One uses point cloud data without an object recognition component [16], which is susceptible to the same problems as the previously discussed partial shape grasp planners. The other grasp planner incorporates Willow Garage's household objects database [12], which matches detected objects to complete 3D models, and includes a set of example grasps calculated for each model using the GraspIt! simulator. The database was designed to work with a limited set of commonly available objects. It relies on detailed 3D modeling and is constrained to objects that are either rotationally symmetric, or have no indentations or concavities. As a result, the approach is difficult to use in new environments because of the difficulty in adding new object models. Additionally, the grasps in the household objects database were calculated solely based on object shape; they do not account for other object properties such as weight, material, and moving parts. Our method aims to solve the same problem as the PR2's grasp planners, i.e. to determine a successful grasp pose given an incomplete point cloud of an object, but we seek to improve the quality of the grasps from a usability standpoint, remove the restrictions on object shape, and simplify the process of adding new objects.

Work has also been done on autonomous grasp evaluation based on usability characteristics. Baier and Zhang [3] present four grasp usability criteria, and include analytical methods for evaluating them. The four criteria are pour-in, pour-out, handover, and movement. The pour criteria are specific to objects that require unobstructed openings for pouring, but the second two criteria generalize well to all objects, in that they measure the ease of which an object can be re-grasped, as well as a grasp's resistance to movement. These criteria represent some usability characteristics of objects, but there are many unrepresented characteristics that are specific to other objects. Additionally, there is also the issue of determining which usability characteristics should apply to which objects. Both of these issues make it difficult to explicitly define usability characteristics for arbitrary object sets.

These usability characteristics are, however, naturally known by humans, and Xue et al. present a system for leveraging human knowledge, as well as other non-vision based object characteristics, into a multi-modal grasp planning system [34]. The system requires data collection of an object's shape, texture, and weight, and also allows input from human instructors who demonstrate areas that either must be touched during grasping, or must be avoided as obstacles. The drawback of this approach is that it requires a complicated setup, including a digitizer, turntable, movable stereo camera setup, and a tactile glove. This approach is also unsuitable for crowdsourcing, as it requires a trained demonstrator to give semantic information about each object.

Other object model databases exist that include grasping information, such as the partial-view-based data-driven grasping system presented in [13]. This system adds the object recognition component to the partial-view depth image grasp planners, and retrieves grasp points from an online database. The system is more concerned with the recognition side of the problem, though, and does not address where the grasp data comes from or how it can be used. Another object model and grasping database addresses the problem of defining grasps by using a set of grasps exhaustively calculated offline based on CAD models [22]; this requires any object to be modeled in detail before it can be used. Similarly, Goldfeder et al. present a large-scale database constructed with grasps calculated using the GraspIt! simulator [14]. The simulator was used to replace human grasp demonstrations to reduce time constraints in constructing a database of thousands of objects. As a result, the database includes many baseline form closure grasps for each object.

An alternative to calculating grasp points is to learn them through trial and error. Detry et al. present an autonomous experimentation system where a robot learns grasp points by repeatedly attempting to grasp an object at various points and adjusting probabilistic grasping models based on the results [10]. We use a similar idea for evaluating grasp models, except we use crowdsourced human-demonstrated grasps to decrease the amount of time of the trial-and-error learning system, while also leveraging human knowledge of object usability characteristics.

3 The Robot Management System

The Robot Management System (RMS) is an open-source framework that allows researchers to quickly and easily install, configure, and deploy a secure and stable remote lab system[1]. The RMS allows naive users to create accounts, gain access to robots, and participate in research studies.

The framework is designed in a robot and interface independent manner. At its core, the RMS is a custom content management system written in PHP backed by a MySQL database. Its main goal is to keep track of different ROS enabled robotic environments, interfaces, users, and research studies with little need of additional programming by researchers. Furthermore, the system is fully integrated and takes advantage of the tools and libraries developed as part of the Robot Web Tools effort [1]. By doing so, such a system enables researchers to focus on the goals of their research without needing to spend countless hours testing and

[1]Documentation and source code are available at http://www.ros.org/wiki/rms. Tutorials are available at http://www.ros.org/wiki/rms/Tutorials.

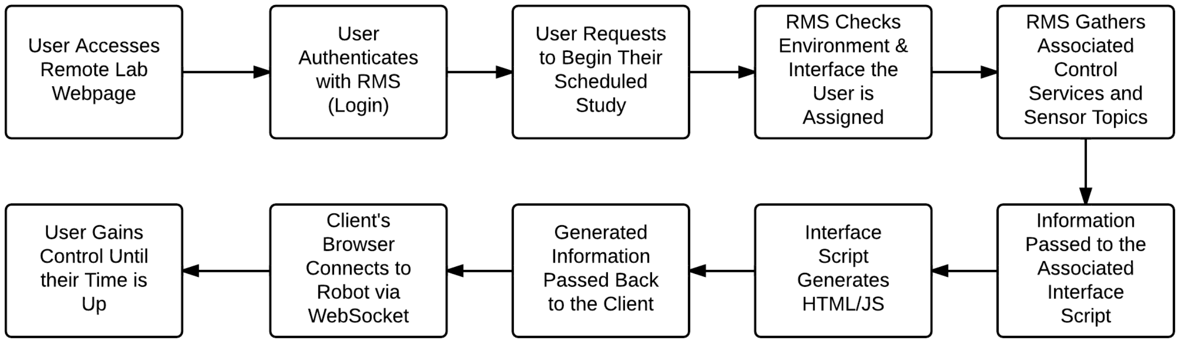

User Accesses Remote Lab Webpage
User Authenticates with RMS (Login)
User Requests to Begin Their Scheduled Study
RMS Checks Environment & Interface the User is Assigned
RMS Gathers Associated Control Services and Sensor Topics
Information Passed to the Associated Interface Script
Interface Script Generates HTML/JS
Generated Information Passed Back to the Client
Client's Browser Connects to Robot via WebSocket
User Gains Control Until their Time is Up

Figure 1: A Pipeline Showing a User Connecting to a Robot Environment via the RMS

implementing a custom web solution. The RMS was developed with the following goals in mind:

- Robot and interface independent design
- Support for easy creation and management of new widgets and interfaces
- Secure user authentication and authorization
- Creation, management, logging, and analysis of multi-condition user studies
- Website content management

The complete pipeline for RMS is shown in Figure 1. We will briefly explain parts of the pipeline relevant to this thesis below, including robot, environment, interface, and user study management. For more details on the complete system, see [32].

Managing Robots and Their Environments: The RMS focuses much of its efforts on managing robots and robot environments. We define a robot environment as a single robot and its associated surroundings. The RMS uses the genericness of ROS message and service types to allow the control and sensor displays of an abstract set of robots. Within our research group alone, the RMS has been easily connected to complex, research-grade robots such as the PR2 from Willow Garage and the youBot from KUKA, to simple robots such as the Rovio from WowWee. The job of the RMS is to keep track of how to connect to each robotic environment and its associated control and sensor feeds. Researchers can quickly configure each environment by adding information such as camera feeds, arm and base control services, or point cloud topics. In this work, we use a PR2 robot with a head mounted Kinect sensor. The RMS allows us to stream camera feeds from the PR2 and point cloud data from the Kinect to remote user study participants, while also allowing remote users to control the robot's end effectors through ROS messages.

Interface Management: An important feature of the RMS is the ability to manage different interface layouts. This allows the ability to conduct studies such as A/B interface testing or creating interfaces based on varying levels of user expertise. In this work, we use the RMS to define three versions of a PR2 teleoperation interface with varying levels of feedback.

User Study Management: Researchers can create user studies that can then have one or more associated conditions. Each condition is also associated with an interface. For our user study, further described in Section 4, we defined three study conditions corresponding to varying levels of feedback, and then associated each condition with an interface that displayed the appropriate level of feedback. With conditions defined, researchers are then able to schedule individual experiments (trials). Such a trial consists of mapping a user to a given condition (and thus interface), environment, and start/end time. The RMS then keeps track of this information and will allow each particular user to gain access to the appropriate robotic environment using a given interface at the correct times. The users will only be allowed to use the interface once their time slot begins, and will be automatically disconnected once their time has ended.

4 Data Collection

Leveraging the Robot Management System, we conducted a crowdsourced user study to gather object recognition and grasping data for the database. In this section, we describe our data collection process, user interface and three study conditions.

4.1 Experimental Setup

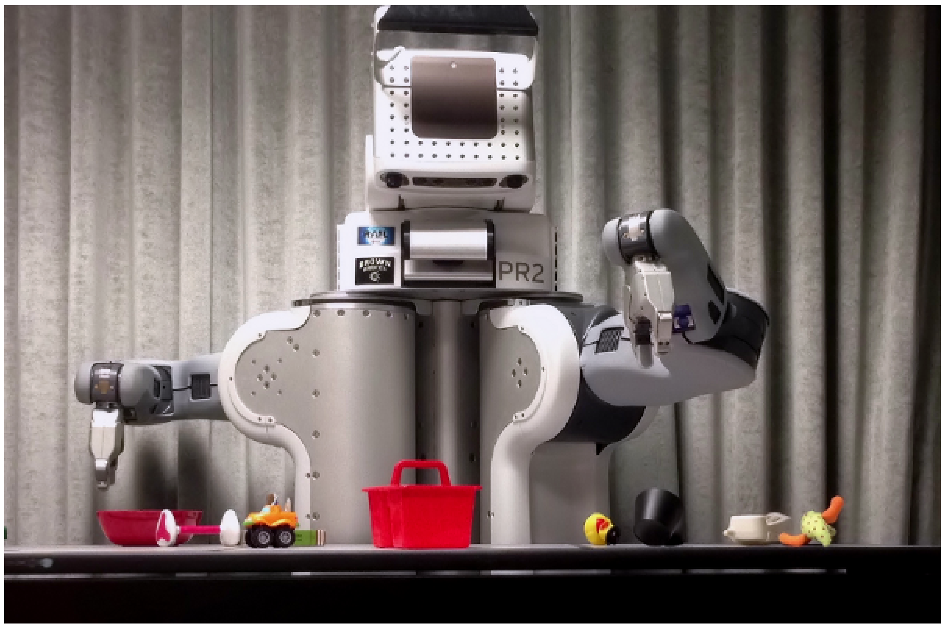


Figure 2: Physical setup of the user study

For the data collection study we recruited 42 participants through advertising on the campus of Worcester Polytechnic Institute and in the surrounding area. The study setup consisted of a PR2 robot placed in a fixed position. The robot faced a table containing ten household objects arranged in random positions and orientations, as shown in Figure 2. The table itself consisted of six sections marked with boundary lines.

The study was conducted using RMS, which enabled participants to remotely connect to a PR2 interface from their home computers using only a web browser. The web interface consisted of two main components, shown in Figure 3. In the view component, participants could switch between video feeds from the cameras in each of the PR2's arms, as well as the RGB video stream from a head-mounted Microsoft Kinect sensor. Participants could also control the direction the head was pointing to adjust the video feed view by using the arrow keys. In the control component, participants used interactive markers on a simulated robot model to change the position and orientation of the physical robot's end effectors. This window also showed the segmented point clouds of each object detected on the table.

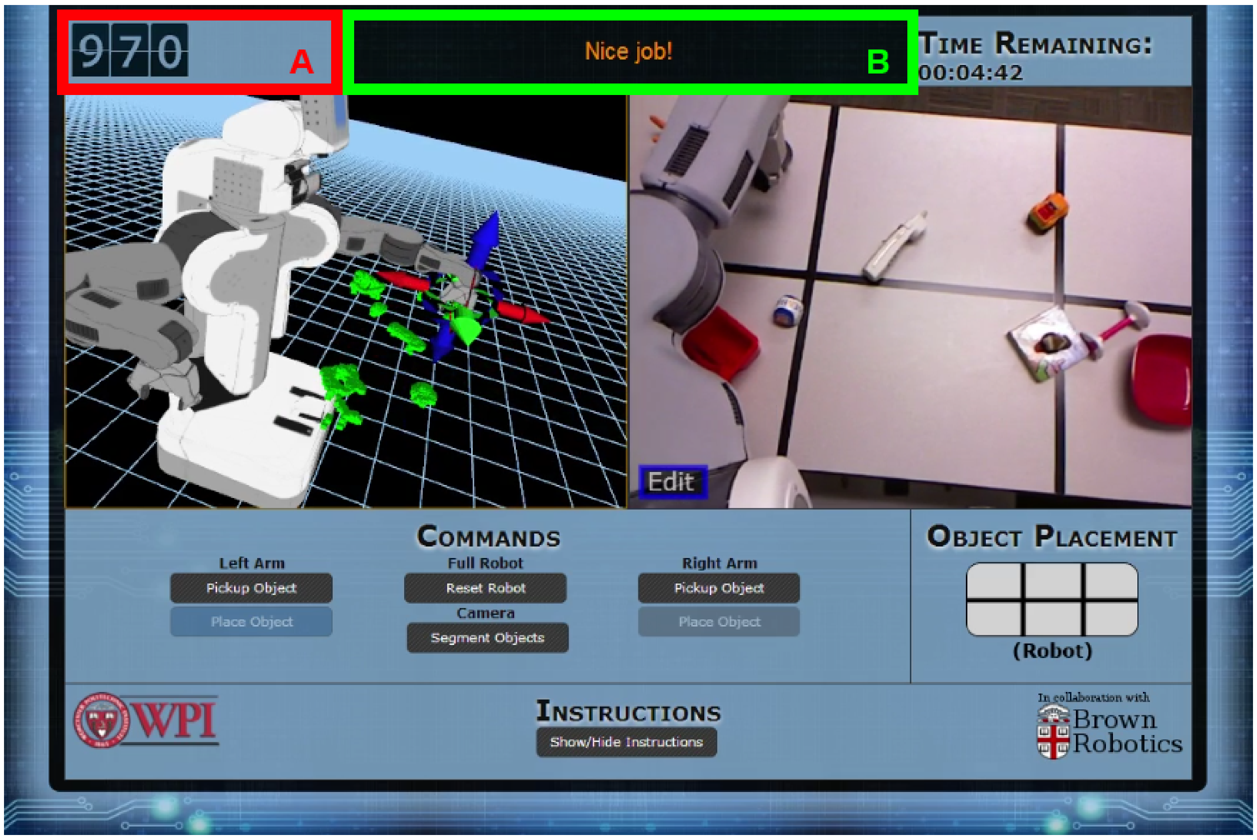
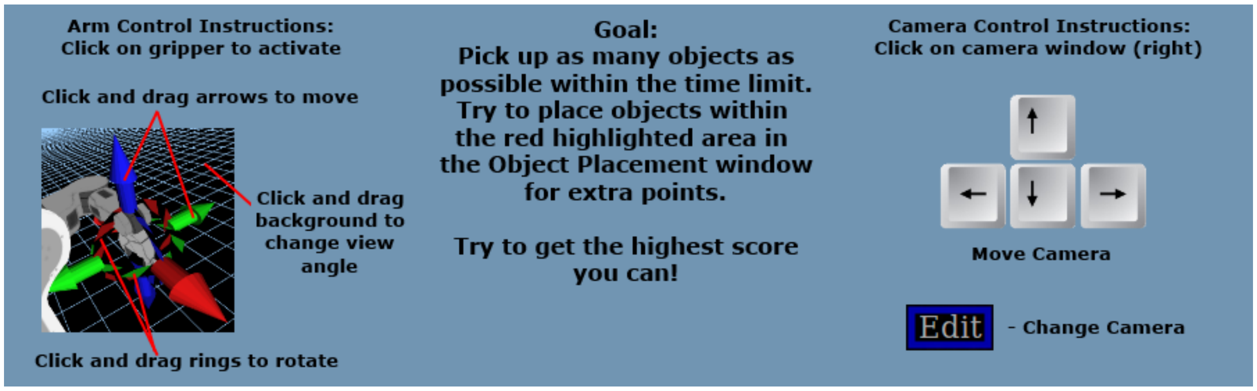


Figure 3: An example of the web interface used in the remote user study. The interface element (A), shown in red, displayed the participant's current score. This element was active for the Score Only and Full Feedback conditions. The interface element (B), shown in green, displayed comments based on the participant's performance. This element was active for the Full Feedback condition only.


Arm Control Instructions:
Click on gripper to activate

Click and drag arrows to move

Click and drag background to change view angle

Click and drag rings to rotate

Goal:
Pick up as many objects as possible within the time limit. Try to place objects within the red highlighted area in the Object Placement window for extra points.

Try to get the highest score you can!

Camera Control Instructions:
Click on camera window (right)

Figure 4: The user study instructions shown in the interface. For the No Feedback condition, any mention of points and scores were removed.

Participants connected to the robot for sessions consisting of twenty minutes. Upon initial connection, the interface displayed the set of instructions shown in Figure 4. These instructions explained both the interface controls and the participant's goal, in as compact a space as possible. At any time during the user study, participants could show the instructions again by clicking on the Show/Hide Instructions button. All participants were instructed to pick up as many objects as possible within the allotted time. After grasping an object, the interface asked participants to label the object. The interface then highlighted a randomly selected section of the table surface where participants were to place the object. In this manner, the user study was automatically resetting, in that each new participant began the study with a different arrangement of objects, determined by how the previous user placed each object. Participants were classified into one of three study conditions, as described in Section 4.2.

Upon successfully picking up an object, the system stored:

- the segmented point cloud of the grasped object using the head-mounted Kinect,
- the position and orientation of the PR2's gripper,
- force data from the sensor array on each gripper finger, and
- the object label provided by the participant.

The system stored this data in a database for later use in constructing 3D object models. The randomized object placement allowed for the collection of data for multiple common orientations of each object, without the need for a researcher to change the object orientations between each user session. Figure 5 shows an example of point cloud data gathered in the user study, as well as an object model constructed from that data.

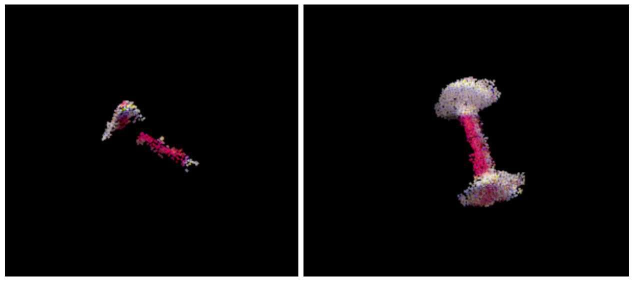


Figure 5: Left: An example of point cloud data gathered upon a successful object pickup. Right: An object model constructed from the database generated by the user study.

4.2 Feedback and Participant Motivation

Because this was our first user study using RMS to collect data entirely from remote users, we collected further data to determine how best to motivate participants to provide high quality data. We equally divided participants among three conditions:

- **No Feedback** - Participants were simply instructed to pick up as many objects as possible within the allotted time.

- **Score Only** - Participants were shown a score that rewarded them for accomplishing the goals of the study. Participants were given 100 points for successfully grasping an object, 50 points for successfully placing an object within the correct table section, and -10 points for each failed pickup attempt. The scores were stored in an online leaderboard so that participants could compete with eachother.

- **Full Feedback** - In addition to the score described above, the interface also provided text feedback, shown in Table 1, offering praise for successful pickups, encouragement

Table 1: Feedback for the Full Feedback study condition

Activation Condition	*Text Feedback*
Successful pickup or placement	Nice job! Way to go! Great job! Keep it up! Nice work! Nicely done!
Failed pickup or placement	Give it another try! Nice try! Keep at it! You can do this! Never give up!
Idle for 20 seconds	Let's get some objects! Take your time to figure it out...
Idle for 30 seconds	If you are confused, try clicking on the Show/Hide Instructions button below...
Object held off of table	You're holding an object off of the table edge

to keep trying after failed pickups, and suggestions to revisit the instructions if they remained idle for a long period of time.

In the No Feedback condition, participants had only the instruction set to guide their actions, and so the tasks they chose to complete were based entirely on their respective interpretations of the instructions. Since we conducted the study remotely, participants could not ask questions to clarify what they were supposed to do, and we could not correct any misinterpretations of the instructions. This condition provided a point of comparison for how participants would perform without any feedback to guide their interpretation of the task.

The Score Only condition added a numerical feedback element to the study. We designed the scoring system to reinforce the elements of the study that were most important to our data collection. As such, participants were rewarded most for successful object grasps, which were the primary data collected during the user study. To further stress the importance of object grasping, we gave negative feedback only for missed object grasps. Correct object placement was a secondary goal, as it aided in the self-resetting nature of the study, hence the lower point reward. Overall, the scoring also added an element of competition, motivating participants to pick up as many objects as they could to compete with other participants.

While the score added a feedback element based directly on a participant's actions, we also wanted to determine whether semantic feedback would improve a participant's performance. The Full Feedback condition added feedback in the form of pre-authored phrases displayed in response to specific actions. As with the score, we designed these phrases to provide positive reinforcement for successfully completing important actions in the user study, but unlike the score, the text feedback could also encourage participants to keep trying when they failed important actions. The text feedback also helped to supplement the participants' understanding of the environment by displaying a warning when the robot was holding an object off of the table edge, which could be difficult to see given the limited camera views.

We designed the text feedback to fit in with the interface color scheme in a non-distracting manner, while catching the eye of participants with a flipping animation when their score was increased or decreased. Similarly, we designed the feedback text to catch the participant's attention using complimentary colors, while being small enough to not distract from the task at hand. Examples of a score and displayed text in the interface can be seen in elements A and B of Figure 3, respectively.

Importantly, the point cloud and grasp data collection process was the same across all three study conditions. The purpose of this study was to examine which condition resulted in the highest quantity and quality of data.

5 Object Model Construction

Given the set of individual segmented point clouds of each object collected using the above data collection method, the next step was to create more complete object models from these individual views. First, we used the Point Cloud Library (PCL) [26] to implement a pairwise point cloud registration pipeline. This process allows the merging of any two point clouds into a single point cloud according to correlated SIFT features.

Theoretically, we could construct a complete object model by iteratively performing this pairwise point cloud registration process for all of an object's data. In practice, however, pairing point clouds arbitrarily in this way is highly susceptible to error propagation. Aside from minor errors propagating, a single incorrect merge early on in the process can result in highly inaccurate models. It is therefore critical to distinguish *successful merges* from *failed merges*, where we define a successful merge as one in which all overlapping points common to both point clouds are correctly aligned, so that a human being would verify it as a single object.

We present a novel approach for building object models through iterative point cloud registration. For each pair of point clouds, we first calculate a set of metrics that characterize the potential merge with respect to different properties. Using these metrics, and a training set of hand labeled successful and failed pairwise point cloud registrations, we then train a decision tree to predict successful merges. Finally, we leverage the decision tree to construct a graph representation of candidate pairwise merges that are then iteratively performed to generate the object model.

5.1 Classification of Successful Merges

In considering a candidate merge, we define one point cloud model as the base point cloud B, and the other as the target point cloud T. R represents the point cloud that results from registering T onto B, and we define b_n, t_n, and r_n as the individual points within B, T, and R, respectively. We let $distanceXYZ(i,j)$ and $distanceRGB(i,j)$ represent the Euclidean distance in XYZ space and RGB color space between points i and j, respectively; $nn(p,X)$

represents the nearest neighbor of point p in point cloud X. Finally, parameter δ represents a predefined Euclidean distance threshold between two points in a point cloud.

Based on this representation, we use the following set of metrics to describe a candidate point cloud resulting from a pairwise registration:

- **Overlap** (m_o) - percentage of overlapping points after registration

$$m_o = \frac{|\{i|b_i - nn(b_i, T) < \delta, i \in B\}|}{|B|} \tag{1}$$

- **Distance Error** (m_{dErr}) - the mean of the distance error between merged point pairs

$$m_{dErr} = \frac{1}{|B|} \sum_i distanceXYZ(b_i, nn(b_i, T)) \tag{2}$$

- **Color Error** (m_{cErr}) - the mean of the color difference between overlapping points

$$\begin{gathered} m_{cErr} = \frac{1}{n} \sum_n distanceRGB(b_n, nn(b_n, T)), \\ n \in \{i|b_i - nn(b_i, T) < \delta, i \in B\} \end{gathered} \tag{3}$$

- **Average Color Difference** (m_{cAvg}) - average color difference between point clouds B and T

$$\begin{aligned} m_{cAvg} = abs(avg(b_i.r + b_i.g + b_i.b) - avg(t_i.r + t_i.g + t_i.b)), \\ i \in B, j \in T \end{aligned} \tag{4}$$

- **Color Deviation Difference** (m_{cDev}) - difference in standard deviation of color between point clouds B and T

$$\begin{aligned} m_{cDev} = abs(stdev(b_i.r + b_i.g + b_i.b) - stdev(t_i.r + t_i.g + t_i.b)), \\ i \in B, j \in T \end{aligned} \tag{5}$$

- **Size Difference** (m_{size}) - difference in the number of points between point clouds B and T

$$m_{size} = abs(|B| - |T|) \tag{6}$$

- **Spread Difference** (m_{spread}) - difference in the spread (i.e. the distance between the two most distant points within a point cloud) of point clouds B and T

$$\begin{gathered} m_{spread} = abs(max(distanceXYZ(b_i, b_j)) - max(distanceXYZ(t_k, t_l))), \\ i, j \in B; k, l \in T; i \neq j; k \neq l \end{gathered} \tag{7}$$

We designed these metrics to represent different characteristics of the data. m_o and m_{dErr} measure differences in physical shape of the two point clouds. Furthermore, m_o provides an indication of how much new data is added to cloud B by registering it with cloud T. m_{cErr}, m_{cAvg}, and m_{cDev} measure differences in point cloud color; where m_{cAvg} indicates the overall similarity in color of the point clouds, m_{cDev} indicates whether the two point clouds have similar range of color, and m_{cErr} measures differences of color between the points of clouds B and T with relation to their spatial positions in cloud R. The remaining metrics represent a comparison of the overall size of the point clouds, with m_{size} relating their total number of points and m_{spread} relating their maximum physical lengths.

Next, we leveraged these metrics to train a decision tree to predict successful merges. We constructed a training set by applying the seven metrics to 174 instances of pairwise point cloud registrations selected from the user study data. This included both successful and failed merges, and each instance was appropriately labeled. Using this dataset, we used the C4.5 algorithm [24] to generate a decision tree to classify whether a pairwise registration was successful or failed based on the registration metrics. The final decision tree correctly classified 81% of the training data, with a false positive rate of 10%. Minimizing the false positive rate as much as possible was the most important factor in selecting the final decision tree, as accepting a failed merge propagates error throughout the rest of the object model construction process. The decision tree used m_o as the most important classification metric, with m_{dErr}, m_{cAvg}, and m_{spread} also used as important metrics.

5.2 Graph-Based Object Model Generation

To construct the object model, we developed an algorithm (Algorithm 1) that first generates a graph structure representing all of the point clouds for each individual object in a graph, and then selectively registers pairs of point clouds to generate one or more object models. The algorithm initializes a graph with a node representing each point cloud (line 3). It next iteratively considers all pairs of nodes, and constructs edges between them for which the decision tree predicts a successful pairwise merging of the two point clouds represented by the nodes (lines 4-8). The resulting graph structure represents the similarities between the various views of the object.

The object model is constructed by collapsing the graph by merging nodes until no edges remain. The algorithm selects a random edge of the graph and performs pairwise registration on the point clouds connected by the selected edge (lines 10-11). The algorithm then removes the nodes for the two point clouds, and replaces them with a single node representing the new merged point cloud (lines 12-14). Again using the decision tree, the algorithm constructs edges from the new node to the remainder of the nodes in the graph (lines 15-19). The process then repeats until there are no graph edges left.

Figure 6 provides a visual example of the object model construction algorithm for a set of 8 point clouds {A, B, C, D, E, F, G, H}, all collected from the same object (step 1). In step 2, each pairwise combination of the point clouds is checked for successful registrations, resulting in the graph shown in step 3. The algorithm then randomly selects point clouds B and C to be merged, resulting in a new graph shown in step 4 with a new set of unchecked point cloud pairs. In step 5, the unchecked pairs are tested for successful registration. For the next step, the algorithm randomly selects point clouds E and G for merging. This process of testing and merging continues until no successful merges remain, and step 15 shows the final set of models {ABCFH, DEG}.

The result is that the process can represent each object by multiple object models composed of merged point clouds from different views. For objects that look significantly different depending on the viewing angle (e.g. the front and back cover of a book), this removes the risk of poorly merging point clouds taken from significantly different angles and propagating that error through future iterative registration. This method also provides flexibility for adding new data; if any new data is obtained, it can be merged into the existing data by

Algorithm 1 Object model construction by graph

```
Require: List<PointCloud> $P$
 1: List<PointCloud> $nodes$;
 2: List<Edge> $edges$;
 3: $nodes$.addAll($P$);
 4: for all $n_i$, $n_j$ in $nodes$ do
 5:     if isSuccessfulRegistration($n_i$, $n_j$) then
 6:         $edges$.add(Edge($n_i$, $n_j$));
 7:     end if
 8: end for
 9: while $edges$ is non-empty do
10:     Edge $e$ = $edges$.pop();
11:     PointCloud $p$ = pairwiseRegister($e$.node$_1$, $e$.node$_2$);
12:     $nodes$.remove($e$.node$_1$, $e$.node$_2$);
13:     $edges$.removeEdgesContaining($e$.node$_1$);
14:     $edges$.removeEdgesContaining($e$.node$_2$);
15:     for all $n_i$ in $nodes$ do
16:         if isSuccessfulRegistration($p$, $n_i$) then
17:             $edges$.add(Edge($p$, $n_i$));
18:         end if
19:     end for
20: end while
```

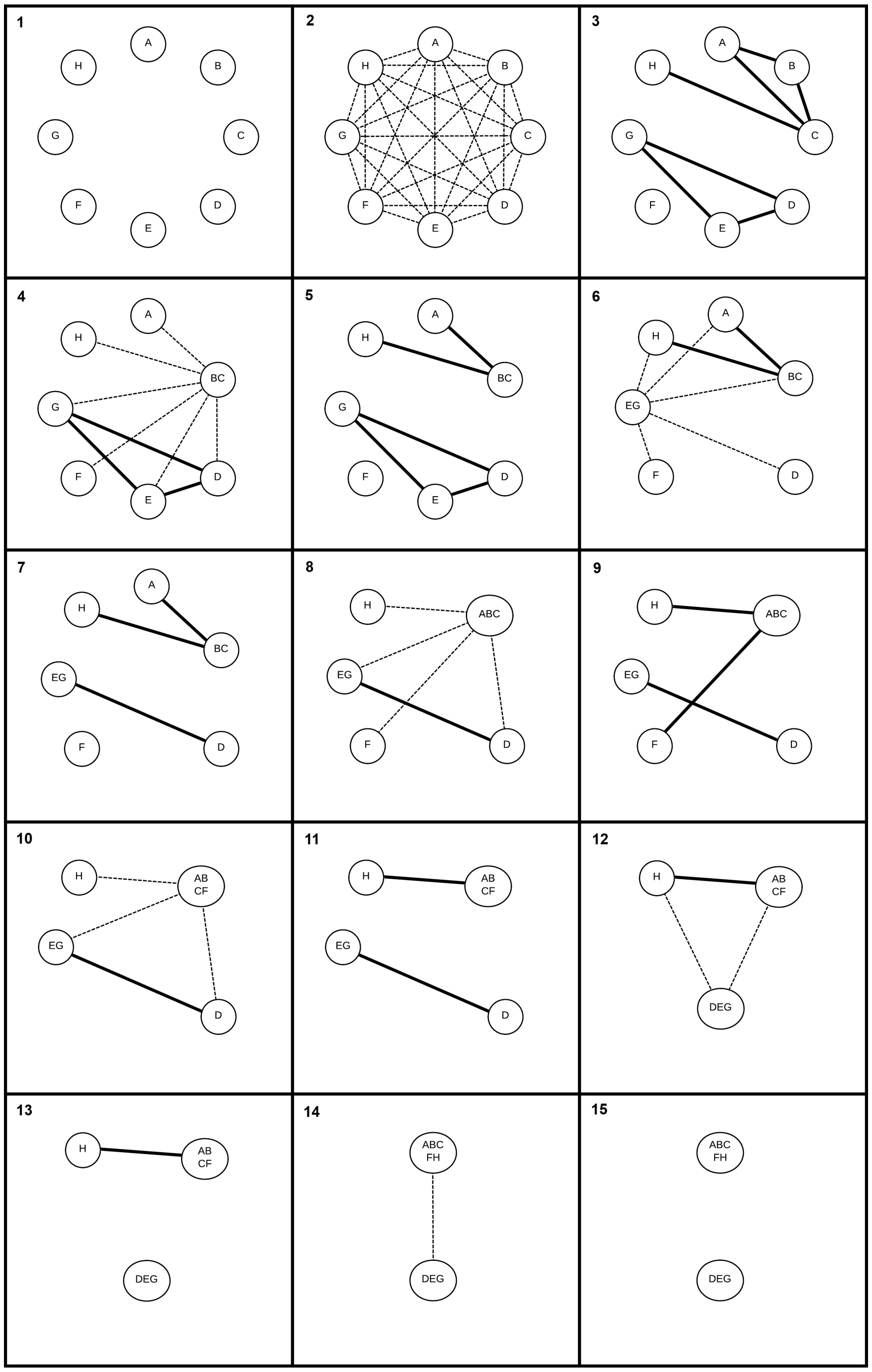


Figure 6: A visual example of the model construction graph. Graph nodes represent point clouds, dashed edges represent unchecked pairwise registrations, and solid edges represent predicted successful pairwise registrations.

initializing graph nodes for both the previously merged object models and the new data.

The entire model construction algorithm takes a set of point clouds as input, and produces a set of object models for each object to be used in object recognition. The algorithm can be run either per object, where a separate graph is constructed for the point clouds associated with each individual object, or for the entire dataset, where a single graph is constructed using all of the point clouds as nodes. The latter approach has a significantly higher runtime, but as this is essentially a training algorithm, it does not have to run in real-time.

5.3 Object Recognition

In order to test the usefulness of the models created by the process described above, we implemented an object recognition system based on processes similar to those used in the model construction. Specifically, using the same pairwise point cloud registration pipeline implemented for the model construction, the object recognition algorithm attempts to register a point cloud from an unknown object to each object model in the recognition database. By calculating the metrics listed above, the algorithm assigns a measure of error to each attempted pairwise registration. The recognition algorithm then classifies the query object as the database object that resulted in the lowest error.

We define the registration error, s_{err}, as a linear combination of the normalized distance error and the normalized color error.

$$s_{err} = \alpha * norm(m_{dErr}) + (1 - \alpha) * norm(m_{cErr}) \tag{8}$$

The parameter α can be set within the range $[0, 1]$ to adjust the relative weighting of the distance error and color error. Setting α closer to 0 causes the algorithm to prioritize differences in color rather than shape; likewise setting α closer to 1 causes the algorithm to prioritize differences in shape rather than color. For all experiments presented in this thesis, we set α to 0.5, since we consider differences in shape and color to be equally important in our object sets.

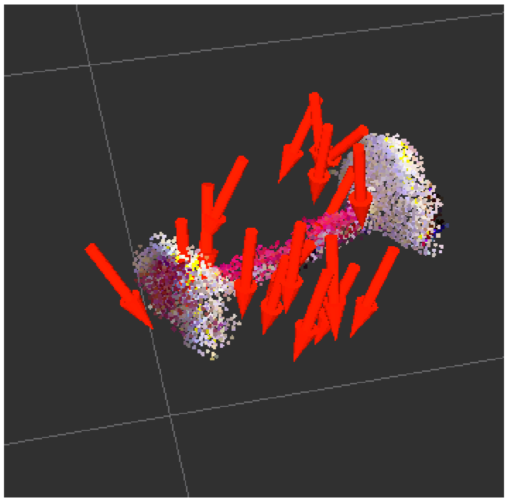


Figure 7: One object model with all of the associated grasp poses demonstrated from the user study.

6 Grasp Learning

Now that we have the ability to model and recognize objects in 3D, we turn to the problem of learning grasp poses. In this section, we first describe how grasps are mapped to the 3D model, and then present an outlier filtering algorithm that removes obviously erroneous grasps. We then present an online training algorithm that learns probabilistic success rates for the remaining grasps, to both determine the order in which grasps should be attempted and remove any erroneous grasps missed during the outlier filtering phase. The result is a database of object models with associated grasp poses ordered by success rate. Figure 8 provides examples of visualized output from each major step of the process.

6.1 Grasp Model

As described in Section 4, each point cloud sample recorded by our system also contains the associated grasp point. By creating a merged object model using the above process, we create a 3D object model with multiple associated grasps. We accomplish this by extending the registration algorithm described in Section 5 to combine the individual grasps into a list of grasps associated with the object model. Each pairwise registration within the algorithm produces a transformation matrix. Applying this transformation matrix to the demonstrated

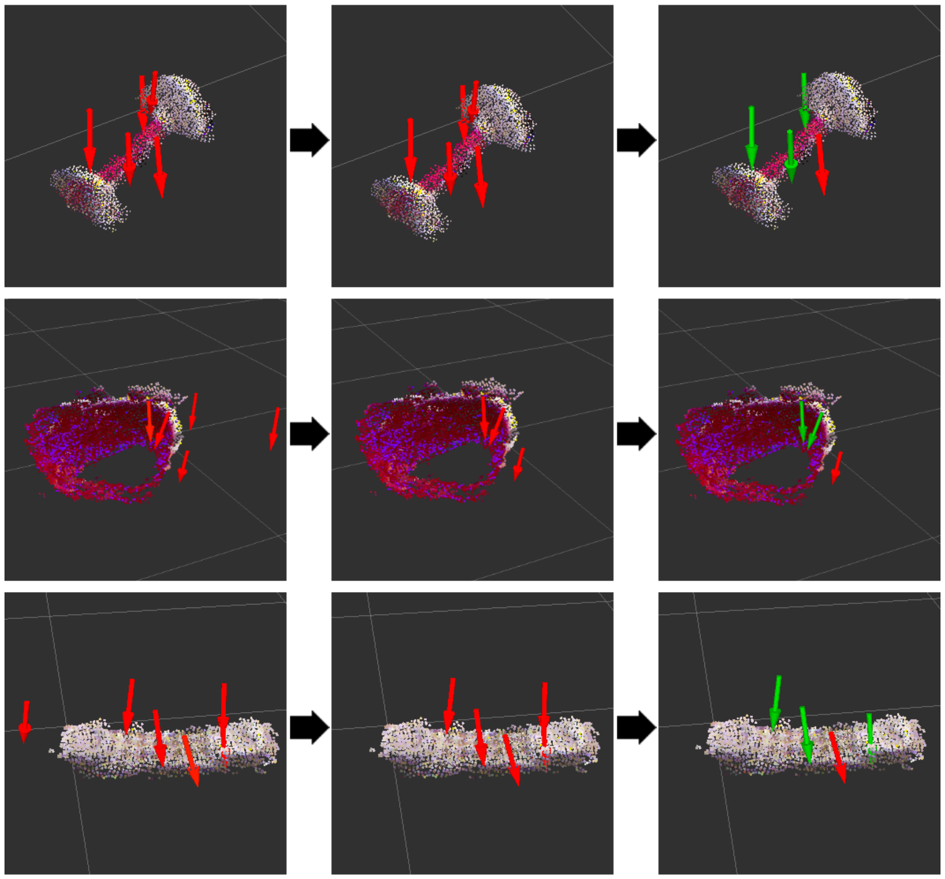


Figure 8: A visualization of the grasp learning pipeline for three object models. The figure shows only a subsample of the collected grasps to better facilitate visualization. Left: All grasp examples from data collection. Middle: Remaining grasps after outlier filtering. Right: Final set of grasps after online training, sorted into high-probability (green) and low-probability (red) grasps. All of these examples were generated from the user study data.

grasp pose at each pairwise registration step results in each object model also containing an associated set of grasps in its local reference frame. Figure 7 shows one of the object models generated from the user study, and further examples are shown in the left column of Figure 8.

6.2 Outlier Filtering

With the object models constructed, the next step of the grasp learning system focuses on evaluating the usefulness of the demonstrated grasps. This is a crucial step because demonstrated grasps can result in low quality grasp poses, due to poor object segmentation or grasps that accidentally nudge an object before lifting it up. This is particularly true when crowdsourcing grasps demonstrated by non-expert users, where quality of data cannot be guaranteed.

The grasp training algorithm, described further in the next section, also detects and removes unsuccessful grasps from an object model. This is an online algorithm that requires the robot to determine each grasp's quality through trial and error, and as such it becomes time consuming with large numbers of grasps. The goal of the outlier filtering phase is to eliminate as many grasps as possible with an offline algorithm, before the object models are passed through to the online grasp training phase.

With only a point cloud representation of an arbitrary object, it is difficult to determine the effectiveness of a grasp based solely on its location relative to the point cloud. To prevent the removal of potentially successful grasps, the only grasps that can be removed with high certainty are those that lie on the outside of the point clouds. Specifically, the algorithm fits a bounding box around the point cloud, and removes any grasps that are located outside of the bounding box at a distance greater than half the length of the robot's open gripper. These grasps will most likely miss the object entirely; this can be seen in Figure 8, where one outlying grasp is removed for both the bowl and the phone.

6.3 Grasp Training

The final phase of the grasp learning system learns probabilistic success rates for each grasp, which are then used to determine a grasp order for each object. An epsilon-greedy algorithm, Algorithm 2, learns these probabilities by testing each grasp repeatedly with the robot. The algorithm begins with a list of grasps to be tested, an empty list for storing sufficiently explored grasps, and by initializing the exploration variable, ϵ (lines 1-2). The algorithm continues to select and test grasps until every grasp associated with a model has been attempted a minimum of N times (lines 3 and 13). During each loop iteration, the algorithm selects grasps by either randomly exploring the set of untested grasps, or by selecting an already explored grasp with the highest chance of success (lines 4-9). The robot then performs the selected grasp, evaluates its success, and updates the success rate and number of grasp attempts accordingly (lines 10-12).

The value of ϵ determines the exploration strategy by which new or previously attempted grasps are selected. Higher values of ϵ result in more frequent exploration; conversely, lower values of ϵ result in more frequent execution of the previously explored grasps. The algorithm begins with a high value of epsilon, which decays exponentially (line 17) as the training continues. As such, the algorithm first spends most of its time attempting unexplored grasps, and as time goes on, it spends more time refining the success rates for the learned grasps. Over time, ϵ decays to zero, and further exploration of untested grasps will no longer occur. For larger grasp sets, such as a large crowdsourced dataset of grasp demonstrations, the algorithm can be adjusted to terminate after it finds a sufficient number of successful grasps, by changing the loop condition in line 3.

Upon completion of the training, the algorithm discards any grasps with a success rate of zero from the model. For example, Figure 8 shows one grasp removed from both the bone and the phone after the training algorithm determined that the grasps had no chance of success. The example also shows high-probability ($graspProbability > 0.5$) grasps in green. These grasps will be attempted first, and the low-probability grasps ($0 < graspProbability \leq 0.5$), shown in red, will only be attempted if all of the high-probability grasps are unreachable

Algorithm 2 Epsilon-greedy grasp training

```
Require: List<Grasps> $grasps$
 1: List<Grasps> $exploredGrasps$;
 2: $\epsilon = 1$;
 3: while $grasps$.size() $> 0$ do
 4:     $r =$ rand(0, 1);
 5:     if $r < \epsilon$ then
 6:         Grasp $g =$ pickRandom($grasps$);
 7:     else
 8:         Grasp $g =$ maxGraspProbability($exploredGrasps$);
 9:     end if
10:     Boolean $success =$ testGrasp($g$);
11:     updateGraspProbability($g$, $success$);
12:     $g$.numAttempts $++$;
13:     if $g$.numAttempts $\geq N$ then
14:         $exploredGrasps$.push($g$);
15:         $grasps$.remove($g$);
16:     end if
17:     $\epsilon = 0.975 * \epsilon$;
18: end while
```

from the current robot position.

The results presented below used training with N set to 3. Increasing N can increase the accuracy of the success rates, at the trade-off of increasing training time. To make up for this short training time, we continue to update the grasp model online after the dedicated training process is complete. This lifelong learning could lead to slow adaptability once the number of grasp attempts gets high, which could be improved by keeping grasp success data over a sliding window of previous attempts. For this thesis, we did not perform any long-term testing with the grasping database, and all results were gathered using this continual learning. We leave improvements for long-term use to future work.

7 Results

In this section, we present results of the user study and of the object recognition and grasp learning system. We first present the results regarding the effectiveness of the different feedback levels from each user study condition, to determine which method best facilitates data crowdsourced data collection. We then evaluate the usefulness of an object databases created using the system described in Sections 5 and 6 for both object recognition and object manipulation, based on both user study data and researcher demonstrated data.

7.1 User Study Evaluation

To determine whether the different feedback conditions improved participant performance, we analyzed the number of successful pickups, failed pickups, successful placements, and failed placements from participants in each condition. The most relevant measures for the user study were the total number of pickups, as this determines how much data each experiment contributed to the object recognition database, and the rate of successful placements, as this indicates how willing participants were to follow given instructions. Figure 9 shows plots of the number of pickups and rate of correct placement.

The pickup data has a trend suggesting that feedback can improve participant performance in a crowdsourcing situation. Not only do the two feedback conditions show participants outperforming the No Feedback condition, but participants in the Full Feedback condition had a greater number of successful pickups than participants in the Score Only condition. A similar trend appears in the rate of correct placement, although the differences between the Full Feedback and Score Only conditions are less apparent.

While there appear to be clear trends showing that interfaces with more feedback improve both the performance of participants and their willingness to follow instructions, we require more data to reach statistical significance. We leave the verification of these trends through the collection of more data to future work.

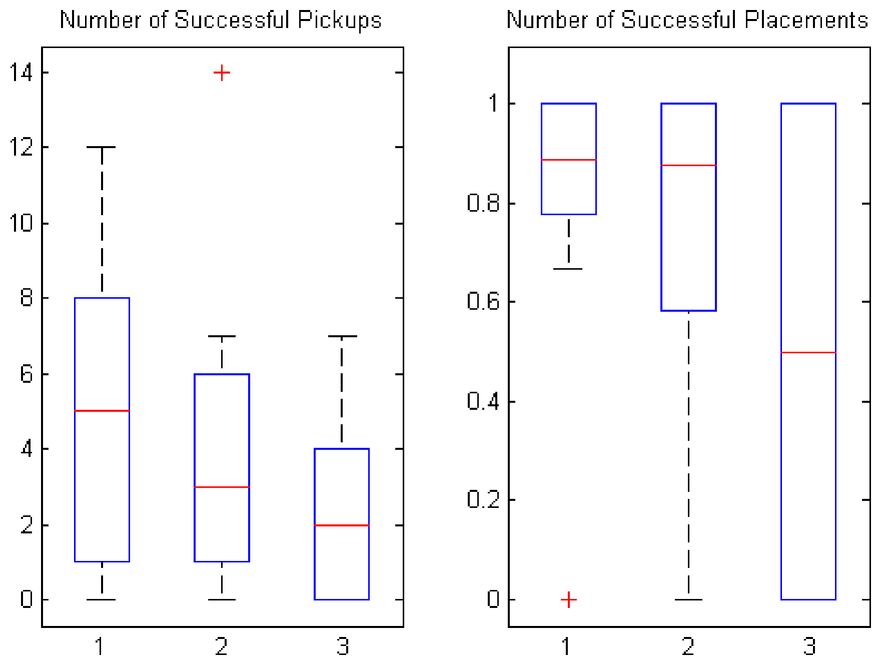


Figure 9: The number of successful pickups per participant (left) and the rate at which participants correctly placed objects within the boundaries of the randomized table section (right), organized by condition. The horizontal axis denotes the condition: (1) Full Feedback, (2) Score Only, and (3) No Feedback

7.2 Object Recognition Evaluation

In the following section, we present an evaluation of the object recognition database using the object recognition system described in Section 5.3. We evaluate the recognition success rate for both the set of objects used in the crowdsourcing user study, as well as a supplemental object set containing a greater number of varied objects.

7.2.1 User Study Object Set

The data collected from the crowdsourcing user study provided a set of point clouds for each object; Figure 10 shows the object set used in the study. The amount of point clouds collected varied in the range of 3 to 27 point clouds per object, as participants could pick up any of the objects as many times as they chose.

As mentioned in Section 5, the object model construction algorithm could have generated models without first organizing the data by object. This works in theory, since the learned decision tree can determine whether two point clouds belong to different objects as they will result in a failed merge. In practice, however, the potential for an incorrect registration between two different but similar looking objects creates a dangerous risk of error propagation, thus we deem the approach not robust enough for practical use. Further refinement to the decision tree could reduce this risk, but that is beyond the scope of this thesis.

To better demonstrate the generalizability of the object construction and recognition system, we evaluated the data using repeated random sub-sampling validation. For each iteration of testing, the point clouds for each object were randomly split into a training set (used for model construction) and a test set (used for object recognition). We performed five iterations of testing. Overall, the recognition system correctly classified objects at a rate of 88.7% +/- 5.1%. Figure 12 shows the confusion matrix for the set of objects, where entries along the main diagonal represent correct classifications and any other cell represents a misclassification. Figure 11 provides examples of object models generated by the object model construction algorithm.

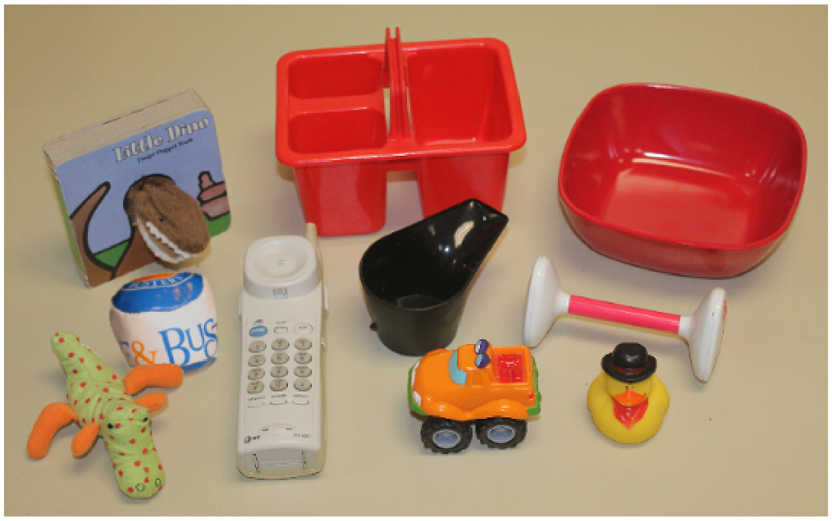


Figure 10: The ten household objects used in the user study.

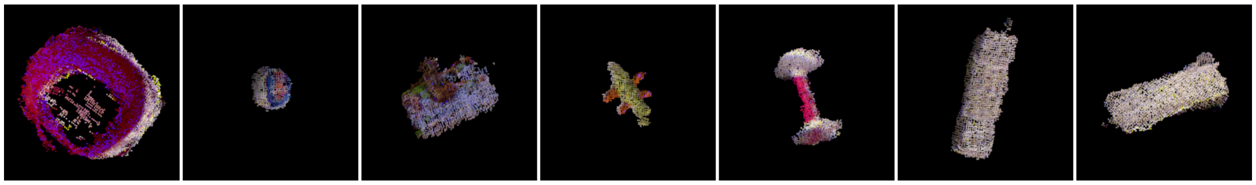


Figure 11: Examples of object models constructed from the user study data. Left to right: bowl, hacky sack ball, dinosaur book, dragon toy, plastic dog bone, phone (front view), phone (back view)

Object \ Classification	Ball	Basket	Bone	Book	Bowl	Cup	Dragon	Duck	Phone	Truck
Ball	0.680	0	0	0	0	0	0.040	0	0.280	0
Basket	0	1.000	0	0	0	0	0	0	0	0
Bone	0	0	0.743	0	0	0	0.029	0	0.229	0
Book	0	0	0	1.000	0	0	0	0	0	0
Bowl	0	0.053	0	0	0.905	0.042	0	0	0	0
Cup	0	0	0	0	0	1.000	0	0	0	0
Dragon	0	0	0	0	0	0	1.000	0	0	0
Duck	0	0	0	0	0	0	0.400	0.600	0	0
Phone	0	0	0	0	0	0	0.050	0	0.950	0
Truck	0	0	0	0	0	0	0.300	0	0	0.700

Figure 12: The confusion matrix for the user study data. The vertical axis consists of the actual object labels, and the horizontal axis represents the labels determined by the object recognition system.

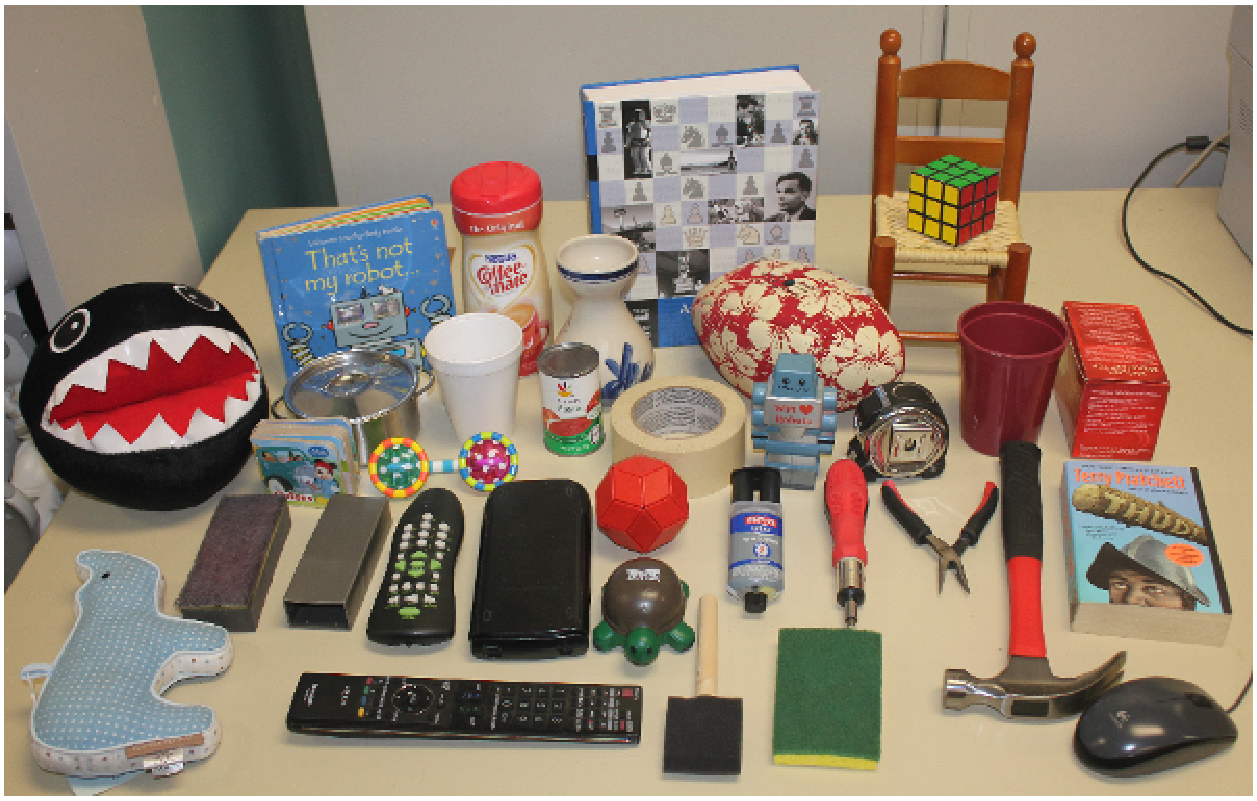


Figure 13: The 34 objects used to form the supplemental dataset.

7.2.2 Supplemental Object Set

In addition to the data collected from the user study, we collected supplemental data on a larger set of thirty-four objects, shown in Figure 13. This data did not include any grasping information, as it was collected to further test the object recognition system over a larger set of objects. We designed this data collection procedure to mimic the data collected in the user study. We placed each object in randomized poses and locations on the table, and stored 20 segmented point clouds captured from an Asus Xtion Pro depth sensor mounted above a table at a similar height and angle as the PR2's head-mounted Kinect. We analyzed the performance of the algorithm on this larger dataset to determine whether our system generalized well to a larger and more varied object set. Again after organizing the point clouds by object type, we tested the object construction pipeline and object recognition system on this dataset.

Before gathering results, however, we removed a few objects from the dataset when it became apparent that they could not be properly registered. This case occurred with small, reflective objects, e.g. a metal tape measure, or a glossy paperback novel. The issue with reflective objects is that depending on the lighting and their position and orientation on the table, the point clouds at each view were too significantly different to allow for accurate registration. We determined that five objects had to be removed from the set, leaving a total of 29 objects with which to complete the evaluation.

We tested this dataset using holdout validation, in which the dataset was split evenly into a training and testing set. The object model construction algorithm used the training set as an input to construct object models, and the object recognition system used the testing set to determine the recognition accuracy. Following this test, we switched the training and testing set so that all of the data could be used for both object construction and object recognition testing. Overall, the recognition system correctly classified objects at a rate of 83.93%, showing a comparable result to the recognition rate of the 10 object dataset from the user study.

7.3 Autonomous Grasping Evaluation

In the following section, we present an evaluation of the grasp learning system itself, comparisons of crowdsourced data to expert user demonstration, and a comparison of the demonstrated grasp database to a more traditional vision-based grasp planner. We also provide further examples of situations where the grasp learning system outperforms shape-based planners.

7.3.1 Evaluation of Crowdsourced Data

The user study provided us with a varying number of grasp demonstrations per object, shown in Table 2. Using the grasp learning pipeline presented above, we created a database of object models with associated grasps from the user study data. To evaluate this process, we performed an experiment in which the PR2 attempted to grasp the objects in randomized positions using the learned database for grasp planning. For a point of comparison, the

Table 2: Number of successful grasps demonstrated by user study participants.

Object	Basket	Bone	Book	Bowl	Cup	Dragon	Duck	Phone	Truck
Grasps	22	15	10	27	3	8	8	16	5

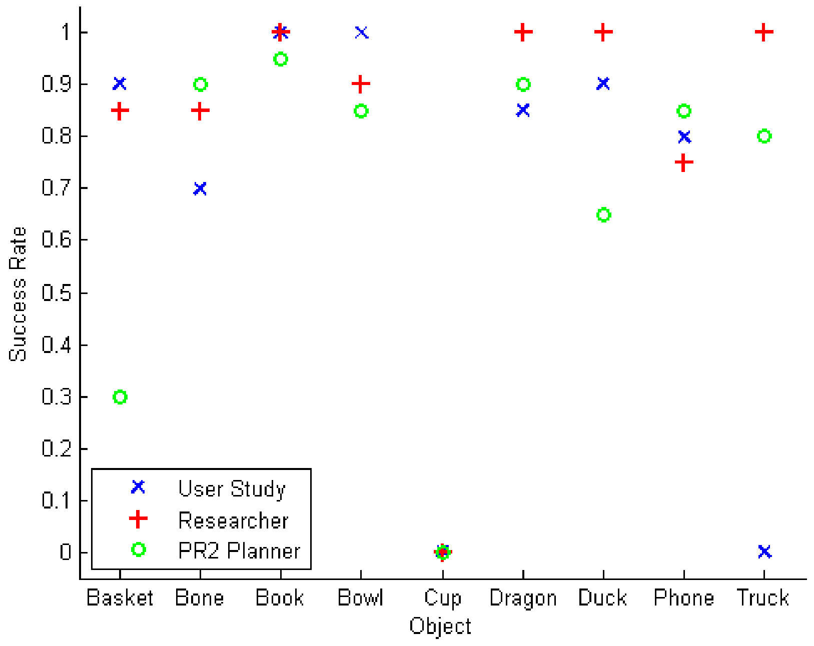


Figure 14: The grasping success rate, per object, for the database created from the crowdsourcing user study's grasp demonstrations, the researcher's grasp demonstrations, and the PR2's standard grasp planning algorithm.

researchers demonstrated grasps for the same object set, by using an offline equivalent of the web-based user study teleoperation interface implemented in rviz, a 3D visualization tool for the Robot Operating System (ROS). The researchers provided ten demonstration grasps for each object. Also, in order to evaluate the approach as a whole, we performed the same experiment using the PR2's standard grasp planning algorithm [16] instead of our database. The results are shown in Figure 14.

In general, the PR2 grasped the objects with a success rate of at least 80% using grasps learned from both the user study data and the researcher demonstrated data. Of note is the black cup, which failed under all three grasping systems. This occurred because the Kinect could not properly segment the object due to its reflectiveness and the lighting conditions of the experiment. Another outlier is the truck, which was rarely grasped successfully during the user study, and the algorithm could not construct a complete object model from the crowdsourced data. The final outlier of the user study data is the dog bone, which was grasped much less successfully using the user study data. This was likely due to the fact that the user study data produced a comparatively low number of high-probability grasps for this object, as shown in Figure 15. We suspect that with more crowdsourced data, the bone would be grasped as effectively with the user study data as with the researcher demonstrated data.

Comparing the demonstrated grasps to the grasps generated from the PR2's planner, we can see some interesting results. For many of the objects, the methods have little difference in success rate, and no method clearly outperforms the others. In a few specific cases, however, the learned grasps outperform the geometrically planned grasps. The first case is the basket, which has many potential grasp areas due to its subdivided sections, as well as the handle on top. The geometric planner had difficulty determining which grasp would be most effective, and often selected grasps that were blocked by the handle or the inner edges of the subdivided sections. The demonstrated grasps, however, often picked up the object by the handle on top, since that is where a human would naturally grasp the object. This grasp was more successful with the PR2, since the handle is in an open area not blocked by other parts of the object. The second case is the duck, which was simply too small for the PR2's planner to consistently analyze. The learned grasps worked well, though, since they required no planning. The final case is the truck, where the geometric planner occasionally failed due to grasping the moving wheels of the object, which caused the truck to slip from the gripper. Thanks to human knowledge of the truck's moving wheels, the demonstrated grasps did not have this problem.

Figure 15 shows the difference in quality between the crowdsourced user study grasps and the researcher demonstrated grasps. The ratio of high-probability grasps to total grasps

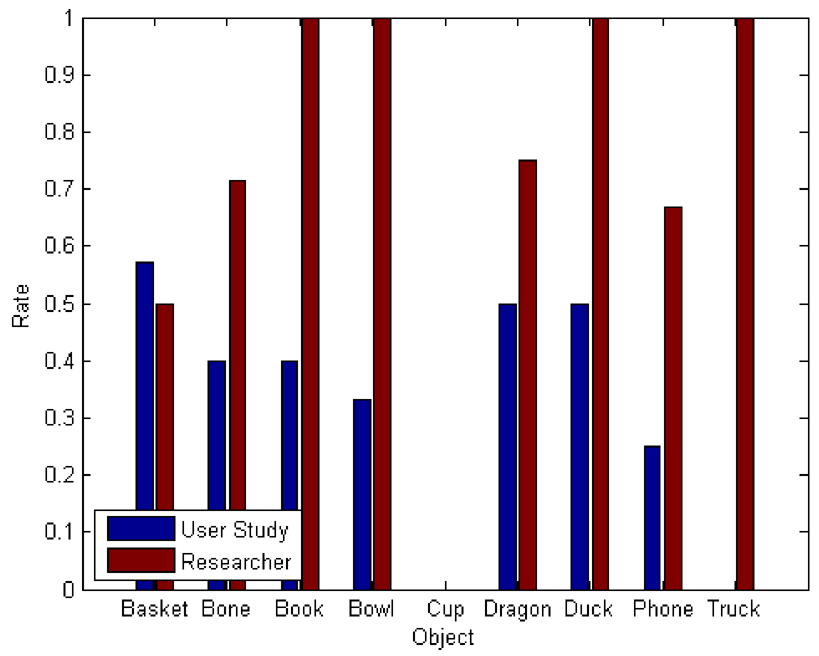


Figure 15: A comparison of the ratio of high-probability ($graspProbability > 0.5$) grasps to total grasps for the user study data and the researcher demonstrated data.

was significantly higher for the researcher demonstrated data than for the user study data. The graph shows that many more grasps were removed from the database by the grasp learning system for the user study data. The non-expert users could demonstrate high-quality grasps, but it often took more attempts than with the researchers' more consistent high-quality grasp demonstrations. This is not really an issue, however, as crowdsourcing is designed to collect large amounts of data, and we designed the grasp learning system to sort through the inconsistent data that crowdsourcing tends to produce. Also of note is that the researcher demonstrated grasps did not always have a success rate of 1.0. This likely occurred due to sensor error, and reinforces the need for the grasp learning process, regardless of whether the grasps were demonstrated by expert or non-expert users.

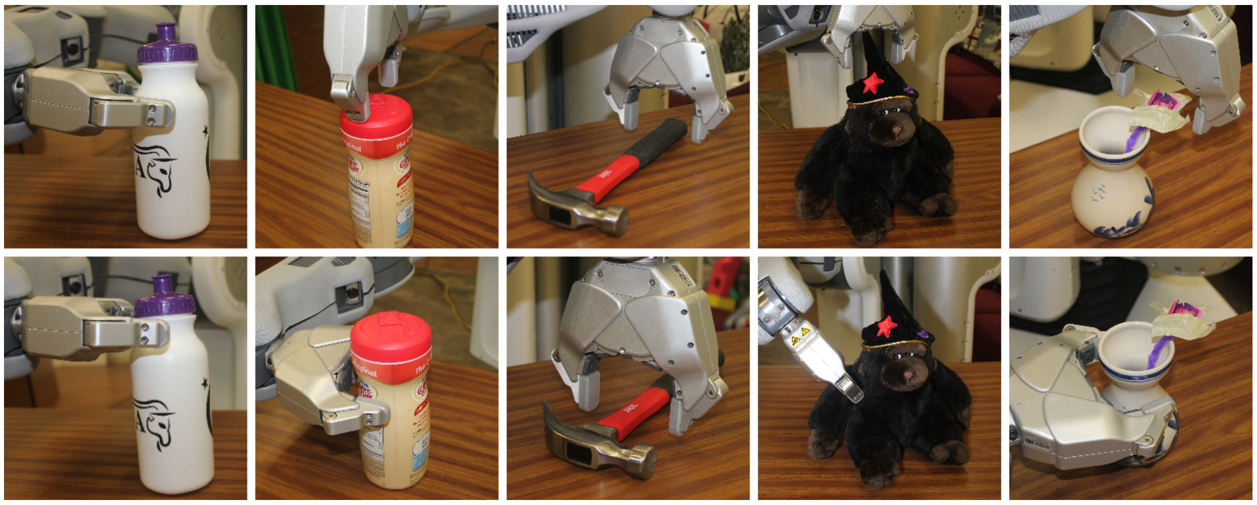


Figure 16: Example grasps calculated by a geometric grasp planner (top) and demonstrated by an expert user (bottom).

7.3.2 Advantages of Demonstrated Grasps

Following the results of comparing the demonstrated grasps to the PR2's geometrically calculated grasps, the researchers demonstrated additional grasps for a supplemental object set. Each object in this set represents a special case of constraints for grasping, which the geometric planner was unable to detect. Figure 16 shows each supplemental object with both an example calculated grasp and an example demonstrated grasp. The first object, the water bottle, is difficult for the PR2 to grasp because most of the bottle's surface is too slippery for the gripper to hold securely. There is a graspable region at the neck of the bottle, but the geometric planner would often grasp below it on the smoother surface. Next is the coffee creamer, which presents a challenge to the PR2 since its diameter is about the same distance as the width of the robot's open gripper, causing small miscalculations to result in missed grasps. The grasp planner favors grasps from above, but the object has a much more consistent grasp from the side. The hammer represents objects with extreme weight distributions, which slip from the gripper when the object is not grasped near its balance point. The monkey has a removable part, its hat, and again due to the geometric planner's preference for grasps from above, the PR2's planner picks up only the hat instead

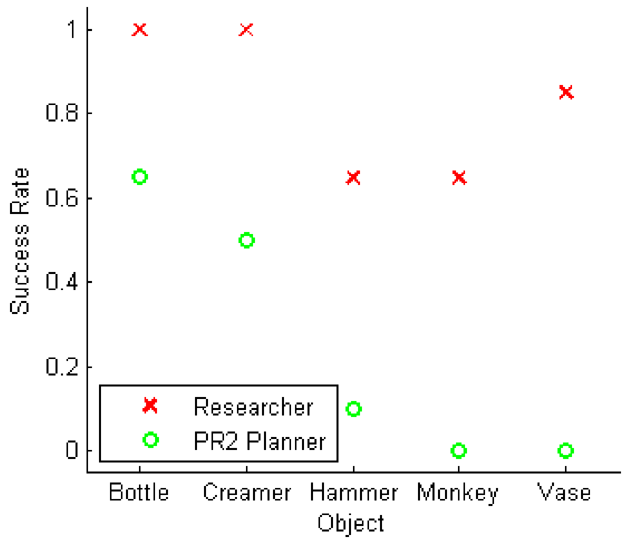


Figure 17: A comparison of grasp rates for the supplemental object set.

of the monkey. Finally, the vase represents objects containing fragile parts. The PR2's grasp planner always attempts to grasp the flower, which will both ruin the flower and fail to pick up the vase.

In each case, the learned grasps are superior to the geometrically calculated grasps for overall grasping success rate. Figure 17 shows a comparison for each object. For the water bottle and coffee creamer, switching to a grasp from the side at an appropriate height improved the grasp success rate to 100%; accounting for the balance point significantly improved the success rate for the hammer, which otherwise was grasped at random positions along the handle; for the monkey and the vase, the demonstrated grasps resulted in successes where the geometrically planned grasps could not succeed at all.

8 Conclusion

Our work seeks to overcome the drawbacks of existing techniques for generating 3D object recognition data sets for mobile manipulation. We presented a novel approach for constructing an object recognition database from crowdsourced information, contributing a framework for collecting data through the web, preliminary results of a study on the effects of incentives on online user behavior, a novel computational method for construction and recognition of 3D object models through iterative point cloud registration, and a system for learning the effectiveness of demonstrated grasps.

We validated the object recognition approach with data collected from a remote user study experiment, as well as with a larger supplemental dataset. The developed object recognition system was successful, correctly classifying objects at a rate of 88.7% for the experimental data and 83.93% for the larger supplemental dataset.

We showed that by using crowdsourcing, we can leverage human knowledge to create databases for autonomous object manipulation. We have shown that, with a sufficient amount of data collected and an appropriate system for determining which data is useful, non-expert users can demonstrate grasps that a robot can use to successfully manipulate household objects. Furthermore, this data can be collected using a minimal amount of setup, requiring nothing more than the robot and the object set. Due to the varied quality of crowdsourced data, this system requires more data collection than if expert users were demonstrating the grasps. The advantages of crowdsourcing mitigate this extra data collection, though, as it still requires less time and effort per individual user than if a researcher took the time to demonstrate all of the grasps on their own.

The presented techniques form a strong foundation for future research into mobile manipulation in real world environments. The user study and object recognition and grasp learning systems presented here are but one of many possible ways of using the new techniques that web robotics has to offer. The object recognition and grasping database built by this system is a strong example of the power of applying crowdsourcing to robotics.

References

[1] B. Alexander, K. Hsiao, C. Jenkins, B. Suay, and R. Toris. Robot Web Tools [ROS Topics]. *Robotics Automation Magazine, IEEE*, 19(4):20 –23, December 2012.

[2] Teresa C.S. Azevedo, João Manuel R.S. Tavares, and Mário A.P. Vaz. 3D Object Reconstruction from Uncalibrated Images Using an Off-the-Shelf Camera. In João Manuel R.S. Tavares and R.M. Natal Jorge, editors, *Advances in Computational Vision and Medical Image Processing*, volume 13 of *Computational Methods in Applied Sciences*, pages 117–136. Springer Netherlands, 2009.

[3] T. Baier and Jianwei Zhang. Reusability-based Semantics for Grasp Evaluation in Context of Service Robotics. In *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, pages 703–708, Dec.

[4] Cynthia Breazeal, Nick DePalma, Jeff Orkin, Sonia Chernova, and Malte Jung. Crowdsourcing human-robot interaction: New methods and system evaluation in a public environment. *Journal of Human-Robot Interaction*, 2(1):82–111, 2013.

[5] M. Brown and D.G. Lowe. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*, pages 56–63, 2005.

[6] Elisavet Chatzilari, Spiros Nikolopoulos, Symeon Papadopoulos, Christos Zigkolis, and Yiannis Kompatsiaris. Semi-supervised object recognition using Flickr images. In *Content-Based Multimedia Indexing (CBMI), 2011 9th International Workshop on*, pages 229–234, 2011.

[7] S. Chernova, N. DePalma, E. Morant, and C. Breazeal. Crowdsourcing human-robot interaction: Application from virtual to physical worlds. In *IEEE International Symposium on Robot and Human Interactive Communication*, Ro-Man '11, July 2011.

[8] C. Crick, G. Jay, S. Osentoski, and O.C. Jenkins. ROS and rosbridge: Roboticists out of the loop. In *th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 493 –494, March 2012.

[9] C. Crick, S. Osentoski, G. Jay, and O. Jenkins. Human and robot perception in large-scale learning from demonstration. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI 2011)*, 2011.

[10] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater. Learning grasp affordance densities. *Paladyn*, 2(1):1–17, 2011.

[11] Carlos Hernández Esteban and Francis Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367 – 392, 2004. Special issue on model-based and image-based 3D scene representation for interactive visualization.

[12] Willow Garage. The household_objects SQL Database, June 2010.

[13] C. Goldfeder, M. Ciocarlie, J. Peretzman, Hao Dang, and P.K. Allen. Data-driven grasping with partial sensor data. In *Intelligent Robots and Systems, 2009. IEEE/RSJ International Conference on*, pages 1278–1283, Oct 2009.

[14] Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K Allen. The Columbia grasp database. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1710–1716. IEEE, 2009.

[15] Christopher Harris. You're hired! An examination of crowdsourcing incentive models in human resource tasks. In *WSDM Workshop on Crowdsourcing for Search and Data Mining (CSDM)*, pages 15–18, 2011.

[16] K. Hsiao, S. Chitta, M. Ciocarlie, and E.G. Jones. Contact-reactive grasping of objects with partial shape information. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1228–1235, Oct 2010.

[17] Yun Jiang, S. Moseson, and A. 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, May 2011.

[18] Alexander Kasper, Zhixing Xue, and Rüdiger Dillmann. The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934, 2012.

[19] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824, 2011.

[20] A. Makadia, A. Patterson, and K. Daniilidis. Fully Automatic Registration of 3D Point Clouds. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1297–1304, June 2006.

[21] Niloy J. Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas. Registration of Point Cloud Data from a Geometric Optimization Perspective. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 22–31, New York, NY, USA, 2004. ACM.

[22] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann. Integrated Grasp Planning and Visual Object Localization For a Humanoid Robot with Five-Fingered Hands. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5663–5668, Oct 2006.

[23] B. Pitzer, S. Osentoski, G. Jay, C. Crick, and O.C. Jenkins. PR2 remote lab: An environment for remote development and experimentation. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3200 –3205, may 2012.

[24] John Ross Quinlan. *C4.5: Programs for Machine Learning*, volume 1. Morgan Kaufmann, 1993.

[25] R.B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3212–3217, May 2009.

[26] R.B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4, 2011.

[27] Ashutosh Saxena, Lawson LS Wong, and Andrew Y Ng. Learning Grasp Strategies with Partial Shape Information. In *AAAI*, volume 8, pages 1491–1494, 2008.

[28] Anuj Sehgal, Daniel Cernea, and Milena Makaveeva. Real-Time Scale Invariant 3D Range Point Cloud Registration. In Aurélio Campilho and Mohamed Kamel, editors, *Image Analysis and Recognition*, volume 6111 of *Lecture Notes in Computer Science*, pages 220–229. Springer Berlin Heidelberg, 2010.

[29] Alexander Sorokin, Dmitry Berenson, Siddhartha S Srinivasa, and Martial Hebert. People helping robots helping people: Crowdsourcing for grasping novel objects. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2117–2122, 2010.

[30] Jörg Stückler, Ricarda Steffens, Dirk Holz, and Sven Behnke. Efficient 3D object perception and grasp planning for mobile manipulation in domestic environments. *Robotics and Autonomous Systems*, 61(10):1106 – 1115, 2013. Selected Papers from the 5th European Conference on Mobile Robots (ECMR 2011).

[31] S. Tellex, T. Kollar, S. Dickerson, M.R. Walter, A.G. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, August 2011.

[32] Russell Toris, David Kent, and Sonia Chernova. The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *Journal of Human-Robot Interaction*, 2014.

[33] Antonio Torralba, Robert Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1958–1970, 2008.

[34] Zhixing Xue, A. Kasper, J.M. Zoellner, and R. Dillmann. An automatic grasp planning system for service robots. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6, June 2009.