AN ABSTRACT OF THE THESIS OF

Taylor M. Finley for the degree of Master of Science in Robotics presented on December 5, 2014.

Title: A Shared-Autonomy Heavy-Lift Assistive Robot for Manufacturing.

Abstract approved:

____

William D. Smart

Automating pick-and-place operations for large steel castings in a foundry is challenging, particularly from a perception point of view. Accurately estimating the identity and orientation of a given part is an open and complex problem, especially if that part sits in a bin of very similar parts. The problem is made worse when the castings are made in small mixed batches, where many parts appear very similar from a variety of viewpoints.

In this dissertation, we investigate automating the identification and pose estimation of large steel castings in a realistic foundry setting. We look at both fully-automatic and human-assisted techniques, and compare their effectiveness. We report the results of testing our system with foundry workers, wearing full personal protective equipment, in their everyday work environment. By using human guidance in our system we were able to raise part recognition rates from 40% to 98%, and were able to estimate the pose of parts to within a few degrees 86% of the time.

A Shared-Autonomy Heavy-Lift Assistive Robot for Manufacturing

by
Taylor M. Finley

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented December 5, 2014
Commencement June 2015

ACKNOWLEDGEMENTS

I would like to thank my advisor, Bill Smart, for his guidance and support throughout my time at Oregon State. He exposed me to different paths when it was needed, but allowed me the freedom to explore things on my own. I would like to thank the members of the Personal Robotics Lab at Oregon State for their help over the last two years. It was a welcoming environment from day one and it was a pleasure to work with everyone on a daily basis. I would also like to thank ESCO Corporation for funding my program. Without their support, this would not have been possible. My colleague, Ryan Carpenter, was also a pleasure to work with over the last two years. Thanks, Ryan. Last, but for sure not least, I would like to thank my loving wife, Megan. I would not have been able to accomplish this without her support and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

1 Introduction

1.1 Current State and Motivation

The foundry industry is labor intensive. The automation of labor-intensive processes can increase safety and reduce the overall costs of production. However, automation solutions are not typically feasible in a foundry process with low batch sizes and a large product library being manufactured in a single production line. This type of manufacturing environment causes automation flexibility challenges. The traditional industrial robotics method of hard fixturing all parts is not possible with a large product offering through a single production line. In many scenarios within a foundry, the cost and tooling management does not make a robotic system financially feasible [1].

The finishing process of steel castings is major source of both labor costs as well as ergonomic issues. The physical manipulation of castings during the finishing processes is a waste and an automation opportunity to increase both efficiency and safety. The current finishing process uses an overhead crane to lift parts from a pallet or tub onto a workbench as previously discussed. The casting is raised, manually positioned over the table by pushing the hanging casting and then lowered onto the workbench shown in Figure 1.

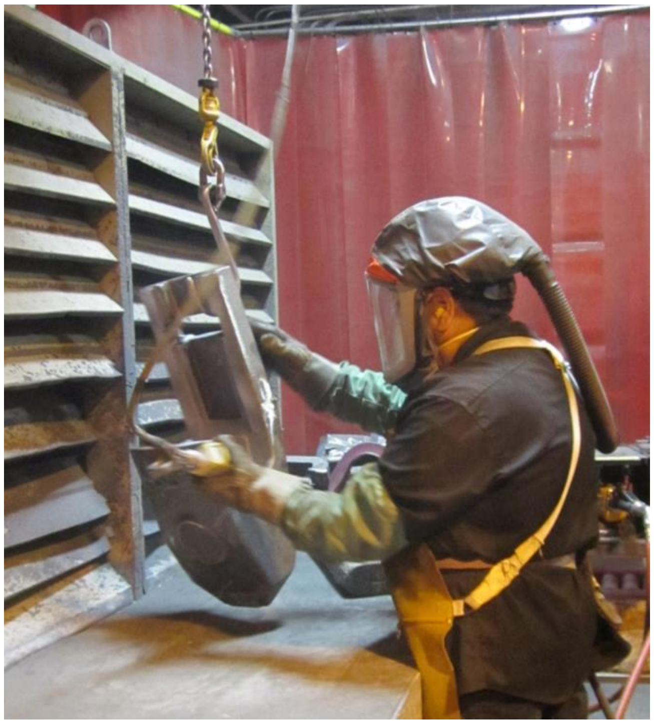

Figure 1: Manual casting manipulation using an overhead crane while wearing personal protective equipment

The worker must position the casting in an ergonomic manner to perform their task. This manual manipulation and casting setup often involves straps, wooden blocks and wedges. The irregular shapes of the casting make this task a serious ergonomic challenge and create a risk to hand injury. The castings are required to have a single lifting eye if they weigh more than 40 lbs. The lifting eye is generally located in line with the center of mass to allow for an orientation conducive for installation by the end-user. However, this single lifting eye does not make it easy when the worker needs to position the casting in multiple orientations on the workbench. The worker needs to position the casting in different orientations to gain ergonomic access to accomplish their task during the finishing process. A time study of a few different parts showed that casting manipulation and repositioning accounted for 45% of the pitch time for the finishing process. This is a significant waste.

A heavy-lift assistive robot will reduce this part manipulation waste. The heavy-lift assistive robot concept is diagramed in Figure 2. The robot grasps the part off of a pallet while the worker was outside of the work zone. In order to comply with safety standards, the work zone uses an area scanner or light curtain to ensure the worker is not in the work zone while the robot was moving. Then the robot presents the part in an ergonomic and stable position for the worker. The worker then enters the work zone, which triggers the robot into a safe and locked-out state. The worker then performs their task. After the task is completed, the worker steps outside the work zone and indicates to the robot to either change orientation or to place the casting back on the pallet. The whole process is completed without the worker having to manually manipulate the casting.

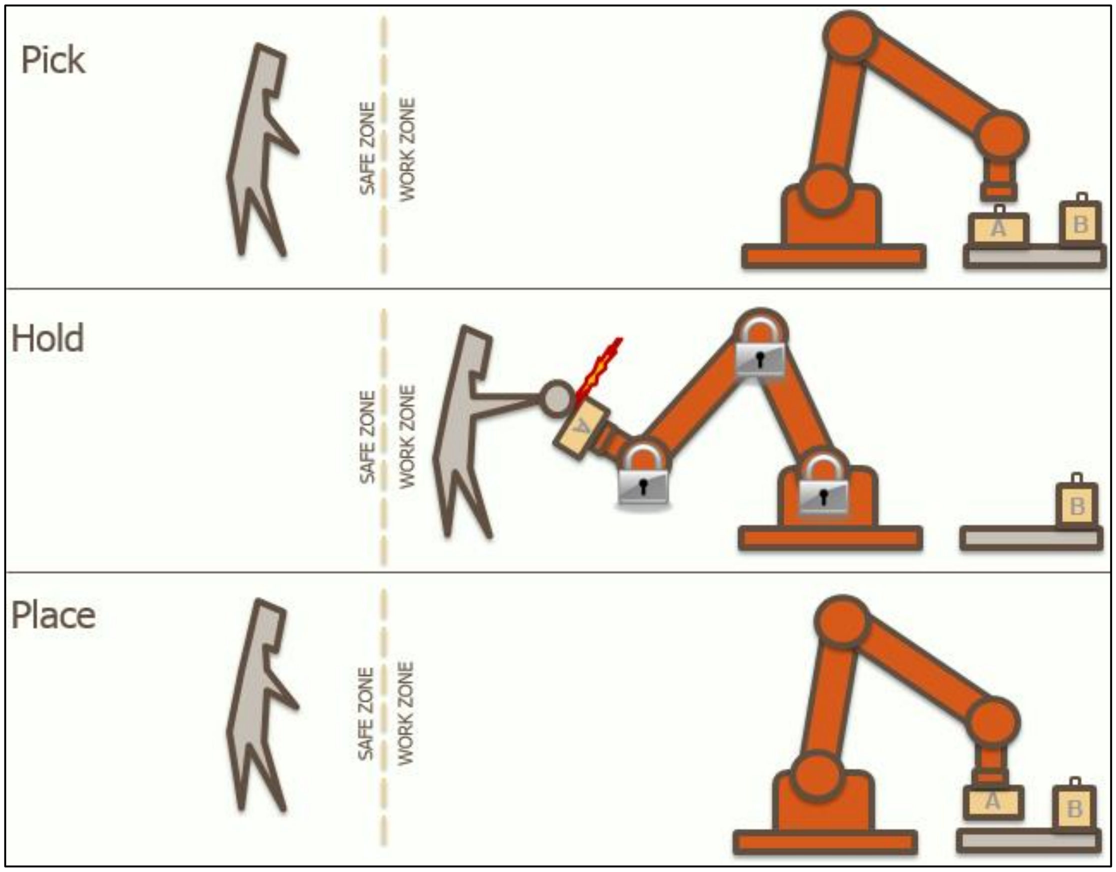

Figure 2: Heavy helper concept process

This heavy helper concept can be broken down into two systems. There is a pick and place system integrated with a work-holder system. The work-holder system can be accomplished with the industrial robot controller. Additional features such as customizable ergonomic settings per worker could be added with off-the-shelf solutions. The pick and place system is more difficult. New sensor technology is allowing for industrial robots to be intelligent and sense their environment. In some cases, this allows for the elimination of fixturing by implementing the necessary feedback into a closed-looped system. This feedback is typically derived from of a 2D or 3D camera with computer vision programmed into the system for decision making. The task of picking up a part in a batch or randomly oriented parts is known as bin-picking. Bin-picking with industrial robots is a challenge many researchers are developing solutions for [2]–[4]. Off-the-shelf solutions are available, but are expensive and require extensive engineering programming time for each part [5]. A large product library makes this option less attractive. Also, it restricts the scope of the heavy helper concept to the functionality of off-the-shelf solution.

A custom solution allows for an unrestricted scope but can be technically challenging or even unachievable. A good option for a custom solution is to use open source tools to shorten the project timeline. Often open source solutions are state of art technology that require little to no research and development cost. This option shifts the project workload to the research and development of extending technology rather than recreating it. We aimed our research efforts at an open source solution for a pick and place system that would have the intelligence to adapt to low batch sizes and a large product library [6]. We were able to identify open source tools for robot hardware control as well as and integrated robot perception solution.

1.2 Prior Work

1.2.1 Robot Operating System (ROS)

The Robot Operating System (ROS)[7] is an open source software framework for programming and operating robots. It has grown into the standard for robot research and development within the community of researchers. The open source development process was well thought out when the framework was established. It is a powerful communication protocol of topics and message that are communicated between ROS programs through a central core allows for modularity in the use of different robotic subsystems. For instance, one developer can quickly reuse a package written by another developer by identifying the communication topic. Others have attempted to establish a standard operating system for robots, but ROS ultimately prevailed as the dominant standard [7]–[9].

ROS was originally named Switchyard when it was developed by Morgan Quigley at Stanford University in the Stanford Artificial Intelligence Laboratory. Swtichyard was later renamed ROS at Willow Garage and the first official release was in 2010. In 2013, ROS found a stable home at the Open Source Robotics Foundation (OSRF) [7].

Within an hour a new user can install and configure ROS on a Linux platform. A few minutes after that they can be teleoperating a simulated robot with the help of online tutorials. That same teleoperation program that is used to control a simulated robot can later be used to control a physical robot. ROS can be used for all different types of robots. It doesn't matter the shape, size, number of arms, legs, wheels, or sensors. ROS is a very capable platform which led us to investigating its capabilities with industrial robots.

1.2.2 ROS-Industrial

The team at Southwest Research Institute (SWRI) in San Antonio, Texas recognized the power of ROS and decided to extend this capability to industrial robots. Industrial robot manufacturers have typically locked down their controllers to keep their functionality and communication proprietary. However, the team at SWRI convinced the robot original equipment manufacturers (OEM) that the benefit of opening their controllers up to communication to ROS would be beneficial to their bottom line. In exchange for opening their controllers, the OEM would essentially be receiving free R&D for their products and extending their market by adding optional functionality to their products [10].

Industrial robot controllers are a black box of proprietary hardware and software. This subset of ROS by SWRI became known as ROS-Industrial. It aimed to open this black box, or at least bypass it after getting the approval from the robot OEM. ROS-Industrial turned the black box controller into a slave and gave control of the robot to a computer running ROS. By using the backbone of ROS, ROS-Industrial was built on a well-established base. It was able to utilize the robust architecture and program availability that is being actively developed by the ROS community.

ROS-Industrial aims to tackle problems that are seen in real production environments. In order to accomplish this, they are working to partner with companies in industry in a manner that allows them to collaborate and solve actual issues such as cast part material handling [11]. These partnerships formed into consortium of research institutes and industrial companies working with ROS-Industrial. The consortium serves as a steering committee as well as a research partnership opportunity for many of its members [7].

1.2.3 Point Cloud Library (PCL)

The Point Cloud Library (PCL) is an open source toolkit for processing 3D point clouds that is well integrated with ROS. The toolkit “contains state-of-the art algorithms for: filtering, feature estimation, surface reconstruction, registration, model filtering and segmentation” [12]. The PCL object recognition pipeline is a powerful set of algorithms to sense and determine the world using 3D point clouds: keypoint detection, feature extraction, descriptor matching, correspondence grouping, pose estimation, pose refinement, and hypothesis verification [13]. This recognition pipeline is supported through online tutorials and forums. It gives the user the ability to trial different configurations and test which one works best with their scenario. For instance, their global recognition framework program is preprogrammed to use different classification algorithms for its feature extraction: ensemble of shape functions descriptor and the viewpoint feature histogram descriptor.

The viewpoint feature histogram (VFH) is a descriptor that uses a histogram to describe the set of angles between the point cloud cluster's centroid and the normal estimated for each point. The method is able to describe a point cloud using 308 bins in a histogram [14]. This allows for the classification to be much less computationally expensive when using this descriptor to classify a point cloud.

The ensemble of shape functions (ESF) is another descriptor used by the global recognition framework program. It also uses a histogram as a descriptor. The 64-bin histogram consists of angle, point distance, and area shape functions values that used to describe the point cloud. Unlike VFH, the descriptor does not require preprocessing, such as normal estimation [15].

1.3 Project Structure

Our project was conducted in two phases. The initial phase focused on a fully autonomous solution for a pick and place operation. It relied on the success of a part recognition and six degree of freedom pose estimation program. After testing, we determined that this recognition program was not reliable enough to meet our needs. This led us to our second phase, which focused on a shared autonomy approach to a pick and place operation. The approach incorporates the worker into the system as a key component to one or more tasks in a semi-autonomous system. In this phase, we developed a system with the worker integrated in the decision making process of part and pose recognition. We designed, conducted and analyzed a user study to test the functionality and intuitive design of the system. We then performed physical testing with an industrial robot. This paper will discuss both of these phases of the project and close with project conclusions and future work suggestions.

2 Phase One – Fully Autonomous Pick and Place

2.1 Scope Definition and System Requirements

The initial system design started with a scope definition. The scope was focused on the automated pick and place task for the heavy helper concept as depicted previously in Figure 2. An operator was available to troubleshoot the system if the system encountered an error, but the system was designed to be fully autonomous. The end of arm tooling for the robot was not a part of the scope. The system requirements for this scope were as follows:

1. Determine the identity of steel castings on a pallet.
2. Estimate the pose of the identified casting.
3. Execute a grasp plan with an industrial robot.
4. Provide a human intervention interface for error troubleshooting.

2.2 System Design

The system was designed to use ROS as its backbone. The ROS-Industrial packages allowed for the external control of the industrial robot. Open Natural Interaction (OpenNI) was a non-profit organization that established a framework for communication with open source devices. Using an OpenNI-compliant device for our system allowed point clouds to be easily captured and processed with standard programs within ROS. There were two OpenNI-compliant devices that were readily available: the Microsoft XBOX Kinect and the Primesense Carmine 1.08 [16]. We performed an accuracy test to determine the proper camera for our system. We created a calibration tool using two spherical balls at the end of a carbon fiber rod (Figure 3).

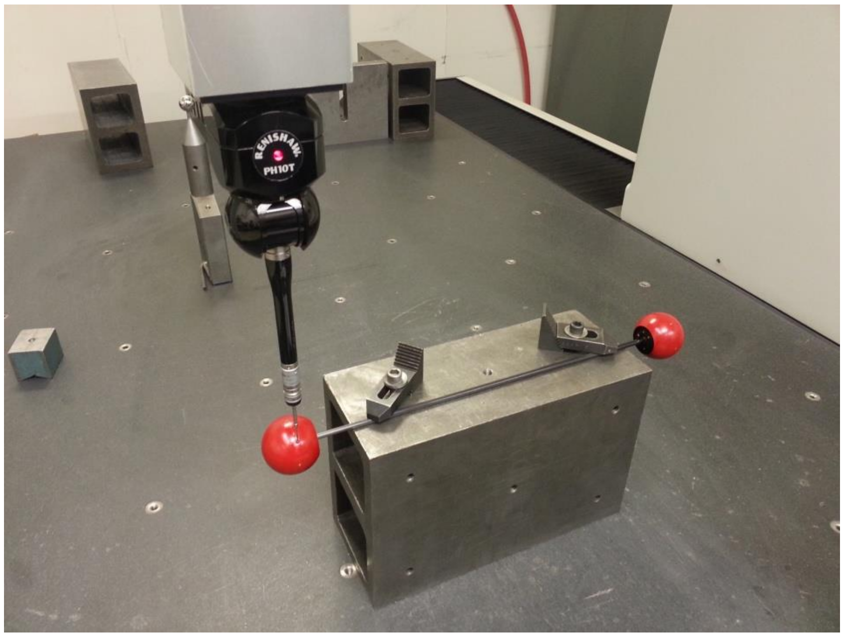

Figure 3: Calibration of the calibration rod

We measured the rod with a coordinate measuring machine to confirm its roundness as well as determine the exact distance between spheres. We then used our test robot to position the spheres in various locations within the working area of the robot (Figure 4).

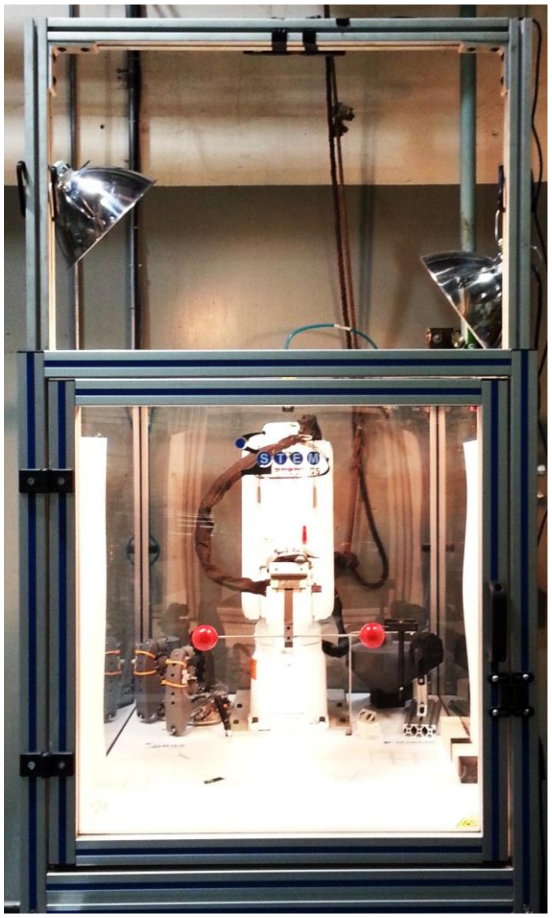

Figure 4: Camera testing setup

For each position we measured the distance between the spheres using the different cameras. The measurements were performed by using a color segmentation algorithm to create two clusters. The color segmentation process is similar to a passthrough filter. Each point in the point cloud has not only has 3 values for its position (X,Y,Z), it also has a 3 values for its color (R,G,B). We adjusted the filter to allow for a range color values that were captured on the red spheres. We then used the RANSAC [17] sphere shape detection to determine the locations. The RANSAC algorithm randomly selected 3 points in the point cloud and constructs a sphere. It then determined how many points fall within a range of that sphere. For each cluster, we used 500 iterations within the algorithm to identify the best fit sphere [17]. We determined that both cameras had an equal accuracy of ±1.8mm when determining the location of two objects within the work envelope of our workcell. Shortly after the test, the manufacturer of the Primesense Carmine stopped manufacturing the sensor. We decided to move forward with the Microsoft XBOX Kinect for its popularity and support within the open source community [16], [18].

The PCL 3D object recognition framework was chosen to use as a guide for the part recognition and pose estimation subsystems. The PCL framework nicely integrated with standard ROS programs [7]. RVIZ, a visualization tool of ROS, was selected as the operator interface for troubleshooting [19]. The PCL recognition framework gave us the ability to test both VHF and ESF classification as previously discussed [20].

As the part recognition program was being created, it was tested using sample castings. The initial results were not impressive. We wanted to investigate the part recognition performance in a scientific manner before proceeding with building the system. We did not want to build a system that relied on the part recognition, only to have it fail after the entire system was completed.

2.3 Part Recognition Testing

The primary goal of the initial testing was to determine if the object recognition solutions provided by PCL were capable of properly identifying a set of castings. To make the identification challenging and as real as possible, 6 castings were chosen from the same part family. This meant that the parts were of similar shape and size. Specifically, 3 part shapes and 2 sizes were chosen to determine if size, shape or both could cause issues in the part recognition. The back half of each shape is similar and the size varies between 229 mm long on the largest to 176 mm long on the smallest (Figure 5).

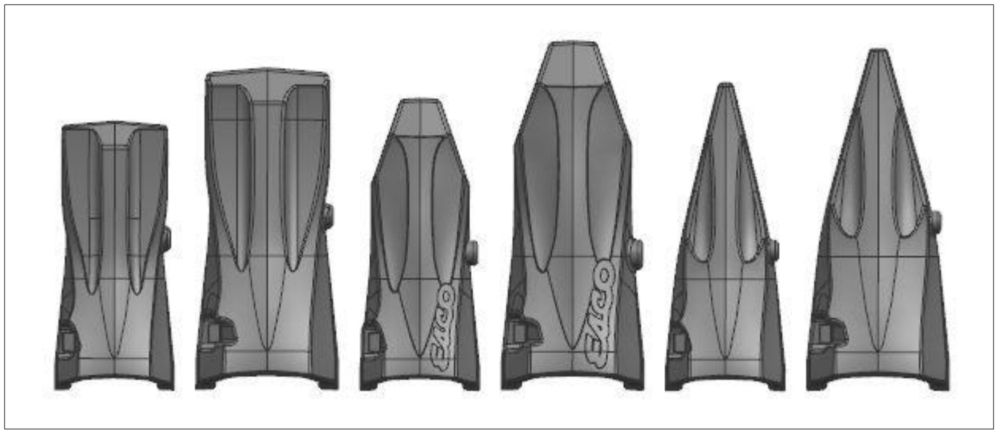
Figure 5: Sample of casting used for part recognition initial testing. From left to right: U20S, U25S, U20C, U25C, U20P, and U25P.

To eliminate the accuracy of the 3D camera as a factor in the performance of the system, a more accurate camera was used to gather data. Point clouds of each casting were obtained using a Steinbichler structured light scanner. Its accuracy is approximately ±0.02 mm which is around 100 times more accurate than the proposed scanner, the Microsoft XBOX Kinect. The castings were set up on turntable in front of the camera which was mounted on a tripod (Figure 6). 3D snapshots were taken at different angles to simulate a random orientation in front of the robot (Figure 7). A single piece setup was also chosen to eliminate any occlusions which would impact the ability to easily segment the individual part.

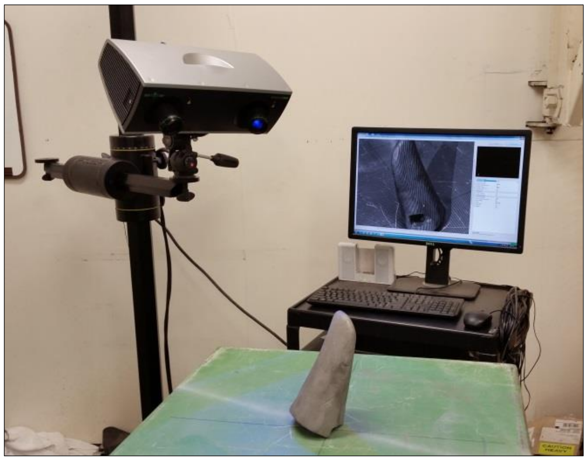
Figure 6: Data capture setup using a Steinbichler structured light scanner

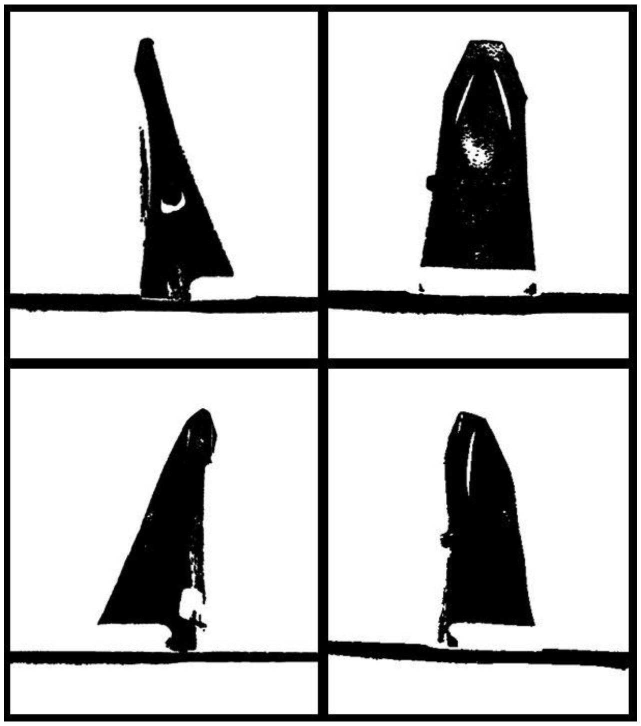
Figure 7: Example snapshots of 3D data for object recognition testing

The 3D images were converted into point cloud data (pcd) files which is the native file format for PCL programming and processed [21]. The point clouds were then processed to segment the casting from the table using the RANSAC plane segmentation and cluster extraction algorithm. The RANSAC plane segmentation works very similar to the sphere recognition as we discussed earlier. Instead of fitting a sphere to random point, it fit a plane. Then all inliers within a specified distance to the plane were removed from the scene. The cluster extraction algorithm used a search method to identify groups of points and segment them into individual point clouds [17], [22].

The recognition program needed a library to reference in order for the casting to be identified. We used the Steinbichler scanner to create a complete surface point cloud of each casting. These point clouds were used to build the reference library. The library was built by taking snapshots around point cloud from different angles. The recognition program used this library to compare and identify the closest match from the library. There were two different descriptors used in the recognition program. These two descriptors that were trialed were ESF and VFH as previously discussed.

The PCL open source program example was setup for live streaming. The program was reconfigured to systematically load pcd files captured from the Steinbichler scanner. The program loaded each of the snapshots and compared it to the library to recognize the part. The two algorithms with the two different descriptors were run using this same data and a report was generated.

2.4 Part Recognition Results

Table 1 shows the part recognition using the VFH method. We created 32 point clouds from a different viewpoints around the casting. The actual target casting is represented in the rows of the table. The values in the cell represent the number of times the part was recognized as the parts labeled in each column. Successful match percentages are shown in the grey cells along the diagonal. The average part recognition success rate for the VFH classifier in our test was 47.3%. VFH results from initial part recognition testing. Grey cells represent a correct match (higher is better).

Table 1: VFH results from initial part recognition testing. Grey cells represent a correct match (higher is better).

VFH		Part Recognition Estimation					
		U20C	U20P	U20S	U25C	U25P	U25S
Actual Part	U20C	53.1%	3.1%	6.3%	21.9%	6.3%	9.4%
	U20P	3.1%	59.4%	0.0%	9.4%	25.0%	3.1%
	U20S	9.4%	0.0%	53.1%	9.4%	6.3%	21.9%
	U25C	28.1%	6.3%	3.1%	53.1%	0.0%	9.4%
	U25P	12.5%	37.5%	6.3%	18.8%	28.1%	0.0%
	U25S	0.0%	3.1%	40.6%	9.4%	3.1%	43.8%

The same 3D point clouds were processed using the ESF classification method and the results are shown in Table 2. The average part recognition success rate for the ESF classifier in our test was 31.3%.

Table 2: ESF results from initial part recognition testing. Grey cells represent a correct match (higher is better)

ESF		Part Recognition Estimation					
		U20C	U20P	U20S	U25C	U25P	U25S
Actual Part	U20C	21.9%	18.8%	21.9%	9.4%	12.5%	15.6%
	U20P	21.9%	9.4%	6.3%	3.1%	50.5%	9.4%
	U20S	0.0%	6.3%	50.5%	6.3%	12.5%	25.0%
	U25C	12.5%	3.1%	21.9%	12.5%	25.0%	25.0%
	U25P	12.5%	6.3%	12.5%	9.4%	40.6%	21.9%
	U25S	0.0%	3.1%	28.1%	12.5%	3.1%	53.1%

2.5 Part Recognition Discussion

The VFH classification success rate was greater than the ESF classifier, but well below an acceptable level. It was evident that the accuracy of the 3D camera did not have an impact. The results achieved with the Steinbichler scanner were not materially better than those achieved with the Kinect in early testing. From the results of the U25S, both classification methods had problems with differentiating size, but did an acceptable job differentiating shape. However, the results of the U25P show a poor result of differentiating both size and shape for both of the classifiers.

As we discussed earlier, we could not move forward with an autonomous system that depended on the success of a part recognition system to be reliably accurate. While we knew that there were some off-the-shelf systems that have proven capable of part recognition at a reliable rate, they were not available as an open source option. This caused us to rethink our approach.

3 Phase Two – Shared Autonomy Heavy Helper

3.1 Shared Autonomy Investigation

If a fully autonomous solution for a pick and place operation was not feasible, we explored alternative solutions involving the worker as an integrated part of the system. Typical systems are either fully autonomous or fully teleoperated systems [23]. We didn't want to depend on the workers ability to teleoperate our system. We also didn't want to depend on a worker for failure recovery if the main recognition program of the system was unreliable [24]. We looked at where a human workers perform well and where they perform poorly [25]. We also then looked at where the computer would perform well and where it would perform poorly. Many roboticits have investigated the shared autonomy approach with robot perception [23], [24], [26], [27]. With this new thinking of a shared autonomy approach, we looked to narrow the scope and determine the appropriate autonomy level [26], [28], [29].

3.2 Scope Definition and System Requirements

We narrowed the scope of the project to include only casting with lifting eyes. This simplified the requirements of the perception system by only requiring it to identify the pose of a known geometric shape. Because most lifting eye geometry is standardized, this reduced the scope even further. However, even with a known shape to look for, the perception system still needed to make a few decisions. This included identifying the rough location of the lifting eye. The system also needed to know which part the operator wanted to work on next. The alignment algorithm was not able to determine the part orientation from symmetric geometry of the lifting eye. The orientation decision was dependent on the worker. The worker identified the orientation and communicated it to the system. The key to these decisions was to utilize the capabilities of the worker in this shared autonomy system. The challenge in the design of this shared autonomy system was making it simple for the operator to communicate with the robot in an intuitive interface. It was important to distribute task responsibilities according to strengths of the worker and the computer. IOt was seen that workers were good at identifying castings and deciding on which part to work on next. They were also good at estimating where the lifting eye is on the casting. However, they were not good at identifying the exact location and orientation of the lifting eye with respect to the robot. This is where the perception system performed well. The computer vision algorithms were very good at identifying an exact pose of an object (with good reference geometry), however it needed a rough estimation of where to start. Computer vision has come a long way with initial estimation by using such tools as RANSAC to randomly guess and check. While these tools are available, they system was more robust when the worker provided the estimate.

After the conclusion of phase one, the reinvestigation, and the research into shared autonomy, a new set of system requirements were established:

1. Provide a live visual of the workspace in front of a robot for the user to identify the appropriate part.
2. Provide an intuitive communication interface for the user to communicate the approximate location of the lifting eye of the part identified.
3. Estimate the 6 degree of freedom pose of the lifting eye on a casting selected by user.
4. Provide intuitive communication to adjust the orientation of the pose generated by the computer vision system.
5. Provide intuitive communication to confirm or reject the computer vision's pose output.
6. Physically move the industrial robot to grasp the lifting eye.

3.3 System Design

The system was designed around ROS for its open yet robust platform that allows for the communication necessary in a multi-subsystem solution. As seen in Figure 8, the system was a non-sequential mixture of different subsystems. These subsystems were made up of one or multiple programs that communicate using the ROS platform. These programs are referred to as nodes by ROS. These nodes communicate with one another through a central hub called roscore. The powerful feature of ROS's communication structure is that a node does not care which node produces the communication message as long as it is in the proper format coming from roscore. This allows for nodes to be swapped out with little reconfiguration. This makes repurposing nodes amongst the open source community much easier. For instance, we did not have to write our own 3D camera node. We were able to write the computer vision node such that it was not dependent on the source of the point cloud. This allowed us to reuse previously written programs that are shared amongst the ROS community. It also allowed us to write an offline program to simulate a live feed from a camera. All ROS nodes were written to operate and communicate with the Hydro version of ROS.

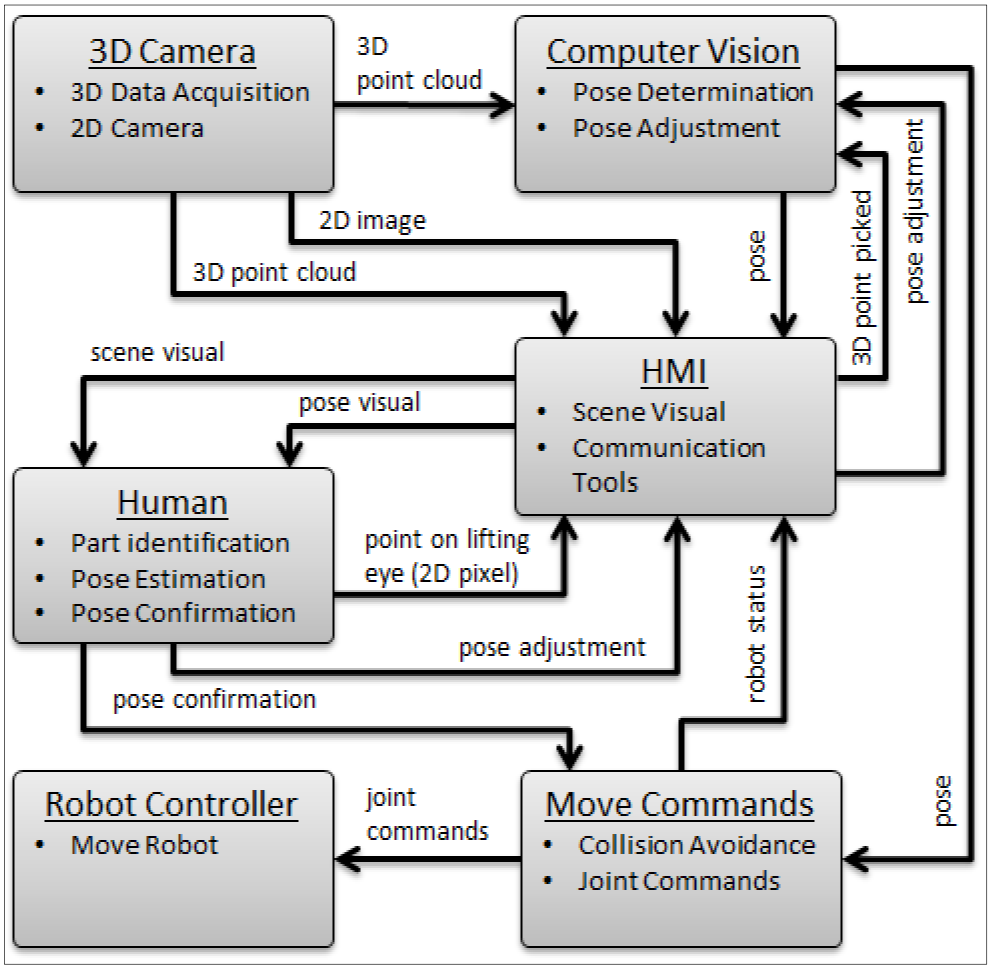

Figure 8: System diagram

The 3D camera subsystem was based on the openni_launch package from ROS. It allowed for an OpenNI-compliant device such as the Kinect to generate a 3D point cloud and 2D image by the same device [30]. This was important because the user needed the 2D image to select the location of the lifting eye. By using the same device, the individual pixels of the 2D image corresponded to a 3D point in space within the same point cloud that was being published by this node.

Initially, the 3D point cloud data was located with respect to the coordinate frame of the camera itself. In order to transform the point cloud data to be usable by the robot, a calibration was needed. We were able to use an intrinsic calibration method [30] to calibrate the intrinsic parameters of the individual Kinect hardware. We then used those intrinsic parameters in the industrial extrinsic calibration to calibrate the camera position to the robot workcell [31]. We constructed a fixture that mounted directly to the wrist of our Motoman MH5F industrial robot (Figure 9). The calibration routine was able to determine the location of the fixture from the grid of circles with respect to the camera. Then, using the robot geometry and joint states, the routine was able to calculate the location of the fixture with respect to the robot. Then it was able to calculate the transform from the robot base to the Kinect coordinate frame. With this transform known, we were able to apply this to the raw data of the Kinect output in order to provide a usable point cloud for the system.

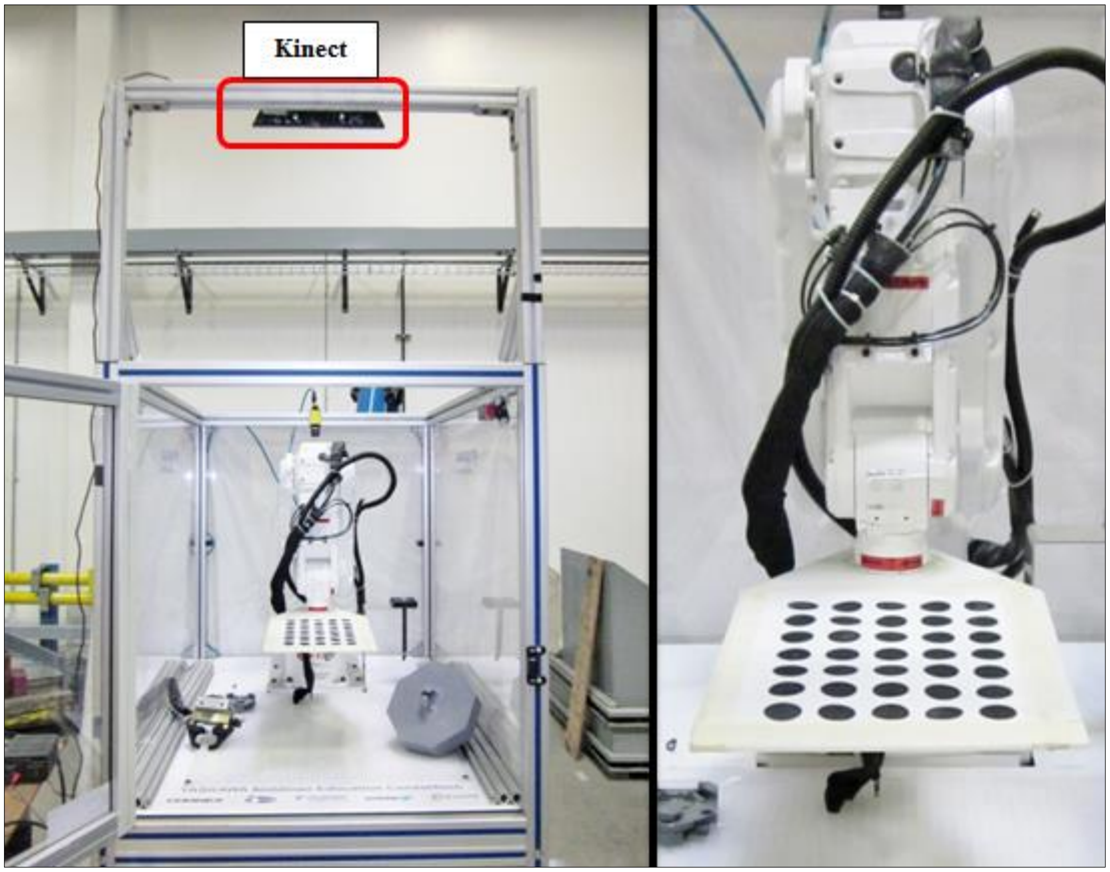

Figure 9: Camera calibration setup. Left – workcell configuration with Kinect located above. Right – calibration plate attached directly to robot.

The human machine interface (HMI) subsystem used RVIZ as its base. We customized the configuration of RVIZ to make the interface as intuitive as possible. We were able to overlay the 2D image from the Kinect onto the calibrated point cloud which looked no different than a normal 2D camera image as seen in Figure 10. We were able to use the publish point tool from the standard set of tools to give the user the ability to pick a point on the image. When the publish point tool was used in the HMI, it communicated the 3D picked point to the computer vision program. The computer vision program used the 3D point to generate an initial alignment for the location of the lifting eye by simply translating the source cloud to the picked location. After the computer vision program finishes its alignment routine, it communicated the 6 DOF pose to the HMI. The cycle time for this alignment is approximately 1 second. When the HMI received the 6 DOF pose it overlaid the lifting eye geometry over the 2D image. We configured the overlay to have enough transparency that the pose could be compared to the 2D image and determine if the pose is correct or not. This allowed the user to determine if the computer vision generated a successful or unsuccessful pose estimation. A successful pose is shown in Figure 10, and an unsuccessful pose outcome is shown in Figure 11. An arrow was also overlaid on the image that showed the initial guess of the direction of the front of the part. Because the lifting eye is symmetrical the user had the ability to reverse the orientation of the pose by 180° by selecting the arrow on the screen. The arrow was generated using the interactive markers ROS package [32]. The interactive marker arrow was a button that communicated with the computer vision program as seen in system diagram in Figure 8. Another set of interactive markers were used to create red and green buttons as seen in upper corners of Figure 10 and Figure 11. These buttons were used to allow the user to accept or reject the pose as seen on the screen. If the user rejected the pose estimation, they then reselected a point and start the process again.

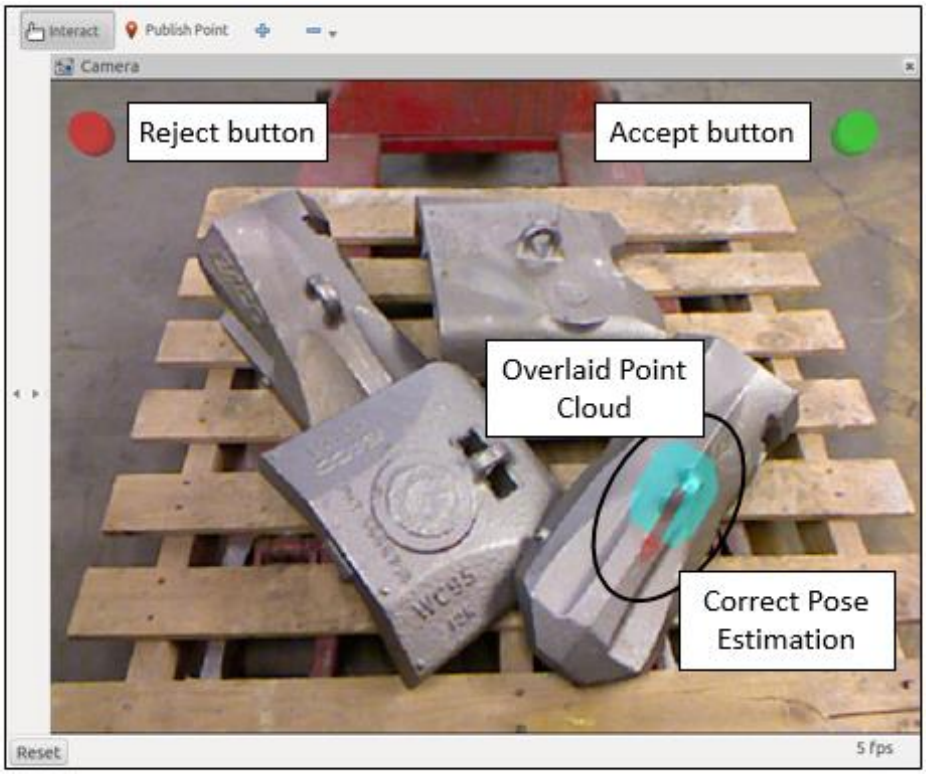

Figure 10: Example of the system user interface with a successful pose estimation (text boxes not shown in system interface)

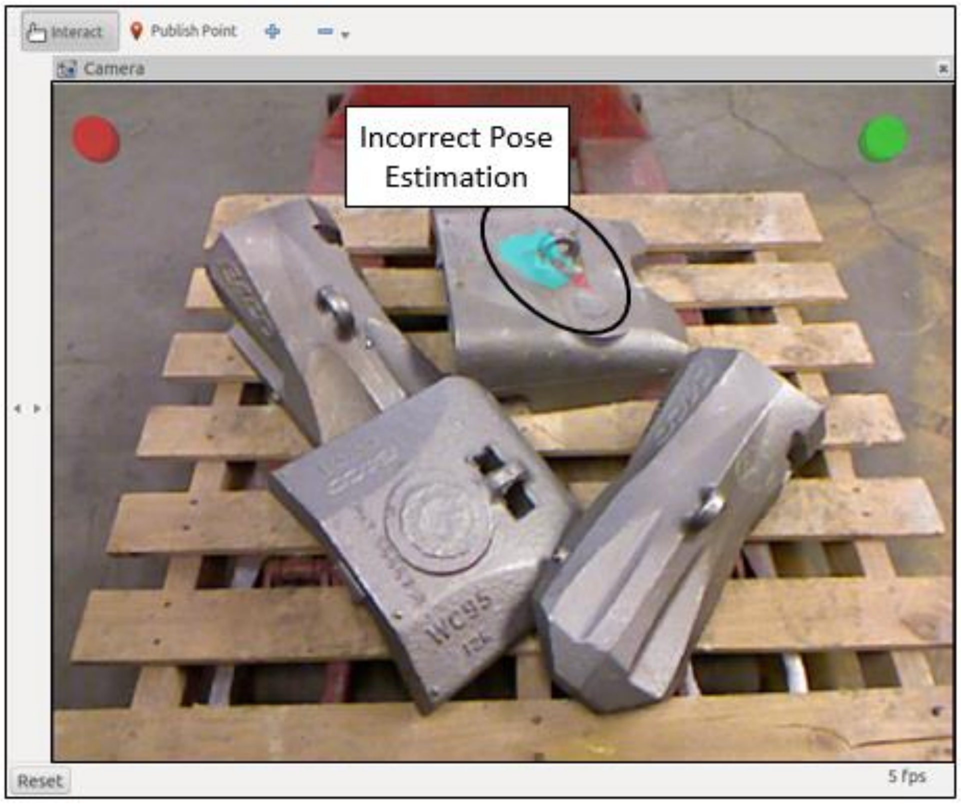

Figure 11: Example of the system user interface with an unsuccessful pose estimation (text box not shown in system interface)

The computer vision subsystem was simplified because of the shared autonomy approach. The node loaded the reference point cloud of the lifting eye. It received the calibrated point cloud from the camera node and waited for the user to pick the 3D point of the lifting eye. When the node received the picked point by the user, it translated the reference cloud using a transform value equal to the picked point values. It then ran the iterative closest point (ICP) algorithm from PCL. ICP is an algorithm that systematically translates one point cloud to match another. For each point in the source cloud, it searches within a distance radius to identify the closest point in the target cloud. Then it applies an incremental translation to the source cloud to reduce summation of these distances [33]. We used the routine in a forced loop with an iteratively decreasing search radius. When the ICP loop finished, the node sent the final pose of the lifting eye to the HMI. If the computer vision node received a pose adjustment from the HMI, the pose is transformed 180° along the z-axis with respect to the pose of the lifting eye and resent to the HMI.

The move commands subsystem was based on the ROS Industrial package [34] which utilized the MoveIt platform of ROS. We were able to use MoveIt to create a Universal Robot Definition File (URDF) for the robot workcell. The URDF includes the robot visual geometry, collision geometry, workcell visual geometry, workcell collision geometry, joint limits and other information to build an accurate workcell model (Figure 12). Our URDF model gave ROS the ability to create an Inverse Kinematic solution to avoid collisions with both the robot itself as well as the workcell around the robot. Collision avoidance using point cloud data was available as a standard tool within MoveIt, but it was not a part of this scope.

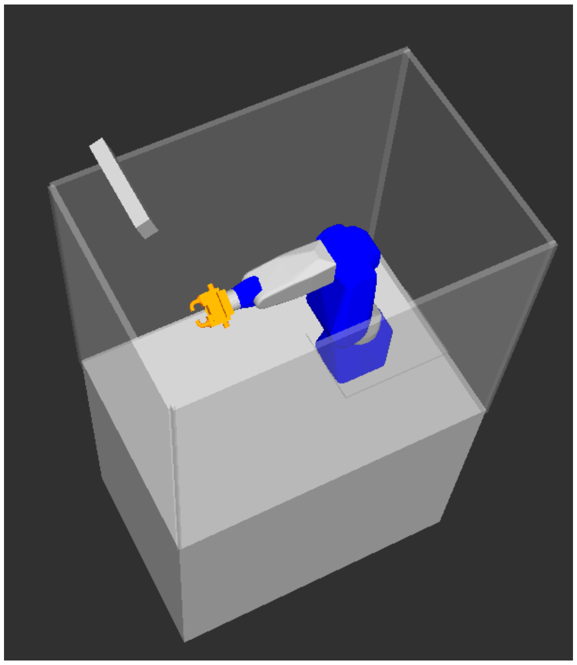

Figure 12: Workcell model in MoveIt

The move command program used the URDF, configuration files [35], pose estimation and the confirmation notification from the HMI to generate three different poses. An approach pose was calculated based on the pose message received. The pose was calculated by transposing the final pose 100mm in the direction that it normal to the surface of the lifting eye. This approach pose allowed for robot to approach the lifting eye with as much gripper clearance as possible as seen in Figure 13.

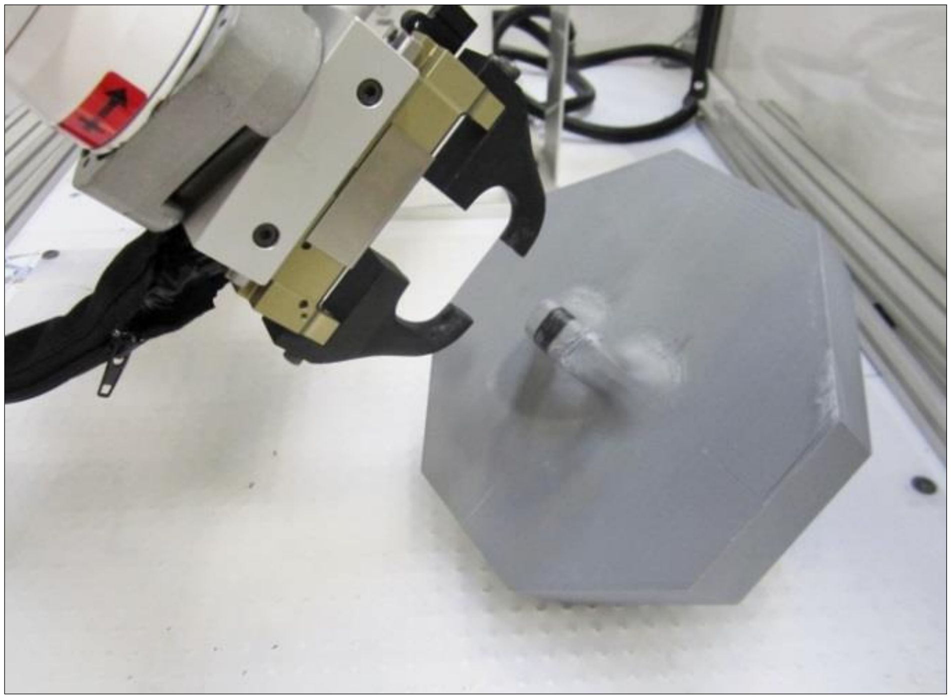

Figure 13: A good approach position ready for final approach

The robot then moved to the pose estimated by the computer vision subsystem and paused for the gripper to close (Figure 14).

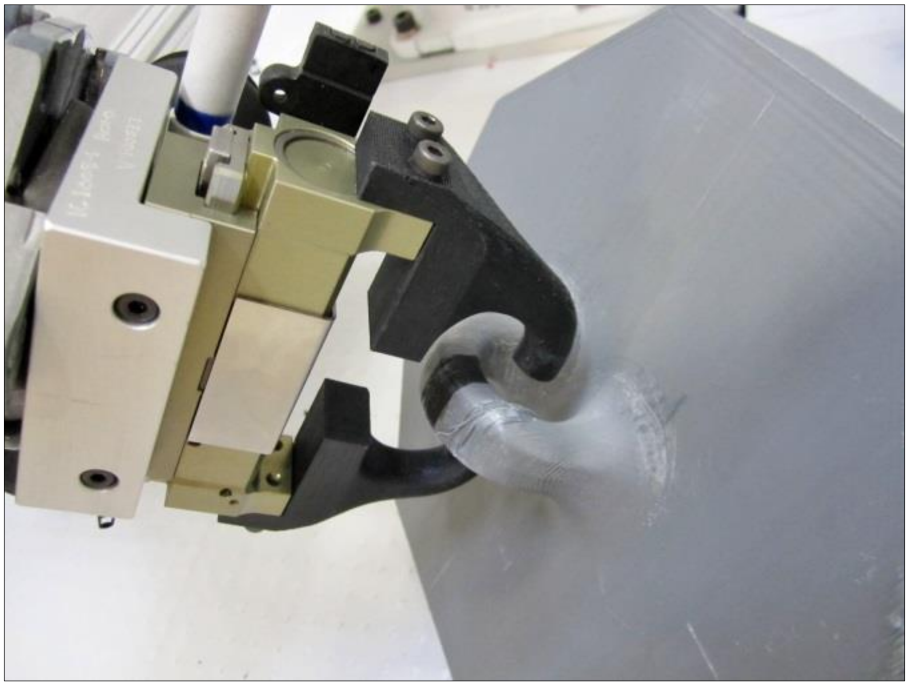

Figure 14: A good final position ready to close the gripper jaws

Gripper I/O control was not a part of this scope because I/O control was not available for this robot controller at the time of our research. After the gripper was closed, the program then moved the robot up vertically 100mm and then to a preprogrammed “work holding” position as described in the concept of the heavy-helper.

3.4 User Study

The performance of the system was reliable in a controlled environment. However, there are many factors in a foundry environment that impact the performance of such systems:

- Ability of a worker to identify castings on a screen from a single camera view
- Ability to accurately use a computer input device (mouse, joystick, etc.)
- Effect of wearing personal protective equipment (PPE)
- Speed of worker to use input device
- Environmental issues such as dirt and dust

The pose estimation program relied on the user to estimate the location of the lifting eye and the accuracy required for this operation was unknown. Without good user data to test with, we could only estimate. We also did not know how often the program would produce a good pose estimate and how often the user would have to retry. The shared autonomy approach was relying on the user’s ability to identify the casting on a screen. User data was needed to compare the user performance to a previously tested VFH recognition program. After defining the purpose and desired outcome of the user study a protocol was formed. The user study was titled “HMI Part Identification and the Effect of PPE” and was approved by the Oregon State University Intuitional Review Board (Study 6276).

For the user study, we selected four castings to be used throughout the study. There were two castings from two different product families. This was done to add difficulty to the part identification task of the user study. As seen in Figure 15, there was only a small shape difference between the two points on the right side of the pallet. The user was given instructions on how to differentiate the castings in the tutorial video which was given at the start of the user study. The four castings were positioned on a pallet together similar to how they are now in the foundry. As seen in Figure 15, there were some occlusions in the scene, but all lifting eyes were visible for all parts in all scenes recorded by the Kinect.

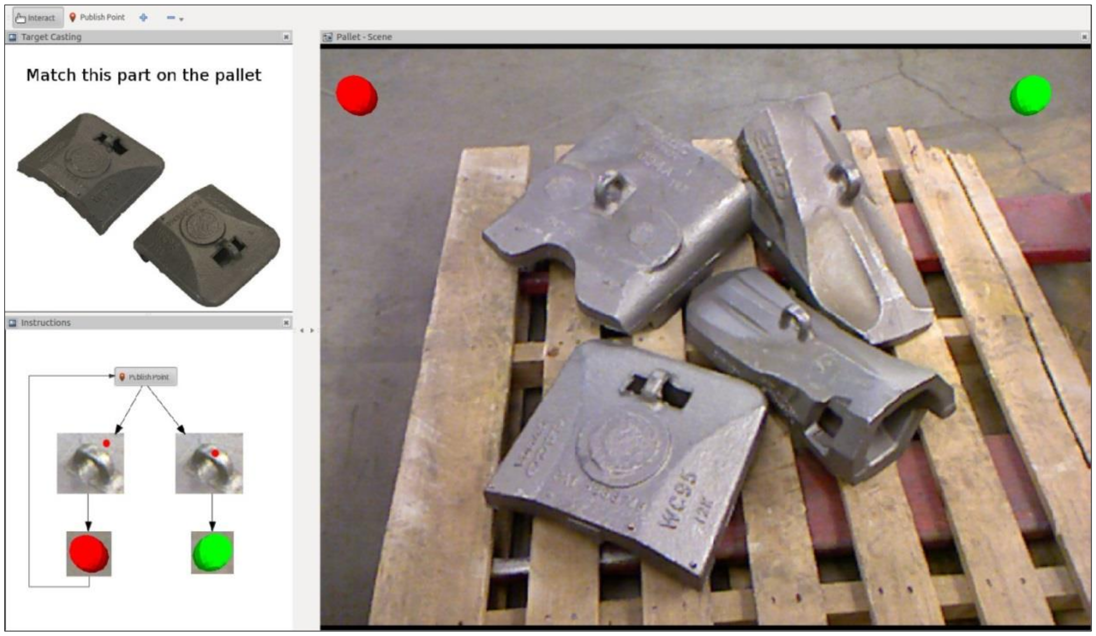

Figure 15: User study interface showing the scene on the right and the target part in the upper left

The user study interface, as shown in Figure 15, was designed to be as close to the real interface as possible. The only difference was the instructions shown on the left side of the screen. A target casting was communicated to the user by displaying two views of the part in the upper left of the interface. The task of the user was to identify the target casting on the pallet and click a point on the lifting eye.

When the user clicked a point on the screen, a red dot was shown to give the user feedback of which point was picked. If the user was satisfied with the point that was picked they clicked the green ball, but if they were not satisfied the red ball was used to reset the picked point. There were instructions on the lower left of the interface to help the user through the steps if necessary. After the green ball was selected, the target casting in the upper left and scene of parts on the right were changed to a random selection. The task was performed 8 times for each input device and the device order was randomized.

RVIZ was used for the main program interface. It used the same controls as the main system. The user used the publish point tool to select a point on the lifting eye similar to the designed system. However, they were not be seeing a live feed from a Kinect. We utilized the rosbag package to simulate a live feed from the Kinect [36]. Rosbag allowed us to record the data generated by the Kinect and replay it to simulate a live feed. The rest of the sub-system were not affected by the difference. Similar to the main system, the user study interface displayed a 2D image on top of the 3D point cloud. A master program for the user study was written to control the different scenes of Kinect data that were replayed. These scenes were randomized from a list of 20 different scenes. This node also randomized which part it asked the user to identify and select on the scene. This image, as seen in Figure 16 and in the upper left of Figure 15, did not have part numbers or names. The image showed the part in two isometric views to help the viewer identify the casting in the scene. The master program also recorded all information for each task: user number, target part number, HMI number, scene number, picked point (x,y,z), duration of task, and reset instances.

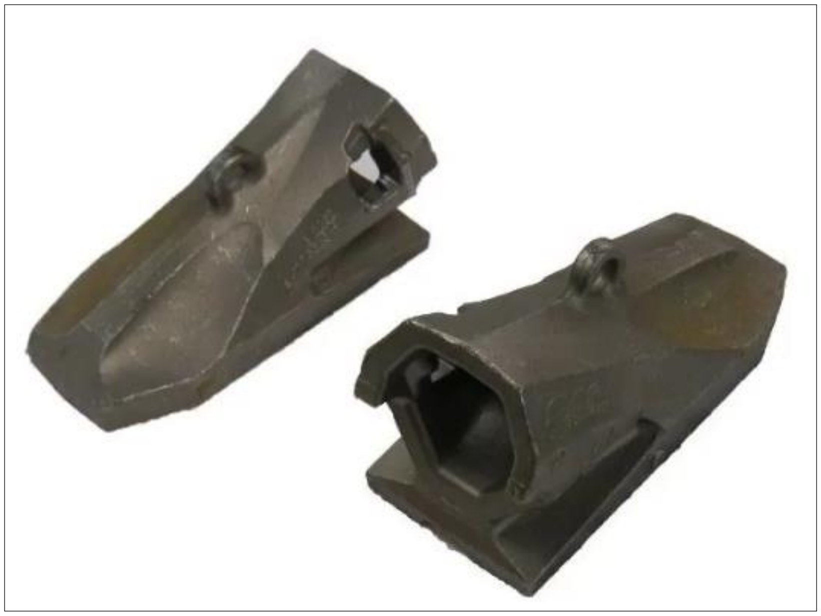

Figure 16: Example image of the two isometric views shown to the user to establish the target part

We were intentionally vague with our instructions to the user on how close to the top center of the lifting eye was considered “good.” We did not want to skew the results by giving them a tolerance to shoot for. The tutorial video showed that if a user selected a point that was not on the lifting eye, the system to fail. However, an accuracy target was not given. We also, decided not to give feedback to the user (other than the red dot) on exactly how close the picked points were to the top center of the lifting eye to reduce the effect of learning during the study duration.

The user study was conducted at the foundry with factory workers who were potential system operators. It was on a volunteer basis and the user study was conducted during their work hours while they were “on the clock.” They were not compensated for their time or effort in addition to their normal pay and all users followed the same protocol.

The user was brought into an office environment where they were given a consent form and time to review and sign. They were given a short pre-questionnaire that gave us context of the user. A copy of this is shown in the appendix. All information was anonymous and users were given a number with no ties to names or contact information. Then they watched a 3 minute video tutorial that gave them context of the user study and instructions on the task they will be performing on the computer. To establish a baseline, they performed the task of selecting the lifting eye using the mouse with no PPE. This first session lasted approximately 10 minutes and they then returned to their jobs. After approximately 3 hours the users were then pulled back into another session. This time the computer and monitor were located outside in the foundry environment. The users kept their PPE on for this session. The minimum PPE worn was leather gloves, hard hat and face shield. Some users also had safety glasses under the face shield. Some users wore thin gloves under their large leather welding gloves. We decided to let them wear their own PPE equipment to give the results a more realistic effect rather than standardize on a single PPE setup that they were not used to. During this second session, the users performed the same task as before but with different input devices. The devices were roller-ball mouse, joystick, touchscreen, and mouse. The device order was randomized for each user. The user performed the task 9 times for each input device. The initial task was not recorded because the time measured to perform the task would not be accurate. The timer was started as soon as we started the program and this would be different for each user. After the all tasks were completed with each input device the user was given a post-questionnaire to compete. A copy of this questionnaire is also in the appendix. After this session, which lasted approximately 10 minutes, the user returned to their job.

3.5 User Study Results

There were 20 people over three shifts that volunteered to participate. There were 7 participants from the first shift (6am – 2pm), 7 participants from the second shift (2pm-10pm) and 6 participants from the third shift (10pm – 6am). Each task during the sessions created an individual text file on the computer. The text file was named for both user number and input device number. Inside the text file, both data about the scene (part number, scene number) and user task performance (input device, picked point, task duration) were recorded.

The post processing of the user study data started with creating a ground truth pose for each of the 4 parts in the 20 different scenes. This was done using an industry leading point cloud processing software, PolyWorks. The point cloud from the Kinect was imported into PolyWorks along with the CAD model for the casting. The CAD model was aligned to the point cloud using PolyWorks' proprietary Best-Fit alignment tool. The position (xyz) and orientation (quaternion) of each lifting eye was determined and recorded to create a key for each scene. This would be used as the ground truth for each pose generated by the system.

A program was written to generate the system results from the user study data. The program stepped through each text file for each task created by the user. It loaded the data into the system parameters. The appropriate rosbag file for the scene was played to generate the point cloud. The point picked by the user was published as if it was picked from RVIZ in the real system. Then the computer vision node used the picked point to translate the reference cloud to the picked point of the point cloud. It then ran ICP as described in the system design. The output pose was recorded. The ground truth pose was then loaded from the key file and the delta transform from the ground truth to the estimated pose was recorded. Because the user was not in the loop during this process to determine if the orientation needed to be flipped or not (symmetric geometry), the estimated pose was flipped and compared to the ground truth. The one that was closer to the ground truth was saved to the report.

Previously we had made the assumption that the actual geometry of the lifting eyes were close enough to use a generic lifting eye geometry as a reference. However, each casting lifting eye geometry is slightly different. To test if this influences the results of the pose estimation results, the ICP routine was completed 5 times or each data point. It was run using the generic lifting eye geometry as well as the actual geometry of each lifting eye that was created using a Stienbechlier white light scanner that is accurate to within 0.02mm. After each ICP routine, the fitness score of the ICP routine as also recorded. This was done to determine if the fitness score would correlate with either the part matching the reference geometry or delta transform (pose accuracy).

The part identified by the user was determined by calculating the shortest distance between lifting eye locations in the key file and the point picked by the user. This was then compared to the target part to determine if the user identified the casting correctly or incorrectly. As seen in Figure 17, of the 800 parts identified, the user identified the correct casting in the scene 98% of the time.

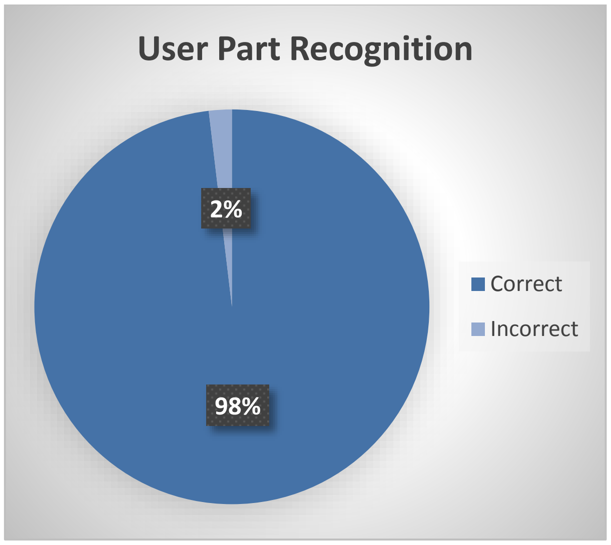

Figure 17: User part recognition results from 20 users totaling 800 part identifications

Table 3 breaks the identification down further by each casting. It shows that users had a slight issue with differentiating the difference between the point castings.

Table 3: User part recognition results breakdown. Grey cells represent a correct match (higher is better).

Human		Part Recognition Estimation			
		Point 1	Point 2	Wear Cap 1	Wear Cap 2
Actual part	Point 1	99.0%	1.0%	0.0%	0.0%
	Point 2	5.8%	94.2%	0.0%	0.0%
	Wear Cap 1	0.0%	0.0%	100.0%	0.0%
	Wear Cap 2	0.0%	0.0%	0.5%	99.5%

As a comparison, we ran each scene through a similar part recognition program as we tested in Phase 1. We modeled our program after the VFH recognition and pose estimation solution from PCL [14]. We trained our VFH library using 30 snapshots from each casting. The snapshots were taken in a uniform distribution around the top hemisphere of the casting, which was the only orientation of the castings on the pallets. The snapshots were created using the Kinect, as seen in Figure 18, at a similar distance compared to the scenes of parts on the pallet.

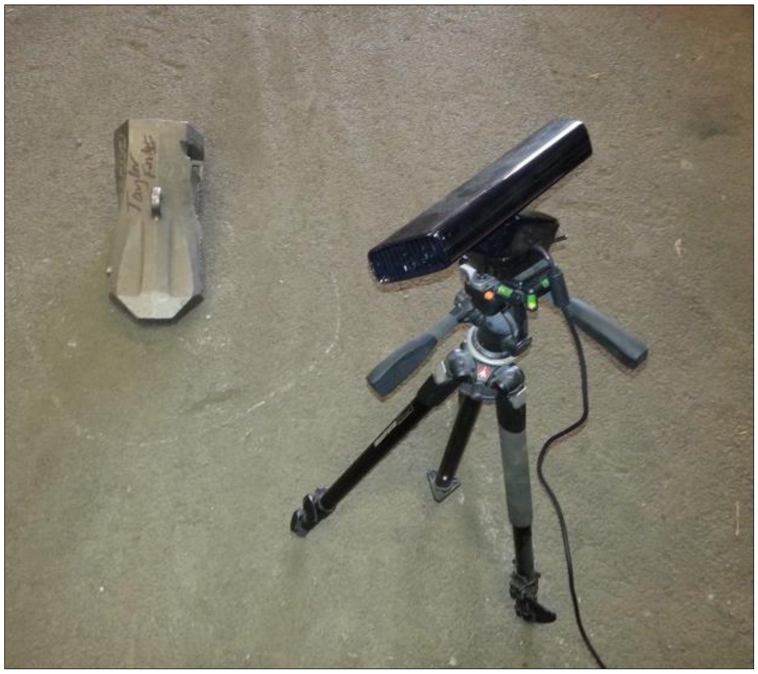
Figure 18: Data acquisition using a Kinect sensor for part recognition testing

The snapshots were taken on a flat concrete floor which made cluster extraction simple using the RANSAC plane segmentation [17]. The VFH classifier relied on parts in the scene to be able to be extracted into an individual point cloud cluster. The VFH was then created for each part cluster view and the library was created. We found when processing the scene of the parts on the pallet that the cluster extraction method is not a trivial task. Initial cluster extraction methods were failing because the RANSAC plane extraction method was not able to identify the pallet as a plane on a consistent basis. This might have been attributed to the irregularity of the boards on an uneven plane or if it identified the concrete floor as the primary plane. The cluster extraction was also failing as a result of the plane segmentation issues. Because we cloud not easily remove the data on the pallet, we were forced to decrease the maximum distance between clusters. This was causing the casting to be split into two point cloud clusters. To eliminate this cluster extraction issue as a factor in the benchmarking result, we used the known lifting eye location and a passthrough filter to segment the data prior to the cluster extraction method. In an autonomous cluster extraction and part identification, this work-around would not be available. However, after the work-around, we were able to get a good cluster for each casting in the scene. Each VFH for each cluster was processed and compared to the library and the top 5 VFH matches were recorded. Two identification methods were used. In one method, the part was identified using the top rated match from the library lookup. In the second method, the most occurring match in the top 5 was determined to be the most likely match. As seen between Figure 19 and Figure 20 the second method of using average of the top 5 matches performs slightly (4%) better than just using the top rated match.

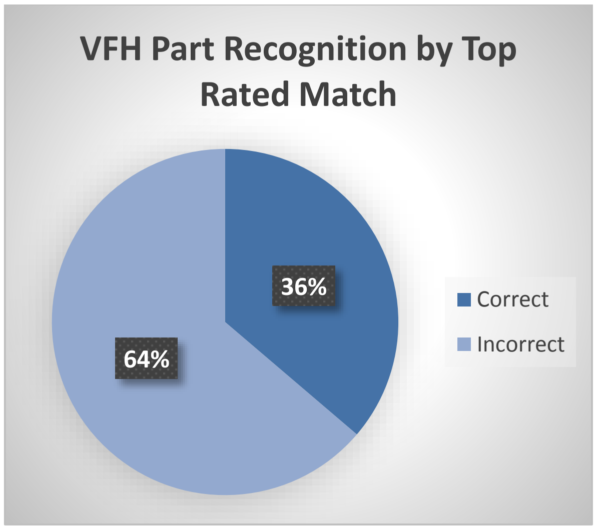

Figure 19: VFH part recognition testing results using the top rated match

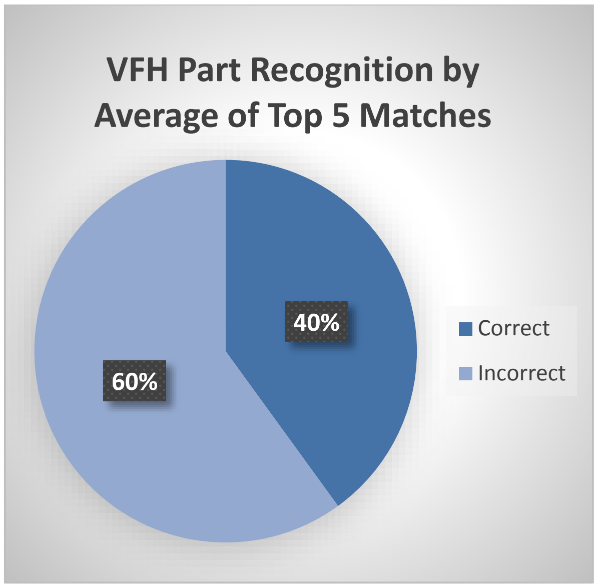

Figure 20: VFH part recognition testing results using the top 5 results

To summarize the part recognition results, the three methods were combined in Figure 21. The part recognition success rate of 98% by the users in the user study show a significant performance difference compared to the automated part recognition that we tested at only 40%.

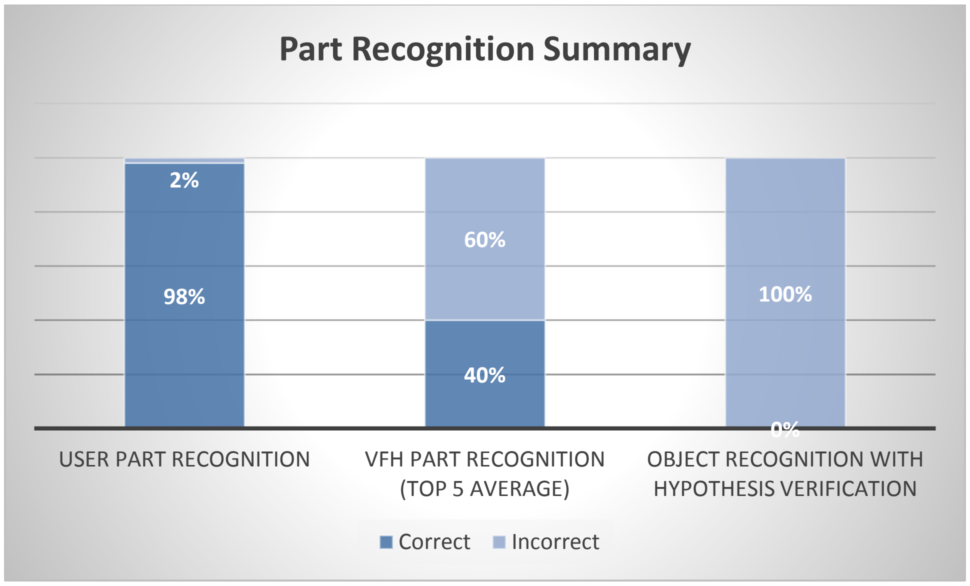

Figure 21: Part recognition summary results

A further breakdown of the VFH matching can be seen in Table 4 and Table 5. The actual target casting is represented in the rows of the table. The values in the cell represent the number of times the part was recognized as the parts labeled in each column.

Table 4: VFH part recognition testing using top match results breakdown. Grey cells represent a correct match (higher is better).

VFH Top Match		Part Recognition Estimation			
		Point 1	Point 2	Wear Cap 1	Wear Cap 2
Actual part	Point 1	60.0%	35.0%	5.0%	0.0%
	Point 2	75.0%	25.0%	0.0%	0.0%
	Wear Cap 1	40.0%	15.0%	45.0%	0.0%
	Wear Cap 2	60.0%	10.0%	15.0%	15.0%

Table 5: VFH part recognition using top 5 results breakdown. Grey cells represent a correct match (higher is better).

VFH Top 5		Part Recognition Estimation			
		Point 1	Point 2	Wear Cap 1	Wear Cap 2
Actual part	Point 1	55.0%	35.0%	10.0%	0.0%
	Point 2	50.0%	50.0%	0.0%	0.0%
	Wear Cap 1	45.0%	10.0%	45.0%	0.0%
	Wear Cap 2	25.0%	30.0%	35.0%	10.0%

We performed a second attempt at an automated part identification method using PCL’s more recent recognition pipeline, global hypothesis verification. It works by first identifying keypoints in the point cloud. It then estimates features such as point cloud normals. Using these keypoints and features, it generated a list of possible matches, then filtered through these matches to identify groupings of similar geometric clusters. This is called correspondence grouping. From there it used ICP to identify the best pose from its initial match. Then through a series of filters and cost functions, the system identified which hypothesis pose is correct [37].

We found that there are a significant number of configuration settings for this process. Each step in the program took some fine tuning for the unique scenario.

Many factors such as point density, part size, number of clusters in the scene and others played a role in determining the right balance of settings to produce a good hypothesis in a reasonable amount of time. The more restrictive the configuration settings, the less likely to get a result. The less restrictive the settings are, the more computing is necessary to identify a correct hypothesis from a much larger list. We tuned our settings to produce between 8-20 potential poses which took approximately 2-3 minutes per scene. An example of the different potential poses generated can be seen in Figure 22. Using these settings we did not see a single confirmed hypothesis through all 20 scenes for any of the parts.

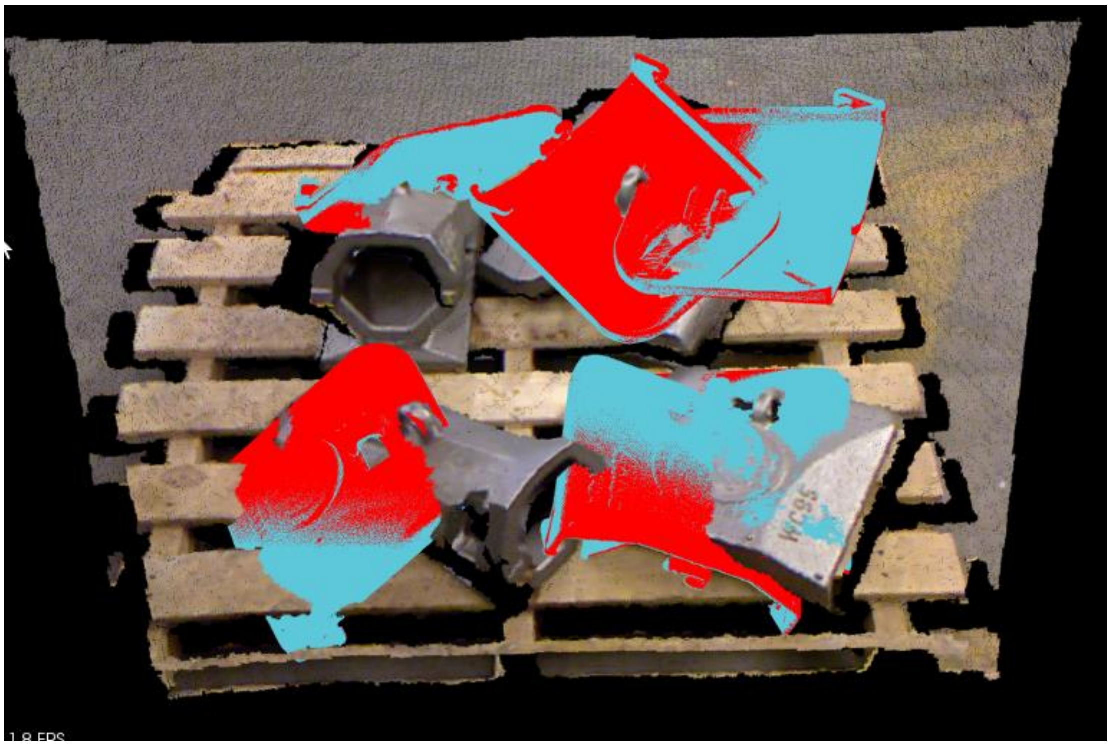

Figure 22: An example of a failed recognition using the Hypothesis Verification for 3D Object Recognition

The user study master program recorded the duration of each task. The data was sorted by input device and displayed in Figure 23 as box plots with the inner quartiles represented by the boxes and the range dipected by the wiskers.

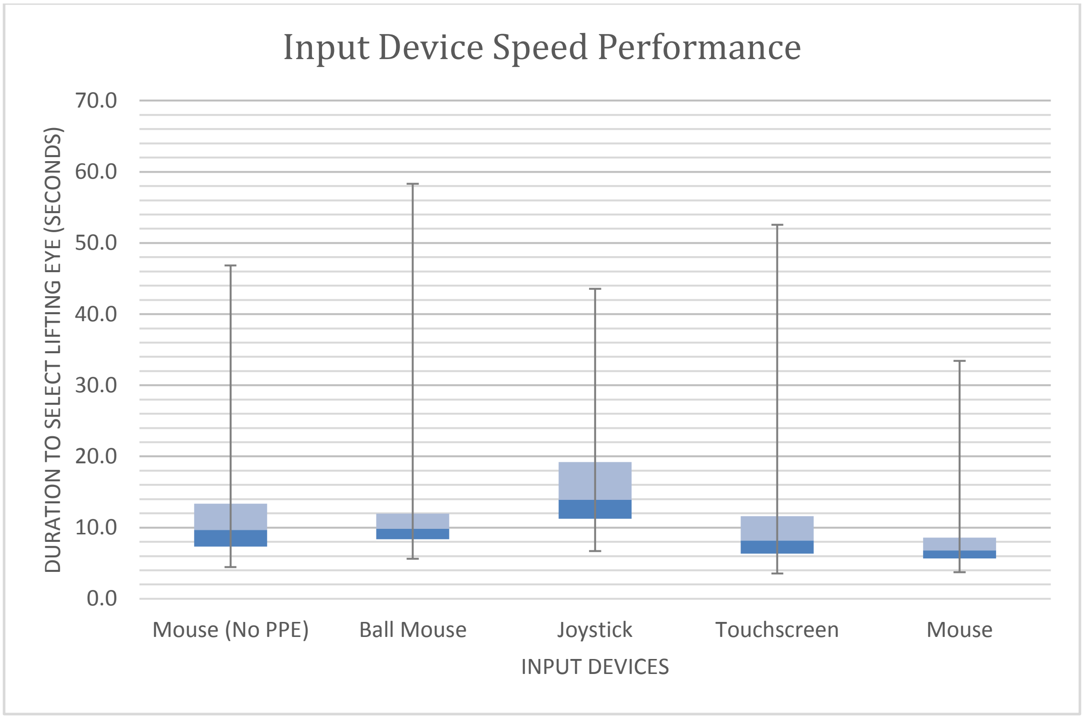

Figure 23: Input device speed performance results

Each picked point by the user was compared to the ground truth and a deviation was calculated. These distances were then sorted by input device. The data is displayed in Figure 24 in the same box plot form as Figure 23.

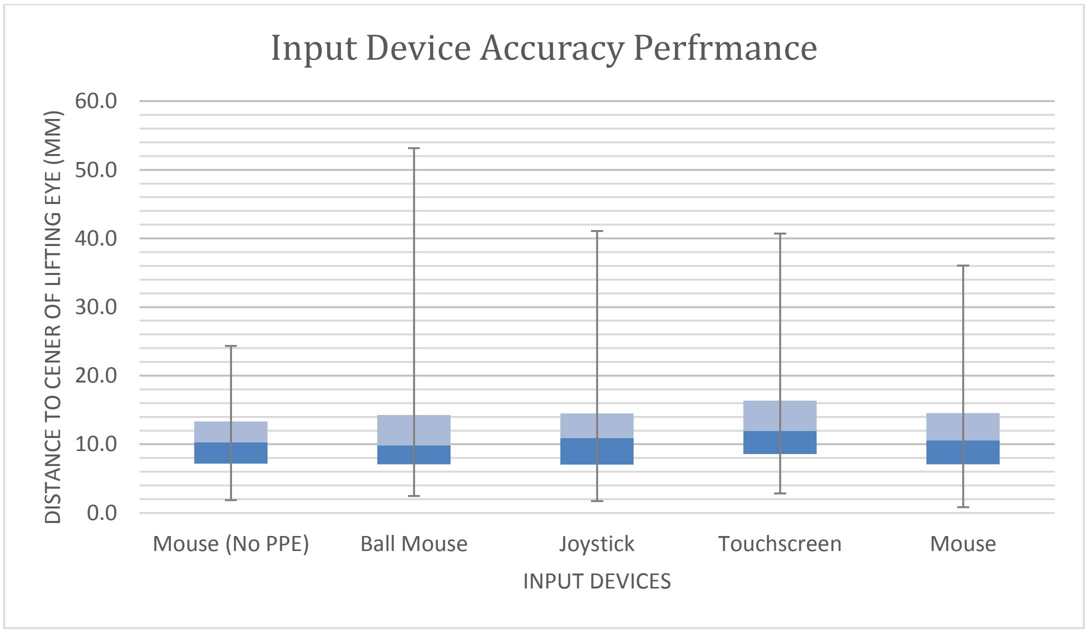

Figure 24: Input device accuracy performance results

As previously discussed, the ICP algorithm was run on each part 5 different times with different reference geometry. Each time the ICP algorithm was run, it was compared to the ground truth pose. The threshold for a successful pose was estimated given the geometry of the lifting eye. We estimated that a good grasp would be achieved if the gripper was within 20mm and within 15 degrees of the actual pose (in all directions) of the lifting eye. Given these estimations, we were able to compare the output of ICP to the ground truth and classify the result as success or failure. These success rates can be seen in Figure 25. The success rate using the generic lifting eye reference geometry showed a 51.5% success rate while a success rate of 86% was achieved when the actual part geometry was used to identify the lifting eye using ICP. A third option was shows on the right by using a single geometry per part family. For both point castings, a single reference geometry was used and for both wear caps, a single wear cap geometry was used. This produced a success rate of 78.9%.

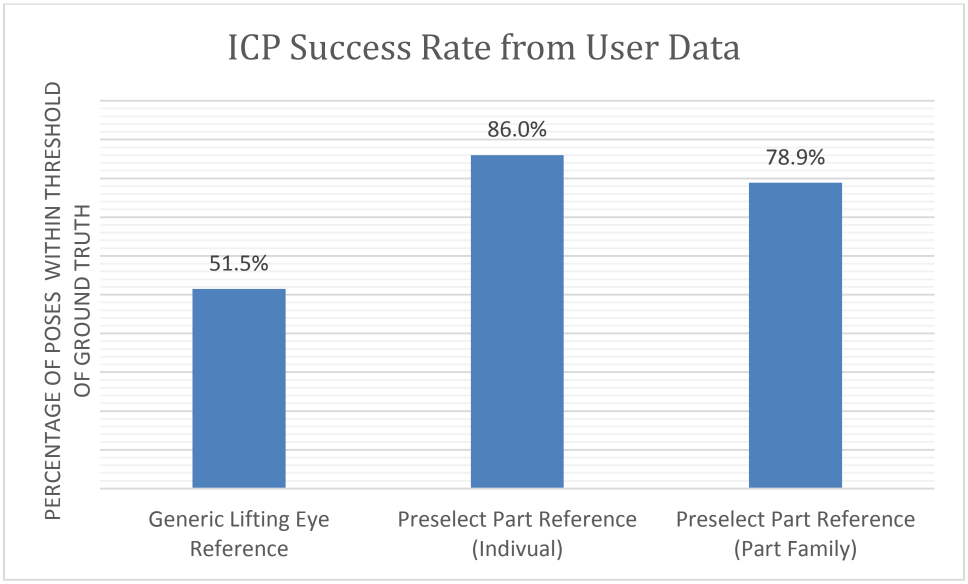

Figure 25: Pose estimation results from ICP using user data

The ICP algorithm produces a fitness score when it converges to an outcome. This score was recorded for all ICP runs with the different reference geometry. We then use these scores to estimate the actual part as a pseudo part recognition test. We found that the ICP score method was 57% successful in identifying the proper part as shown in Figure 26.

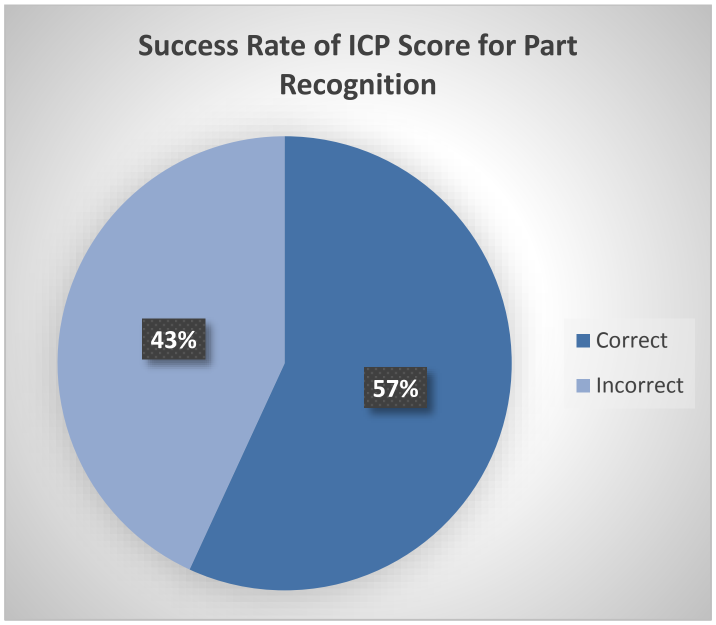

Figure 26: Success rate of ICP score for part recognition

Each ICP routine produced a fitness score with the pose estimation. Lower values represent a better fit. A score of zero would represent a perfect fit between the two point clouds. The fitness scores were sorted into successful and unsuccessful categories for the ICP pose outcome compared to the ground truth. These two categories represent the ICP using the matching part's geometry. As shown in Figure 27, there is a strong correlation between high fitness scores and unsuccessful final poses. There is no overlap between the inner quartiles, which makes classifying an estimated pose's success based on ICP score a viable option.

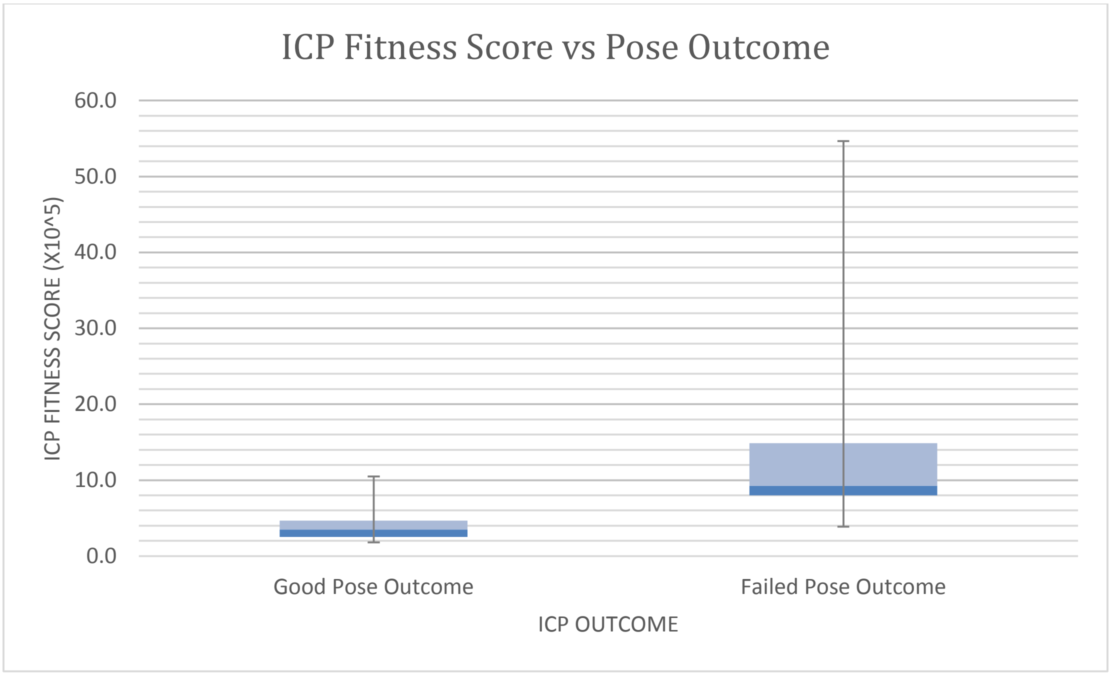

Figure 27: ICP fitness scores for successful or unsuccessful pose estimations (smaller is better)

The users were consistent in their post-task questionnaire. All users agreed or strongly agreed that the system was intuitive. The joystick and mouse received the lowest and highest score respectfully for the different input devices. The average user agreed that the system would increase both their safety and efficiency.

3.6 User Study Results Discussion

There is a significant difference between the computer vision part recognition and the user's ability to recognize a part from an image. With a 98% success rate, it is justifiable to base a system off of the assumption that a user will be able to recognize the correct part from a 2D image. With additional experience or training, it is not out of the question to assume an average better than 99% for a future system. We are satisfied that we made the decision to use a shared autonomy approach rather than rely on a system that gave only a 40% success rate.

The performance of ICP with the generic lifting eye geometry was lower than expected. The performance during initial trials was considerably higher than the 51.5% outcome from user data. This shows the reason why it is important to test the system with data from real users in a typical system environment rather than in an office by someone who wrote the program. The 86% success rate of the ICP algorithm with matching part geometry was significantly higher than using the generic lifting eye geometry. While this complicates the interface and requires one more step for the operator, preselecting the part from a list prior to selecting the lifting eye would not impact the pitch time of the system considerably. The time that it would take to preselect a part prior to picking the lifting eye would be less than the time it would take to adjust a bad pose, or the time it would take to troubleshoot a bad grasp.

There was not a considerable effect on performance by users wearing their PPE. With regard to speed, the average task duration was better with PPE than without, save the joystick (Figure 23). This can be attributed to learning during the user study. In hindsight, efforts should have been taken to better offset the effect of learning. However, the accuracy of points picked was worse with all input devices with PPE compared to the mouse without PPE. Because the performance differences between devices were not enough to effect the systems output, the appropriate device for the final system was determined by the user feedback from the questionnaire. The final system was designed to use a touchscreen with a wireless mouse option. According to the questionnaire this provides an intuitive interface according to 100% of the users we tested.

One more data point that was not measured was the morale of the workers when we first brought up the idea of using industrial robots in the foundry. It was obvious that many were concerned with losing their job to a robot. While we reassured them that the robot system that we were designing was going to use a robot as an assistive robot and not one to replace their jobs, many were not convinced. We think the majority of people who did not volunteer did so because they felt like they would be helping to replace their own jobs.

3.7 Physical Testing

Physical testing was completed using a Motoman MH5F robot in the Motoman Education Cell. The workcell used the Motoman FS100 controller, which has a small form factor compared to the more standard DX100 controller. The workcell was also powered from a standard 110V outlet, which made testing much easier to setup. The robot had an integrated Shunk parallel jaw gripper as an end effector with customized 3D printed jaws. The MH5F had a 5kg payload, which meant testing with steel castings was not possible. A spinning top toy was the inspiration for the test part we designed as seen in Figure 28. We designed it so it would be stable in multiple orientations but always give the robot access to the lifting eye. We made the bottom rounded to help randomize the location and orientation of the lifting eye for each test. The lifting eye scale was identical to the lifting eyes on the castings from the user study. A Stratysis Dimension 1200es 3D printer was used to create the test part. By making the part out of plastic we were able to keep the weight down under the payload limit. However, because the part was such a uniform color, it was hard to identify the lifting eye on camera, which was not the case with real castings. A black stripe was marked on the lifting eye of the plastic test part to help identify it on camera.

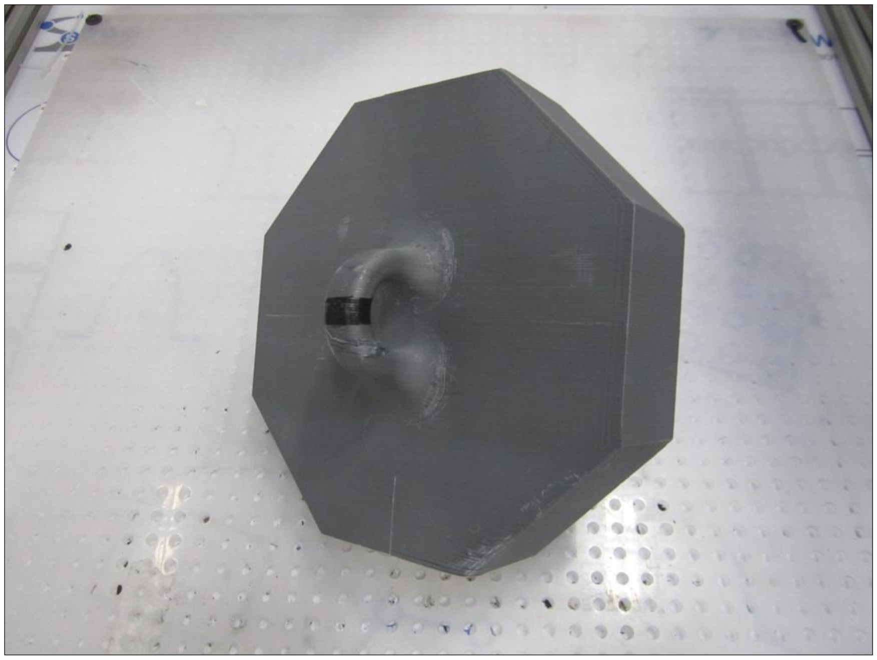

Figure 28: 3D printed lifting eye sample part for physical testing

The gripper did not provide a rigid connection to the casting, but the design of such a gripper was not a part of this project's scope. The MotoPlus ROS application was installed on the FS100 controller. This was the server application that provided the link between the controller and ROS. Communication between the controller and the computer was over an Ethernet cable with dedicated IP assignments. All software was configured as previously described in the System Design.

The Kinect was calibrated to the workcell world coordinate frame prior to testing using the industrial extrinsic calibration as previously described. The test part was randomly in the middle 60% of the table in front of the robot to give the robot a reasonable chance of not crashing against the side walls or with itself. The ROS system was started and RVIZ showed a 2D image looking down on the workcell. We then selected a point on the screen on the lifting eye. The overlaid lifting eye geometry was shown on the image. The arrow gave us a chance to adjust the orientation of the pose. The ICP algorithm used the generic lifting eye geometry, which was the same geometry the test part was designed with. If we were satisfied with the result, we selected the green button on the screen. Then the move commands subsystem used the final pose to create the initial approach pose which was sent to the controller first. This gave the gripper the best chance of making a good grasp when it made its final path into the final pose. Feedback from the controller allowed for the move commands node to send sequential moves between the different poses. We programmed delays into the routine to allow us to confirm the alignments of each pose. Also, because the I/O functionality was unavailable the current MotoPlus application, we manually opened and closed the Shunk gripper via soft buttons on the teach pendant. After the jaws were closed, the part was lifted straight upwards to a hanging position. Then the part was brought to the center of the table, turned to a random position along the vertical axis and dropped from approximately 1 inch above the table. This did an adequate job of randomizing the location of the next trial. Then the robot was moved to position in the back corner that provided good visual clearance to the workspace.

3.8 Physical Testing Results

The routine described previously was competed 40 times. As seen in Figure 29, the ROS-Industrial kinematic solver failed to generate a collision free solution for one or more of the poses 50% of the time. Of the trials where the kinematic solver generated a solution, a successful grasp occurred 80% of the time.

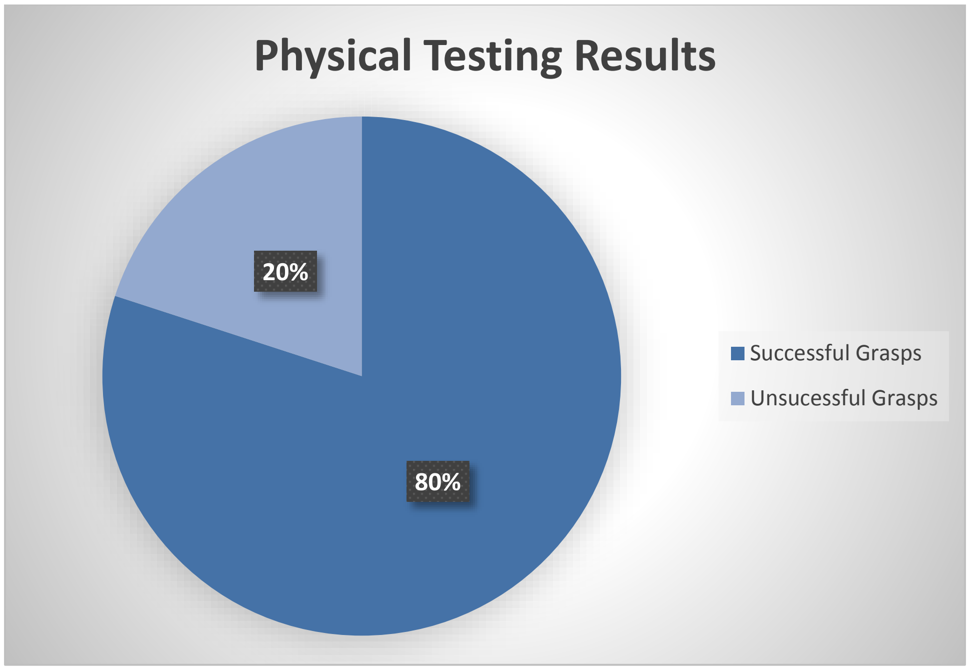

Figure 29: Physical testing results of system using the Motoman MH5F robot of solved IK solutions

A grasp was classified a success if the gripper closed completely through the lifting eye and was able to lift the part as show in Figure 30.

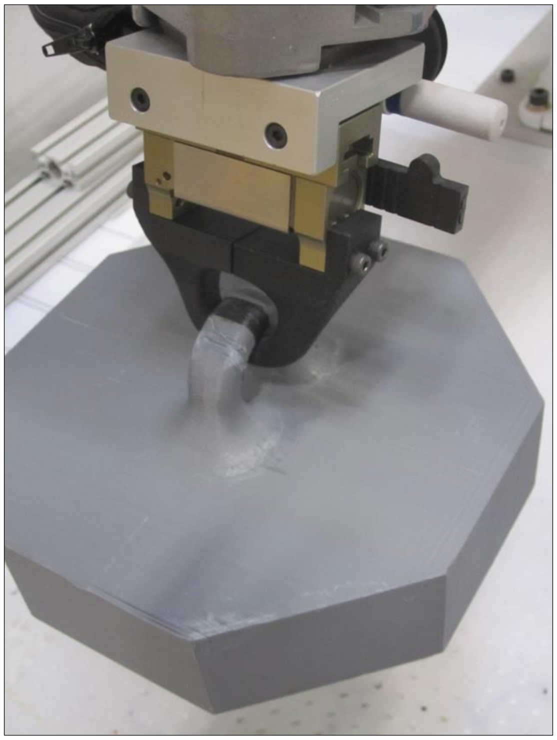

Figure 30: A successful grasp during physical testing

An unsuccessful grasps occurred if the open gripper jaws collide with the lifting eye on approach to the final pose as shown in Figure 31. This typically resulted in the plastic part being pushed and the gripper failing to close through the lifting eye as seen in Figure 32.

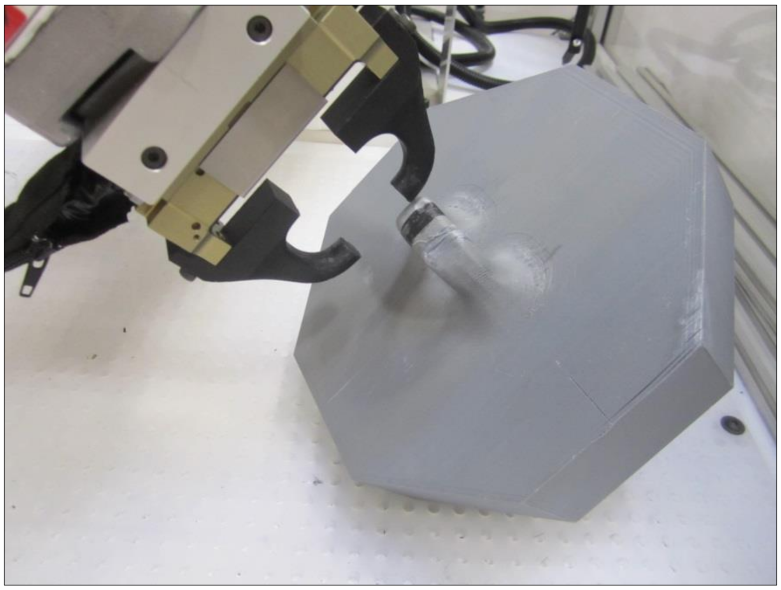

Figure 31: A failed grasp caused by collision

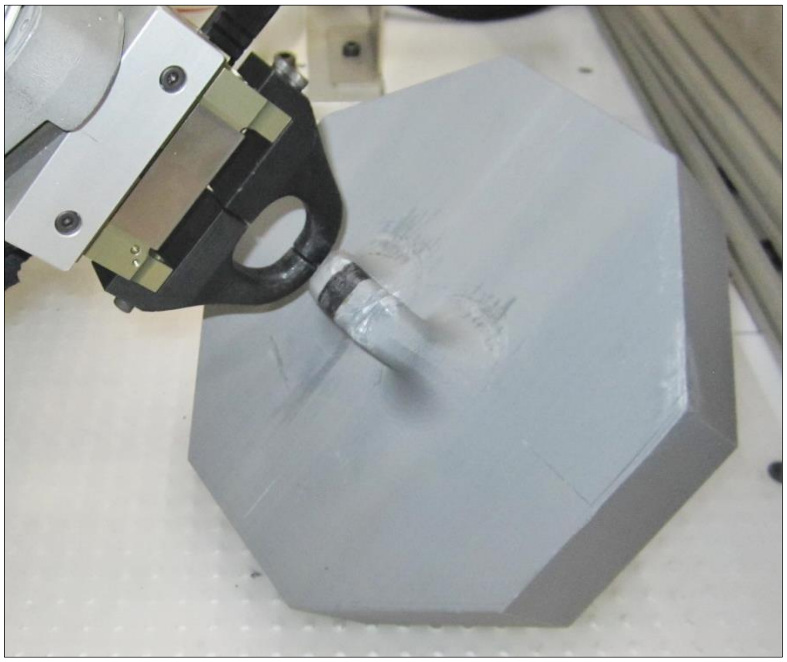

Figure 32: A failed grasp after a collision

3.9 Physical Testing Results Discussion

The IK solver success rate was lower than expected. Prior to all trials, we used our best judgment to determine if we thought the part was in a location that would allow for the robot to pick it up without colliding with the walls or itself. This is why we were surprised that the IK solver did not provide a solution for 50% of the trials. This could be caused by a number of things: an over-conservative collision avoidance model of the workspace and robot links, incorrect joint limit values, an incorrect configuration of the IK solver, or an improperly configured MoveIt package.

Unsuccessful grasps occurred mainly when the pose estimate was off side to side and the parallel jaw gripper did not have enough clearance to accommodate the misalignment as shown in Figure 31. All instances of this failure occurred in the back half of the workspace. The viewpoint of the Kinect was as good in this area as it was for the front of the workspace where this failure did not occur. While further testing would be needed to prove this, a poor calibration of the Kinect to the world coordinate system was most likely the root cause. While a gripper with a larger jaw opening travel could accommodate this error, it is likely solved by a more accurate calibration. The calibration routine used only a single fixed target. A possible solution would be to perform the calibration routine holding the target (Figure 9) in different locations and orientations and then creating an average transform from the multiple routines.

4 Conclusion and Future Study

We have presented an alternative solution to the object recognition pipeline with a shared autonomy approach. We showed that a system reliant on an unsuccessful object recognition system needs constant human intervention to maintain uptime. Our solution leverages the strengths of both the human operator and the automated system.

In general, the ROS framework and open source tools made this project possible. Without the availability of the tools or the plug and play capability of the tools, the scope of this project would have been restricted to a fraction of the final project. We were able to expand on the building blocks created by other researchers that came before us. The time spent troubleshooting and configuring open source software is much less than the R&D time to start from scratch. The open source community was also helpful during the troubleshooting phases. This allowed us to quickly filter out the different tools and programs that would and would not work for our scope without much time invested. Without the open source community of PCL, our entire scope might have been for product recognition, which ended up being a failure. However, because of the open source tools and documentation, we were able to quickly recreate the program for our situation and quickly determine a better path for our project. Because of this, the work that we have completed may be useful to someone else in the future. By publishing our project and source code, we give another student the chance to build off our work and take it another step further. For instance, future testing for this project would be needed before taking it into a production environment. An extrinsic calibration that was accurate and consistent in all areas of the workcell would be needed before moving forward. This could be created by an averaged transform calculated by multiple instances of the same routine that was described earlier.

The mechanical design of the lifting eye gripper was not a part of this scope. The design of a rigid and secure device would not be trivial. It would have to be compliant enough to account for the inaccuracy of the Kinect, while maintaining its rigidity to hold the casting while work was conducted on the part. It would have to be durable enough to withstand crashes when they occur.

Another option to ICP that was discussed during the project was to create a RANSAC torus program that was able to identify torus geometry in a point cloud similar to the RANSAC plane segmentation routine. While the lifting eye is not a perfect torus in all instances, the torus is the closest geometry shape that would identify the lifting eye. The RANSAC torus estimator has been in the PCL queue for a while, but has not been solved.

Future refinement in the ICP algorithm should be considered to increase the pose outcome success rate to closer to 100%. We propose two solutions for this. The first solution is to use a down-sampled point cloud that was segmented around the picked point and do a systematic guess and check of different rotations after the cloud had been translated. This step would be inserted directly before the ICP process in the current process. The second is to create a "bump" out of a local minima during ICP. For instance, if the ICP score was below a certain threshold, the pose was "bumped" to a random orientation within a volumetric window and ICP was started again. Each solution could be implemented and tested with the data gathered by the user study.

These alternative solutions are recommended to be tested prior to taking our proposed solution to a production environment.

5 Bibliography

[1] "Investment is the key to sweden's foundry automation: introduction," *Prod. Eng.*, vol. 58, no. 3, p. 26–, Mar. 1979.

[2] A. Pochyly, T. Kubela, V. Singule, and P. Cihak, "3D vision systems for industrial bin-picking applications," in *MECHATRONIKA, 2012 15th International Symposium*, 2012, pp. 1–6.

[3] K. Kim, J. Kim, S. Kang, J. Kim, and J. Lee, "Vision-based bin picking system for industrial robotics applications," in *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2012, pp. 515–516.

[4] A. Pochyly, T. Kubela, M. Kozak, and P. Cihak, "Robotic Vision for Bin-Picking Applications of Various Objects," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, 2010, pp. 1–5.

[5] Heping Chen, George Zhang, Hui Zhang, and Thomas A. Fuhlbrigge, "Integrated robotic system for high precision assembly in a semi-structured environment," *Assem. Autom.*, vol. 27, no. 3, pp. 247–252, Aug. 2007.

[6] U. Berger and R. Lepratti, "Intelligent PC-based user control interface for on-line correction of robot programs," in *7th International Conference on Control, Automation, Robotics and Vision, 2002. ICARCV 2002*, 2002, vol. 1, pp. 276–281 vol.1.

[7] L. Garber, "Robot OS: A New Day for Robot Design," *Computer*, vol. 46, no. 12, pp. 16–20, Dec. 2013.

[8] T. Fong, C. Kunz, L. M. Hiatt, and M. Bugajska, "The Human-robot Interaction Operating System," in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction*, New York, NY, USA, 2006, pp. 41–48.

[9] J. Kerr and K. Nickels, "Robot operating systems: Bridging the gap between human and robot," in *2012 44th Southeastern Symposium on System Theory (SSST)*, 2012, pp. 99–104.

[10] M. T. Hoske, "ROS industrial aims to open, unify advanced robotic programming," *Control Eng.*, vol. 60, no. 2, p. 20, Feb. 2013.

[11] S. M. Edwards, W. C. Flannigan, and P. T. Evans, "6DOFF Pose Estimation: The Need for Standardization in Industrial Applications," in *Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop*, New York, NY, USA, 2010, pp. 267–270.

[12] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," presented at the Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 1–4.

[13] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 80–91, Sep. 2012.

[14] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the Viewpoint Feature Histogram," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 2155–2162.

[15] W. Wohlkinger and M. Vincze, “Ensemble of shape functions for 3D object classification,” in *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011, pp. 2987–2992.

[16] R. A. El-laithy, J. Huang, and M. Yeh, “Study on the use of Microsoft Kinect for robotics applications,” in *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*, 2012, pp. 1280–1288.

[17] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Commun ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[18] S. Kahn and A. Kuijper, “Fusing Real-Time Depth Imaging with High Precision Pose Estimation by a Measurement Arm,” in *2012 International Conference on Cyberworlds (CW)*, 2012, pp. 256–260.

[19] D. Hershberger, D. Gossow, and J. Faust, *rviz*. ROS.org, 2013.

[20] A. Aldoma, “3D Object Recognition and 6DOF Pose Estimation,” presented at the ICRA 2013, Karlsruhe, Germany, 10-May-2013.

[21] D. Holtz, “PCL Basics,” presented at the ICRA 2013, Karlsruhe, Germany, 10-May-2013.

[22] A. Trevor, “Organized Segmentation,” presented at the ICRA 2013, Karlsruhe, Germany, 13-May-2013.

[23] T. Witzig, J. M. Zollner, D. Pangercic, S. Osentoski, R. Jakel, and R. Dillmann, “Context aware shared autonomy for robotic manipulation tasks,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 5686–5693.

[24] B. Sankaran, B. Pitzer, and S. Osentoski, “Failure recovery with shared autonomy,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 349–355.

[25] J. C. F. de Winter and D. Dodou, “Why the Fitts list has persisted throughout the history of function allocation,” *Cogn. Technol. Work*, vol. 16, no. 1, pp. 1–11, Feb. 2014.

[26] P. Michelman and P. Allen, “Shared autonomy in a robot hand teleoperation system,” in *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems ’94. “Advanced Robotic Systems and the Real World”, IROS ’94*, 1994, vol. 1, pp. 253–259 vol.1.

[27] B. Pitzer, M. Styer, C. Bersch, C. DuHadway, and J. Becker, “Towards perceptual shared autonomy for robotic mobile manipulation,” in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 6245–6251.

[28] D. A. Mindell, “Telerobotics, automation, and human supervisory control [Review],” *IEEE Technol. Soc. Mag.*, vol. 12, no. 3, p. 7–, Fall 1993.

[29] T. Kaupp and A. Makarenko, “Measuring human-robot team effectiveness to determine an appropriate autonomy level,” in *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008*, 2008, pp. 2146–2151.

[30] P. Mihelich, *rgbd_launch*. ROS.org, 2013.

[31] *industrial_extrinsic_cal*. SWRI, 2013.

[32] D. Gossow, *interactive_markers*. ROS.org, 2013.

[33] D. Holz, “PCL::Registration,” presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2011, San Francisco, California, USA, 25-Sep-2011.

[34] S. Edwards, *industrial_core*. SWRI, 2013.

[35] I. Sucan, S. Chitta, and A. Pooley, *MoveIt*. ROS.org, 2013.

[36] T. Field, J. Leibs, and J. Bowman, *rosbag*. ROS.org, 2013.

[37] A. Aldoma, F. Tombari, L. D. Stefano, and M. Vincze, “A Global Hypotheses Verification Method for 3D Object Recognition,” in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Springer Berlin Heidelberg, 2012, pp. 511–524.

6 Appendix

ESCO HMI PPE Study Pre-Questionnaire

Age:
Years working at ESCO:
Current job at ESCO:
Years at current job:
Gender:
Height:
Weight:
Average Computer Use (per week):
Average Tablet Use (per week):
Do you have gaming experience? If so, how much?
Are you a pet owner?
Do you wear glasses (not contacts)?
PPE Requirements (circle all that apply for current position):

Steel Toe Shoes
Ear Plugs
Ear Muffs
Light Gloves

Heavy Gloves
Safety Glasses
Safety Goggles
Face Shield (Papper Units)
Silvers
Other: ____

Health issues at work (circle all that apply)

Sore Muscles

Back Pain
Time off because of an ergonomic injury (back, shoulder injury, etc.)

Crush/Hand Injuries
Time off because of crush/hand injury
Time off because of other work related injury: __________________________

ESCO HMI PPE Study Post-Questionnaire

This system was intuitive:

Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree

The mouse was easy to use:

Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree

The roller-ball mouse was easy to use:

Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree

The joystick was easy to use:

Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree

The touchscreen was easy to use:

Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree

What kind of an affect would this system have on the safety of your job?

Strong Negative | Negative | Neutral | Positive | Strong Positive

What kind of an affect would this system have on your efficiency as a worker?

Strong Negative | Negative | Neutral | Positive | Strong Positive

If this system was optional over the typical crane hook, how likely would you choose this new system?

No Chance | Not Likely | Neutral | Somewhat Likely | Most Likely

Open-ended question:
What kind of a robotic/automated system would make you more safe or efficient at work?