Ruihao Wang
Fan Gao
Cameron Ridgewell
ME 5984: Experimental Robotics

Robot-based Teleoperation and Wrench Localization

Introduction

With the rapidly decreasing limitations of modern robotics, it is becoming increasingly possible to integrate robots into tasks that might prove unpleasant or dangerous to humans. Many developments of robots in this era center around robot's ability to operate in conditions that are hazardous to their human operators. Though the ultimate goal might be to achieve full robot autonomy, there are still a number of steps necessary to achieve robots that can perform even the most basic human tasks without any input from a human operator. In order to advance the capabilities of a standard SMP robot base, this team sought to develop a teleoperated system capable of remotely travelling to a target object and performing visual analysis on this object. This research was completed as a simulation and precursor to Challenge 2 of the Mohamed Bin Zayed International Robotics Challenge, which seeks to integrate aerial and terrestrial robotics in order to simulate potential urban disaster relief.

Background

The challenge described earlier focuses primarily on the semi-autonomous operation of a ground-based vehicle in order to recognize and manipulate a simple tool in order to open a square valve. This task requires a number of steps, most importantly the recognition and localization of the wrench used in the process. The members of the design team had previously encountered this task using camera vision and initially attempted to adapt their previous design into this design model. The previous design integrated 4 main features to identify a wrench from against a plain background. These features were Canny Edge Detection, Hough Line Transforms, Hough Circle Detection, and Harris Corner Detection all of which were implemented in OpenCV. Using these features, the team was able to accomplish some limited success in previous iterations of the wrench detection process.

Canny Edge Detection is a multi-stage algorithm that is able to extract perceived edges in an imported image. The image is processed for noise reduction, removing small pixels that might interfere with continuous processing of the image. This smoothed image is then filtered identify gradient patterns within the image. By detecting these gradients the algorithm is able to determine where the gradients end, so after interfering pixels are again removed, the algorithm performs hysteresis thresholding to determine which gradient ends, and therefore which edges, are real edges, and which are not. The entirety of this process can be controlled in OpenCV by manipulating threshold values to tell the program how sensitive to edges it ought to be.

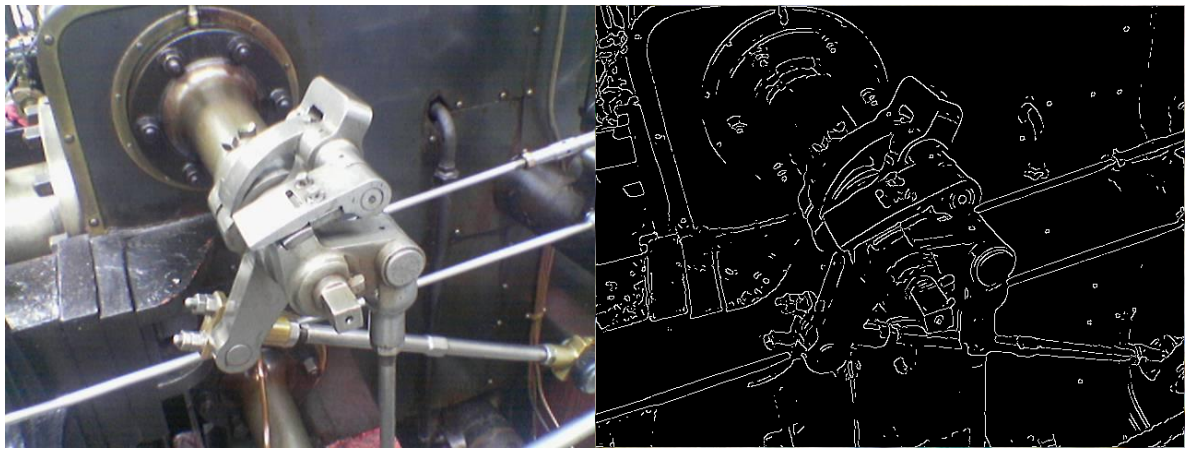

Figure 1.[1] *Canny edge detection is able to take a standard image and extract edge patterns.*

The Probabilistic Hough Line Transform and Hough Circle Detection operate the same principle. After a Canny Edge detection has been run, the resulting image is analyzed for continuous points in either a straight line or in a distinct circular orientation. The edges are analyzed for intersections of curves and, based on the number of curves that intersect along a line, outputs lines or circles that correspond to those intersections. These lines are then filtered based on maximum and minimum length or diameter properties as well as an estimation of how likely they are to be the predicted shape, an estimation established by the number of curve intersections that occur along their length.

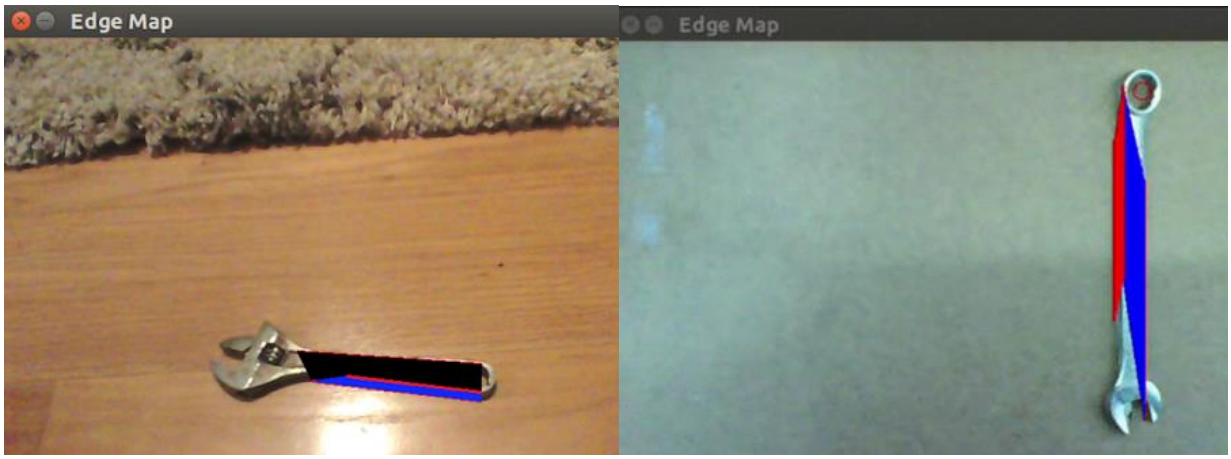

Figure 2. *Line (left) and circle (right) detection can be tuned to find distinct lines in an image. In our wrench detection algorithm, these were then filled in to paint the shape and orientation of the wrench. Only one circle was identified in these figures, corresponding to the head of the right wrench.*

The final aspect of the design that had been previously established was Harris Corner Detection. This detection also employs Canny Thresholding to begin its analysis. Once the edges have been found, the algorithm essentially traverses each, looking for points at which an edge both exists and changes direction drastically. Whenever such a point is identified, it stored and returned to the user.

[1] https://en.wikipedia.org/wiki/Canny_edge_detector#/media/File:Valve_monochrome_canny_(6).PNG

Using these tools, the team was able to identify a wrench by finding all of the distinct aspects of its design, the hole at the top, the shaft of the wrench, and the points of the manipulator end. These points were all connected to provide a picture of the wrench's location. Unfortunately, this design contained many problems. First, it could only be used to detect a single wrench; providing it multiple wrenches would cause difficulty as the program was unable to find the distinct endpoints in the horizontal direction, only in the vertical. On top of that, the program was extremely unstable. Do to changing lighting conditions and imprecise feedback from the cameras used, the image was constantly changing. Though this iteration saw a fairly consistent midpoint of the wrench, the orientation and physical features were constantly flickering, causing imprecise and unstable boundary measurements.

Objectives

In order to complete the desired performance criteria, the team had to be able to perform a number of operations. First amongst these was remote teleoperation of the robotic interface. Since full autonomy was deemed to be beyond the scope of the project, it became necessary to enable a system by which the team would be able to control the robot in the physical environment. This would be accomplished in two ways: first, via direction provided by visual feedback provided by the mounted camera, and second by navigation via a 2D map developed using SLAM protocol.

After accomplishing teleoperated navigation to the wrench location, it was necessary to provide reference locations to the base station pertaining to the robot's current location as well as the perceived location of the wrench relative to the initial starting point. This required recognition of the wrench and and positional analysis of the robot position as well as the wrench position in the global coordinate frame. The relative coordinate frame application would allow for direct manipulation of the wrench or another object relative to the robot, and the global reference frame provides transformation data from the starting point, i.e. near the base station, to the target location. This data is important to maintain correct direction of the robot's actions while being controlled remotely.

One final task, in order to demonstrate the potential for autonomy, was visual servoing based upon Pan-Tilt-Camera motion. While in our evaluations, an operator was always able to control all aspects of the robot via a Graphical User Interface and visual feedback system, this system is still prone to latency and has potential for error. In many robotic applications, such as extraterrestrial robots, communication latency can prove to be huge hindrances in machine performance, and is one of the main motivations behind encouraging robot autonomy in the first place. In order to establish some degree of operational autonomy, the team sought to establish a set of visual servoing protocols, allowing the dorsal-mounted camera to track the wrench through space while the robot was moving.

Additional minor requirements to the robot design were that it should be able to maintain operational parameters throughout the execution of the above-described tasks, and that it should provide a useful and intuitive platform for control and information gathering. This required that the robot have enough onboard power supply to provide power during all operation, and that wireless communication should be enhanced as much as possible to maintain connectivity throughout remote operation. The final aspect of this design required an intuitive GUI for robot operation.

Problem Formulation

The team was provided an SMP Robot that was originally semi-operational. Additionally, an array of feedback elements were provided in the form of Pan-Tilt and RGBD cameras and SICK and Hokuyo LIDAR systems. The team also had access to a number of mechatronic instruments and an Arduino Duemilanove microcontroller to assist in the development process. Operation was required to operate out of an EeeBox Mini PC, mounted to the robot's body.

Using these tools, the team developed a plan to complete the identified objectives. The first tasks consisted of getting the SMP Robot operational, adding ROS Indigo on Ubuntu 14.04 to the EeeBox, and determining the appropriate hardware to control the system. These tasks, while requiring extensive time, were not the primary challenges associated with the development of this project. After reformatting the computer and integrating Ubuntu into it, the team purchased a replacement motor driver for the SMP in the form of a RoboClaw HV 2x60A Motor Controller[2].

After getting all aspects of the system operational, the team began developing a control scheme for the execution of the design tasks. This consisted of the integration and adaptation of a number of existing software packages. These packages were *rosserial_arduino*[3], openCV, *rviz*[4], *hokuyu_node*[5], and *hector_slam*[6].

Approach

The final design of the robot included several key elements. On top of the SMP robot's wheels and metal frame, the team mounted two 12V batteries to provide power to the system. On top of the main body of the robot, the EeeBox was mounted in the rear, with the pan-tilt camera on top of it. This provided a useful vantage point for the camera, while still placing it out of the way of the anterior-mounted LIDAR system. It is important that no object interfere with the *x-y* plane of operation in which the LIDAR sits. This is because, in order to properly form a 2D map of the system, the LIDAR system must have an uninterfered view of the surrounding walls. This was accomplished by placing the LIDAR in the front of the robot, its blind side turned toward the higher objects mounted behind it.

Development of this design began with controlling the physical locomotion of the robot. This was accomplished via Pulse Width Modulated control between the RoboClaw Driver and the Arduino Duemilanove. This PWM control allowed the Arduino to operate the basic controls of the system by outputting a digital signal that mimics an analog signal. By sending out electrical pulses of varying duration and frequency, the Arduino is able to send commands for the robot to move forward and backward at difference speeds, turn left or right, and stop. The response of all of these calls contains a slight delay, a latency was that would later be further extended by the latency of wireless operation.The operation of this control scheme was then extracted from the Arduino IDE on which it had been developed and was placed under the control of ROS via the *rosserial_arduino* package. Using this package, an operator would be able to control the independent elements of control on board the microcontroller, ensuring a full range of motor control.

---

[2] http://www.ionmc.com/RoboClaw-HV-2x60A-Motor-Controller-60VDC_p_12.html
[3] http://wiki.ros.org/rosserial_arduino
[4] http://wiki.ros.org/rviz
[5] http://wiki.ros.org/hokuyo_node
[6] http://wiki.ros.org/hector_slam?distro=indigo

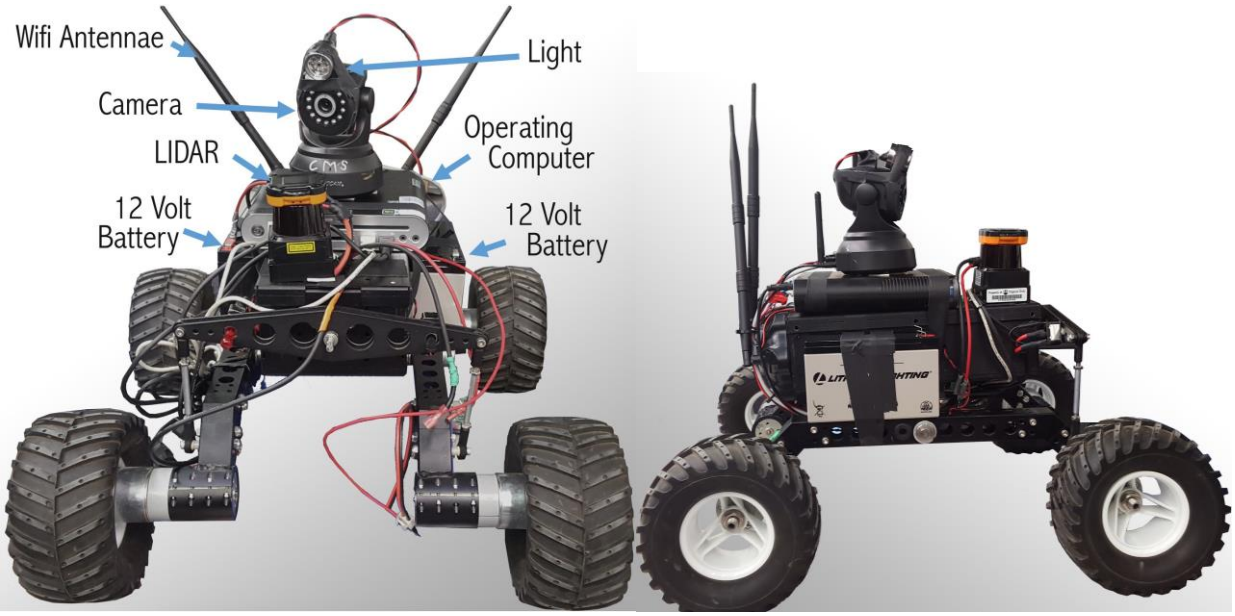

Figure 3. *The robot was constructed with a number of available hardware features, labelled above.*

During this design phase, there were also developments in the area of wireless communication. Though the team had originally sought to employ a Ubiquiti Wireless PicostationHP2 as a wireless bridge to communicate with the base station wireless network, the provided interface proved to be unsupported by the full system. Since the team had identified wireless communication as a potential problem area, a 100 ft rated wireless network antenna was purchased. This antenna was added to the EeeBox and mounted to the rear of the SMP robot, providing internet communication to the robot as it was driven further from the router.

In order to provide visual feedback to the system, a wireless, internet-based pan-tilt camera was mounted on top of the SMP Robot base. This camera had been used in a previous experiment and was thought to be an appropriate way to ensure a full range of visual control and address the visual servoing challenge. Since communication with the camera was based around not only the computer's connection to the internet, but also to an internal server within the camera itself, the team opted to remove the internet-connectivity of communication with this camera in order to reduce unnecessary latency. In order to accomplish this, the HTTP protocol communication with the camera was projected onto a separate subnet shared between the computer and the camera alone, outside of the local internet-based subnet. By using this separate subnet and an Ethernet crossover cable, the operation of the camera was taken from being internet based to being operable on the computer's personal subnet. This reduced communication and latency as well as wireless system complexity. Communication with the camera itself could then be accomplished using custom Python code to issue URLs to the camera's IP, commanding it to pan or tilt as the operator desired.

Commands for this system needed a intuitive control system. In order to accomplish this, the team developed an operational GUI that contained controls for both the movement of the camera and the robot's locomotion. Using this GUI, shown below in Figure 4, the operator was able to drive the vehicle

using the visual feedback provided by an OpenCV capture of the input HTTP stream from the pan-tilt camera. This allowed control of the robot system for driving capabilities.

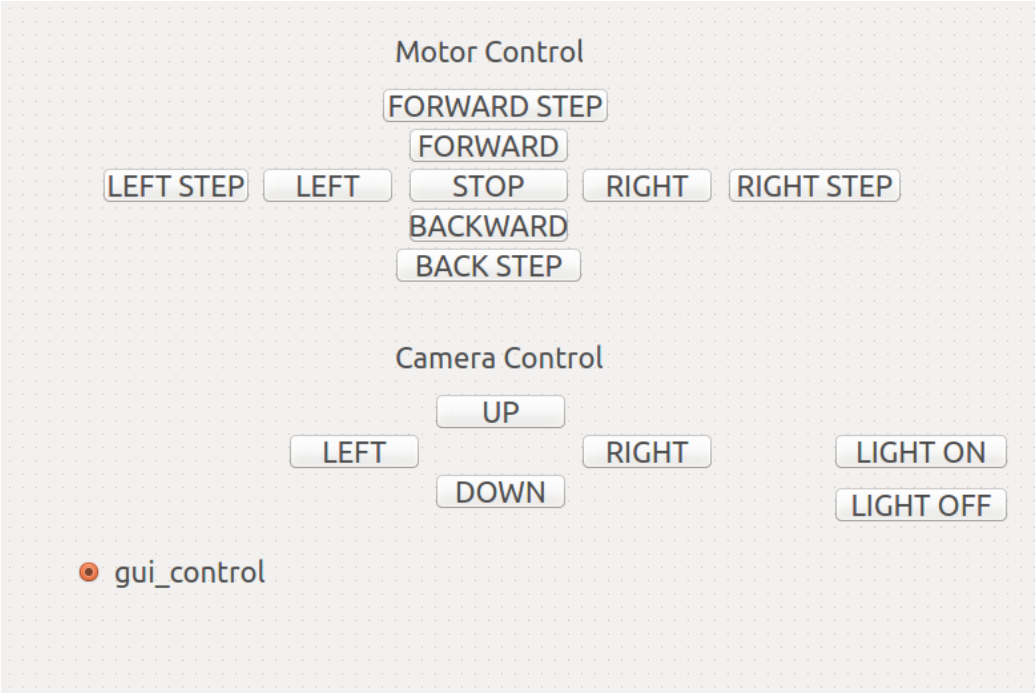

Figure 4. *Above is pictured the Qt-generated GUI used for control of the robot and its dorsal-mounted camera*

Wireless control of the robot was established using the GTK+ Remote Desktop Client, Remmina. Using this client onboard the EeeBox, the operator was able to view all open windows on the Ubuntu computer, even without an attached monitor. This system had both benefits and drawbacks. Its primary advantage came from ease of use through SSH tunneling in the VNC Server. Additionally, it was useful to be able to view all windows on the screen instead of only being able to operate certain commands over a typical, non-graphical SSH. The downsides of this approach were twofold: first, no matter what command or feedback was required, the network always had to send a continuous image, which contains a lot of data and does not deal well with wireless connectivity issues. Second, by mirroring the computer display, all processing power was performed on an underpowered remote mini PC rather than on the many faster computers available as base stations. This lead to performance issues, especially when trying to operate wirelessly with many different memory-greedy programs running in parallel. In spite of these drawbacks, this operation was deemed an appropriate way to control the system and allowed for control to leave the lab bench as the robot could now be remotely controlled and powered.

Power was provided by two side-mounted 12V batteries. One battery was used to power the Hoyoku LIDAR system and the 4 DC motors controlling the wheels. The other battery was used to power the EeeBox, and by proxy the Arduino, the wireless internet antennae hub, and a set of LED lights that were mounted atop the camera to provide consistent lighting conditions during image analysis. This power connection was also stepped down to 5V to provide input power to the pan-tilt camera.

After completing the locomotion and connectivity parts of the challenge, the team took on the challenge of adapting 2D mapping to the LIDAR system mounted atop the SMP Robot. This was completed by using *hokuyo_node* and *hector_slam*. *Hokuyo_node* is driver package for a Hokuyo LIDAR system and was able to transform boundary information achieved from the LIDAR laser input into a ROS node. *Hector_slam* subscribed to the published node and applied simultaneous localization and mapping to the static transform publisher to continuously update the map, which was then presented in real-time in rviz.

Due to the issues with the more complex detection algorithms described earlier, the team fell back to some of the resources that had been previously used in other experiments. Primary amongst these was the ROS package *find_object_2d*[7]. This package uses SIFT, SURF, FAST, and BRIEF to perform feature detection on an identified object. By giving the software an image of the object that it is looking for, *find_object_2d* is able to track the object in real time, providing coordinate data as well as a bounding box to indicate position within the image frame. Using this coordinate data, the team was able to create a bounding box in which to analyze the input image for the smallest wrench in a line. To improve over the team's previous iterations, this system was able to achieve the desired stability while detecting the wrench even when other similar shapes were present.

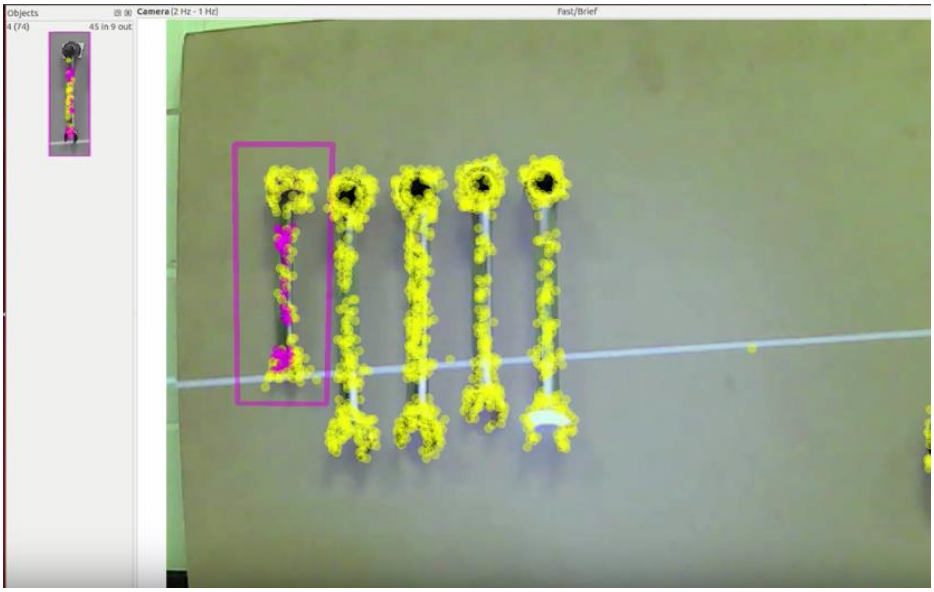

Figure 5. *Find_object_2d is able to track objects in real time by following unique points on the object's surface and outline.*

Finally, using this wrench recognition protocol, the team was able to create a visual servoing operation. This was a simple matter of combining the detection coordinates provided by *find_object_2d* and integrating them with the pan-tilt camera control implemented previously. This program, when run, adapts the coordinates to a perceived centerpoint, then moves the camera, after a brief delay ensuring that the camera moves in a controlled manner. The camera is moved by the controlling program in order to place the input coordinates close to the center of the input camera frame. This allows the camera to follow the wrench as either wrench or the robot is moved in reference to each other. This implementation

[7] http://wiki.ros.org/find_object_2d

has the limitation that the wrench, if moved, cannot be removed from a static plain background, or else the identified points would likely be difficult to reconstruct. However, in the event of no wrench detection, the camera does not autonomously seek out a wrench, returning to user-based control.

Experimental Results

Though initial runs of the competition course took as long as 20 minutes, as the operator learned the controls and constraints of the system better, the robot was able to travel to the wrench board and orient itself for detection is less than 6 minutes. During this process, a map, shown below in Figure 6 was recreated. This map displayed a reference axis at the robot's starting point as well as a

Figure 6. *The SLAM package was constantly logging data throughout the teleoperation, allowing live-updated 2D mapping feedback*

second axis at the robot's current position. With this feedback, it was possible to navigate without directly using the pan-tilt camera. This mapping also provided positional data with regards to the relative and global reference frames of the wrench and robot locations. With the *x* axis aligned straight along the first hall, and the *y* axis aligned straight along the second, the SLAM package gave the final robot position as (21.3, 17.9) meters. The wrench itself was localized as being -0.68 meters back along the *x* direction when the robot was properly aligned with it. This gives a global coordinate position of (20.6, 17.9) meters for the wrench, with no change in the *z* direction.

To confirm these values, the team measured the distance in both the X and Y directions using a standard measuring tape. These values were found to be (21.7, 18.6) for the position of the robot, and (20.8, 18.6) for the position of the wrench. These values are approximate as well, given the inherent error to which human measurement is prone. However, if they are considered accurate, this gives a maximum

LIDAR error of 1.8% in the X direction and 3.8% in the Y direction. While these values are not ideal, Hector SLAM is not perfect, and some error is to be expected.

The actual wrench identification was carried out reliably, with visual servoing tracking the wrench easily on robot rotation steps. The wrench was identified using the calibration image in

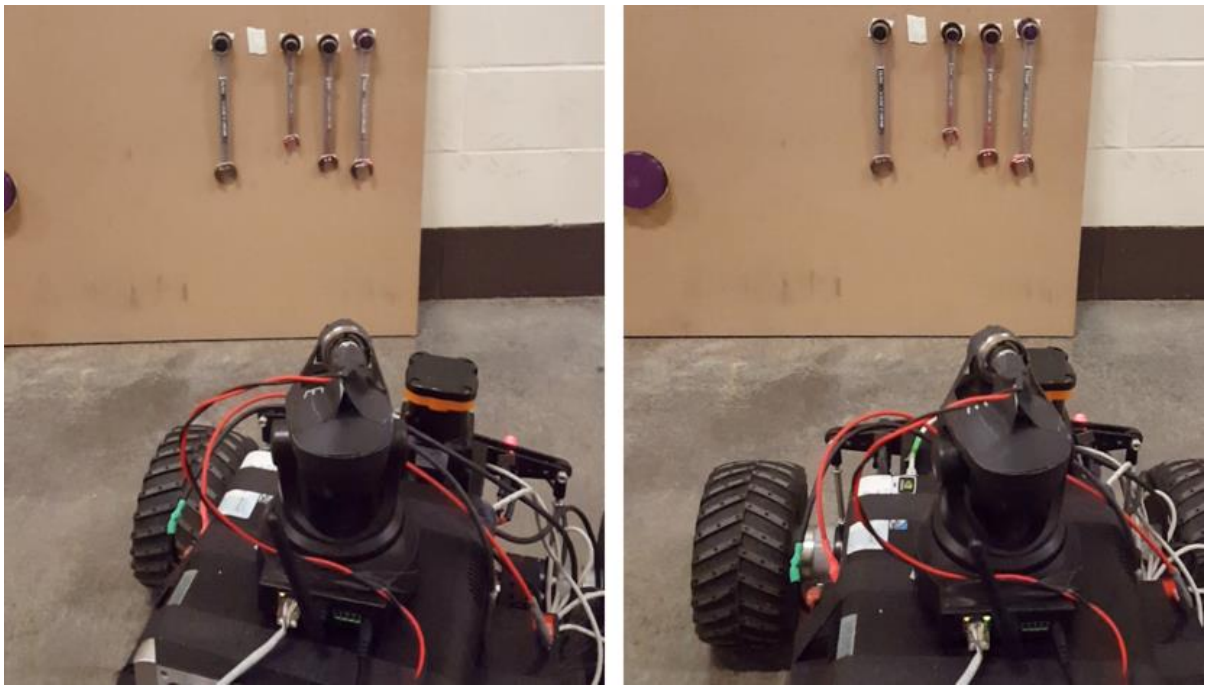

Figure 7. *The camera was able to easily move to compensate for robot rotation, recognizing the center point of the smallest wrench and autonomously orienting the camera to face it.*

find_object_2d which was shown earlier. The stability of the identification was considerably better than the previous circle-detection-based model, which would not have been able to pick out specifically the smallest wrench from this line. Using this ROS package, the robot tracked the wrench's center point with limited variance. Below we show a scatter plot of the wrench's located positions. As can be seen, the variance is relatively low, with an average value of (256, 105) with a standard deviation of (11.2, 22.5).

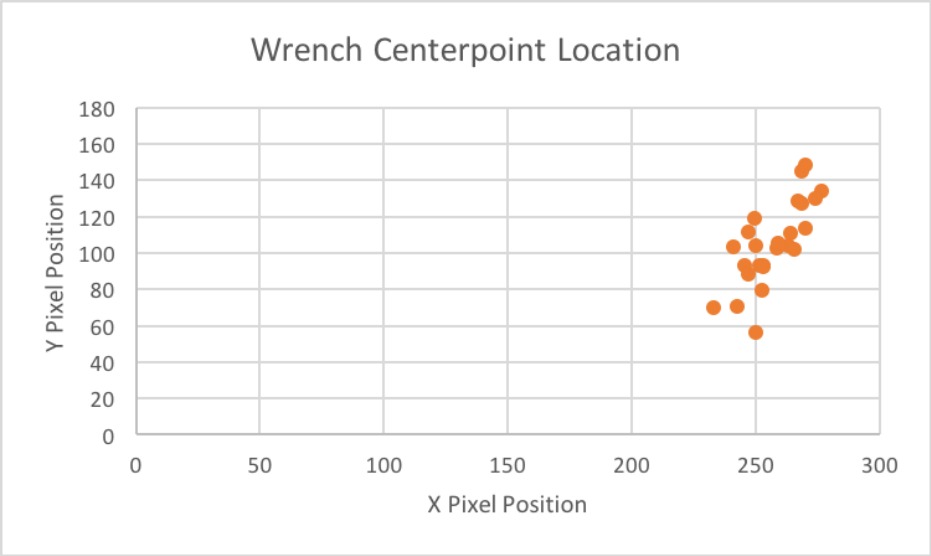

Figure 8. *The center point was unmoving during this analysis the wrench detection. Ideally there would be no variance in these values.*

These values are representative of the pixel interpretation of the centerpoint of the wrench. To offer some perspective, on the 640 x 480 pixel resolution of the ROCAM output feed, the variance in the Y direction accounts for 4.7% of the available display, while the X variance is a mere 1.7%. This means that the overall error caused by either of these interpretations is largely negligible. Looking at the above graph with the full range of possible X and Y values, it can be seen how precise the measurements actually are:

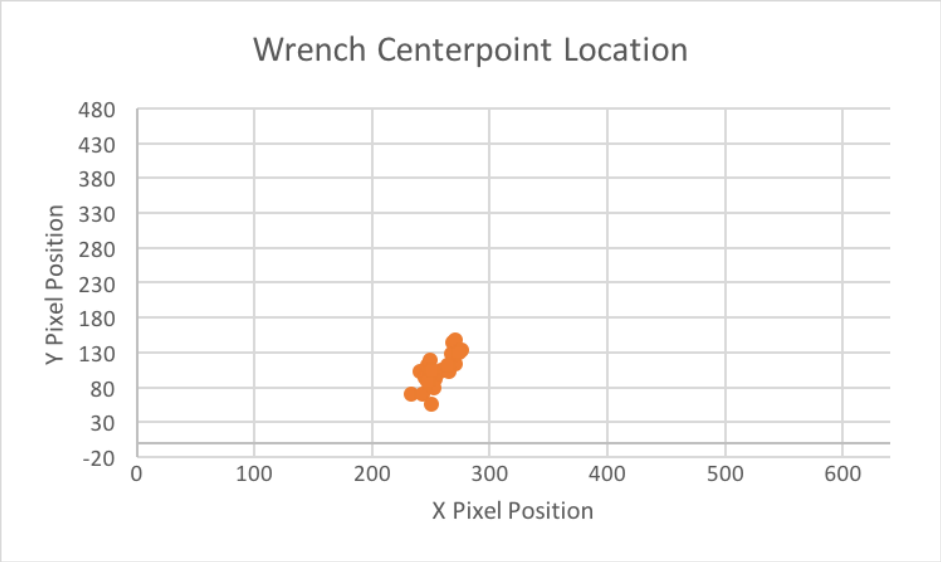

Figure 9*. Out of the entirety of the available screen, variance was quite low over the 25 trials.*

Conclusions and Future Work

The team was able to complete all of the desired objectives to a satisfactory degree. Wrench localization was achieved with minimal error. While it would prove useful to establish an even more precise methodology for localization, it could be difficult to improve over the 1-5% that is already achieved. Similarly, though it might be possible to establish a more precise method of identifying a wrench in the object frame, the amount of error in the system is low enough that this is not a priority for future study. However, there are still a number of improvements that could be made on the system as a whole.

Currently, to make sure the movement of vehicle is stable enough, the speed is set up to be slow (about 0.1m/s). While the reasoning behind this was to provide a better mapping speed, in the future, the team will look to modify the existing Arduino code by adding a speed control adjustment, allowing operators to adapt the speed according to the environment and objectives. to address wireless communication, the team will update it using XBee, which has longer operating distance, stable output, and high anti-jamming capability. With the capabilities of this device, the team can fix the current wireless communication problem: that WiFi connection is unstable due to DHCP IP address swaps between routers. At the same time, every team can get a better communication with their robots without other teams' influence while sharing the same WiFi. Final development would promote autonomy; instead of operating the robot manually, the team expects to develop a ROS node to realize automatic movement control based on mapping. With these adaptations, our robot would be better suited to accept integration of the robot arm and operational parameters of the Mohamed Bin Zayed International Robotics Challenge.

Team Participation:

Team Member	**Accomplishments**	**%**
Ruihao Wang	Assisted in developing overall mechatronic system, primary engineer in the troubleshooting and calibration of the robotic locomotion system, developed global and relative frame localization methodology, assisted in development of 2D mapping protocol, acted as primary controller of teleoperated robot	33%
Fan Gao	Researched circle detection methodology, primary developer of wrench detection system, assisted in developing overall mechatronic system, assisted in the troubleshooting and calibration of the robotic locomotion system, established 2D mapping protocol	33%
Cameron Ridgewell	Assisted in developing overall mechatronic system, assisted in the troubleshooting and calibration of the robotic locomotion system, performed evaluation on previous communication systems, established new wireless communication protocol, provided camera interaction functionality, integrated ng on previous wrench detection processes, assisted in the hardware onto final robot design, performed troubleshooti development of final wrench detection processes	33%