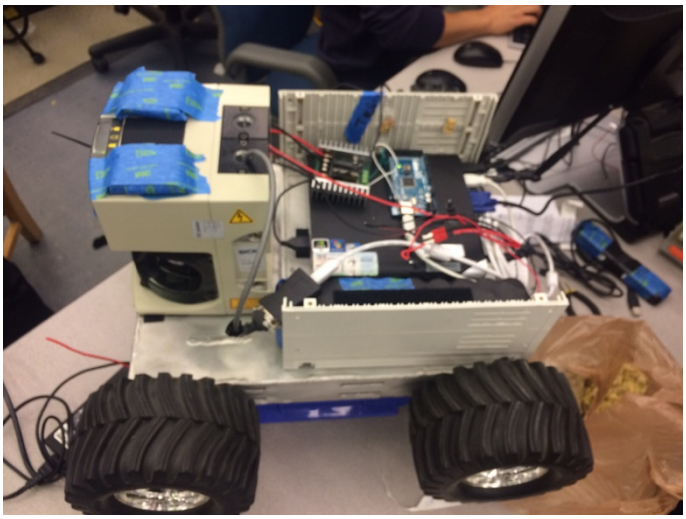
# ME5984 SS: Experimental Robotics

# Final Report



Team #1

By,
Mengyu Song
Sai Tej Paruchuri
Mingyi Liu
Bhavi Bharat Kotha

## Statement of Responsibility

Task 1: Hardware and Software Integration
Task 2: Mapping, Navigation and Global Localization
Task 3: Human-Machine Interface and ROS Networking
Task 4: Wrench Detection and Local Co-ordinate System

| | Weight (1 or above) | Mengyu Song | Sai Tej Paruchuri | Mingyi Liu | Bhavi Bharat Kotha | Sub-total | Total (Weight * Sub-total) |
|---|---|---|---|---|---|---|---|
| Task 1 | 1 | 25% | 25% | 25% | 25% | 100% | 100% |
| Task 2 | 1 | 25% | 25% | 25% | 25% | 100% | 100% |
| Task 3 | 1 | 25% | 25% | 25% | 25% | 100% | 100% |
| Task 4 | 1 | 25% | 25% | 25% | 25% | 100% | 100% |
| Total | 4 | 100% | 100% | 100% | 100% | | |
| Normalized | 1 | 25% | 25% | 25% | 25% | - | 100% |

## Background

A vehicle which runs without a person on board is called as an uncrewed vehicle or an unmanned vehicle. These kind are usually controlled by a person sitting in a room called the "server room" and controls the vehicles remotely. Sometimes they are autonomous, meaning they are partly programmed with artificial intelligence codes which help it to sense and navigate their environments without any human interaction. There are many types of unmanned vehicles: Unmanned ground vehicles (autonomous cars), unmanned surface vehicles, autonomous underwater vehicle, unmanned spacecraft, etc. [1]

There has been a lot of talk regarding drones or unmanned vehicles in the recent years. There are many advantages of having an unmanned vehicle. The human error is completely eliminated which helps in reducing the maintenance costs to a large extent. Present-day unmanned vehicles also have more fuel efficiency and have longer operational durations. In the fields like precision agriculture and public safety, unmanned vehicles contribute and can accomplish tasks with a much greater efficiency than humans. Some of the other fields in which UAV's can make a major contribution are wildfire mapping, agricultural monitoring, disaster management, thermal infrared power line surveys, telecommunication, weather monitoring, aerial imaging/mapping, oil and gas exploration, etc. Unmanned Aerial Vehicles contribute a lot in the sector of defense and can help in lowering the human causalities as humans are far away from the battlefield. [2]

The following are the few important sectors which help in making an effective unmanned vehicle:

1. Sensor Fusion: Using different sensors and combining the information we get from the sensors will provide us more data to make the vehicle less human dependable.
2. Communications: Handling communication and coordination between multiple agents in the presence of incomplete and imperfect information.
3. Motion/Path Planning: Determining an optimal path for the vehicle to go while meeting certain objectives and constraints, such as obstacles.
4. Obstacle Avoidance: Sensing the environment for obstacles

One of the first semi-autonomous vehicle was developed by General Motors. The General Motors' Firebird II was programmed to move into a lane and follow it along and this was made possible with the help of a metal conductor placed on the car. Google Self Driving Cars are one of the current project running in this world providing us with a completely autonomous ground vehicle helping in transportation of people. Their objective is to provide with an access to safe transportation for everyone and especially for people who are not capable of driving on streets [3]. According to lot of surveys 94% of accidents in the U.S. involve human error. By having autonomous vehicles, we can drastically reduce the number of accidents and deaths due to this problem.

There are many challenges in developing an unmanned vehicle. One of the main problem for a software platform dedicated for running and computing data for an Unmanned Vehicle. The absence of fast computing power provides delays and in turn creates problems for the vehicle. Mapping and self path planning are also one of the difficult areas in this field. Also one of the most important problem when it comes to designing a unmanned vehicle comes in critical decision making. A scenario where an autonomous car has to decide whether to swerve right or left, for instance – either injuring three people in a truck or potentially killing a person on a motorcycle is an ethical dilemma. [4] Some of the practical problems which the Google Car is still facing are intense weather conditions, potholes, roads that haven't been thoroughly googled (i.e., google maps), old road constructions, unexpected human interactions, dense traffic areas. [5]

## Objectives

Obtaining a recognition algorithm to locate an object of our desire. This area of research can contribute in the area of obstacle avoidance for an autonomous vehicle helping it to become more robust.

Making it unmanned by using a software which relays commands to the vehicle from a server station.

Mapping the travelled area using a fusion of hardware and software. This helps in path planning and decision making of the unmanned vehicle.

Problem Formulation

The following are the tasks which were fulfilled for our competition:

1. Identify the location of the shortest wrench hanging on the wall in a global coordinate frame.
2. Tele-operation to wrench board.
3. 2D mapping of the environment.
4. Automatic Visual Detection.
5. Visual Servoing.
6. Relative Localization Accuracy.
7. Global Localization accuracy.
8. Time limit for the tasks to be completed is 10mins.

## Approach

ROS (Robot Operating System) was used as the software element for achieving the tasks for the competition. ROS is an open source software development feature which provides a functionality like an operating system. It has a vast collection of tools and libraries to help developers in the fields of software to create varied application for development of robots. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, etc. ROS is used in many areas like Perception, Object Identification, Segmentation and recognition, Face recognition, Gesture recognition, Motion tracking, Egomotion, Motion understanding, Structure from motion (SFM), Stereo Vision, Motion, Mobile Robotics, Control, Planning, Grasping, etc.

For the locomotion of the vehicle, four DC motors were used which were controlled with the help of the Sabertooth Motor Driver and the Arduino Mega micro-controller (as shown in figure 1). Sabertooth 2x25 is a motor controller which was used to control the DC Motors with the help of the Arduino.

- Input Voltage: 6-24V nominal, 30V absolute max
- Output Current: Up to 25A continuous per channel. Peak loads may be up to 50A per channel for a few seconds [12].

The simplified Serial Mode configuration was used with a baud rate of 38400. Figure 2 is Dip Switch Configuration for Simplified Serial Mode with a 38400 baud rate.

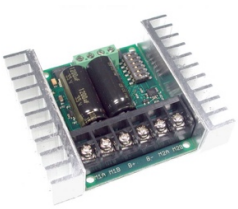


Figure: 1

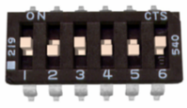



Figure: 2

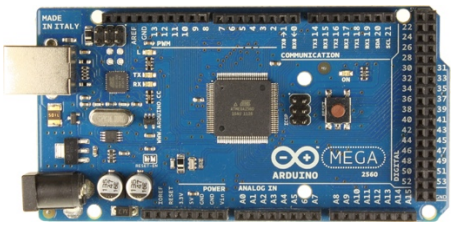



Figure: 3

Arduino Mega 2560 (as shown in Figure 3) is a microcontroller board used to interface between the motor controller and the human element. It has 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header and a reset button. The Serial Port (UART) was connected to the motor controller as the S1 supply signal and was used to control the motors [13].

The following are the list of the other hardware which we used in our vehicle to achieve our tasks:

1. Logitech wireless gaming controller (figure 4) to control the vehicle from the server room with the help of the master computer. The “joy” ROS package was used in ROS for integrating the joystick hardware to the software.

Figure: 4


Figure:5


Figure: 6

Figure: 7

2. SICK LIDAR (figure 5) for mapping our environment [14].
3. Asus Xtion camera (figure 6) for formulating the depth and the RGB image of the environment.
4. EEE PC box (figure 7) as the onboard computer.
5. USB wi-fi adapter (figure 8) to increase the range of the wi-fi signal.
6. We used two 24V battery pack on board the vehicle to power the whole vehicle.

All these components have been integrated together to achieve the final result.

Figure 8

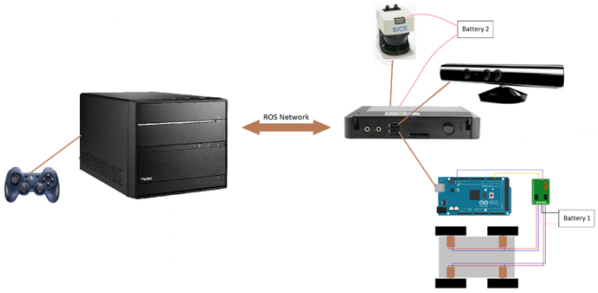



Figure: 9

## Human-Machine Interface, Networking and Joystick Integration

ROS is a distributed computing environment. A running ROS system can comprise dozens, even hundreds of nodes, spread across multiple machines. Depending on how the system is configured, any node may need to communicate with any other node, at any time.

The task (remote control and detection) requires two computers, i.e. the host (onboard computer) and the client (display and command computer) to communicate. A reliable network must have established by connecting the two computers together. First we tried wireless communication network: CMS-Wireless. The problem is that the signal coverage is not strong enough when the vehicle got the corridor, so we build a LAN. Client computer is connected to the hub via a internet cable and host computer is connected to the hub via wireless signal. In order to cover the entire range of the route, the hub is placed at the sharp point of the L shaped route, i.e. the middle point of the route. Here is the description of the detailed setup and configuration of ROS network.

To get to know the IP addresses, we look them up in the connection information menu under the wifi icon of the task bar. Then we get the administrator rights to write access the hosts file and save. So, open the editor by evaluating the following command line:

sudo gedit /etc/hosts

Then change the host and client IP address to the IP address we get from connection information category.

==========

10.1.160.91 host
10.1.160.36 client

==========

ROS needs to know the name of each machine it is running on. This done by setting the ROS_HOSTNAME environment variable of each machine with its own name. Besides, we should provide ROS with URI of the master node, that is launched by the roscore command. Since in a ROS graph, there should be only a single master node, its URI should replicated on all machines by setting the ROS_MASTER_URI environment variable.

In our setting, we chose to run ROS master node on host. So, in host we set the environment variables as following:

export ROS_HOSTNAME=host
export ROS_IP=host
export ROS_MASTER_URI=http://host:11311

On client computer, we set the environment variables as client.

For each of the Linux terminal we opened, we configure them in the example above. In this way, once we open an roscore on master, all the other nodes subscribe the same roscore, i.e. the roscore running on the master computer. Since the on board computer (master) is comparatively old, i.e. less calculation power and slow, this is a very good way to finish the task since the nodes running on client computer use the calculation power on its own, while only subscribe the information from the host computer. Every time we run a node on the client computer with right ROS environment configured, we will use the calculation power on the client computer, which will not drain the host computer resources. At the same time, the node running on the client computer will subscribe and use all the information on the host computer.

Joystick

Although we do not use complete autonomous system to navigate the vehicle, feedback control is needed. Real time video flown of the vehicle's sight is shown on the computer screen. One of our teammate serves as the operator: control the vehicle by looking at the screen. In order to get better performance, we use joystick to control. On the joystick, the definition of the joystick

| Button | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Function | Left Turn | Backward | Right Turn | Forward | Stop | Slow Forward |
| Method | Left:CCW Right:CW | both CCW | Opposite to 1 | both CW | both stop | both slow CW |

Table: 1

CW: clockwise CCW: counter clockwise

The function is connected with the correspondent wheel movement.

## Navigation

One of the goals for an autonomous robot operating in an unknown environment is to be able to construct a map and to localize itself in it. Robotic mapping deals with the study and application of ability to construct map by the autonomous robot and to localize itself in it. In robotic mapping, simultaneous localization and mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.

The SICK LMS 291 Laser Measurement Systems is a non-contact measurement systems (NCSDs). The system scans its surroundings two-dimensionally with a radial field of vision using infra-red laser beams (laser radar). Such laser measurement systems can be used for area monitoring, object measurement and detection, determining positions.

SICK LMS 291 technical specifications

- Operating Voltage: 24V
- Required Power: 20W
- 75Hz Scan rate over 180$^{o}$ range
- 0.25$^{o}$ angular resolution
- Sensing Range: 30m at 10% reflectivity; 80m of max. range
- Error: 10mm

ROS Packages used are the following:

1) Laser range finder driver package

Siscktoolbox/Sicktoolbox_wrapper packages were used to obtain scan data from the SICK LMS 291 laser range finder. [8]

2) Mapping package

Hector mapping is a simultaneous localization and mapping (SLAM) package that can be used without odometry. It provides 2D pose estimates at scan rate of the sensors. Even though this system does not provide explicit loop closing ability, it was used for mapping of unknown environment by the tele-operated robot because it is considered to be sufficiently accurate for many real world scenarios. The system has successfully been used by members of robotics community on Unmanned Ground Robots, Unmanned Surface Vehicles, Handheld Mapping Devices and logged data from quadrotor UAVs [6]

Implementation:

- The laser data from SICK is obtained using sicktoolbox_wrapper package [7]. This package is run in the client computer (robot).
- The laser data is converted to GeoTIFF using Hector Mapping package. The launch file on host computer uses a transfer function to convert the laser scan data to the map data.
- The GeoTIFF file is visualized using rviz package on host computer.

Figure: 10

## Global Localization

Robots navigating in unknown environments have to be localized with respect to a fixed point so that it is always possible to keep track of the total distance travelled form the origin. This information might be necessary when the robot has to be navigate back to its origin or to different points in the map. The global coordinate of the tele-operated robot has been obtained using the measure tool in rviz on the map created. The measure tool gives the global coordinate measurement in meters. Since the measure tool doesn't necessarily measure distance along the

axis or in straight lines, there is possibility for error in the distance measured. The following table shows the actual distance and the global coordinates measured using the measure tool in rviz.

| | | x-coordinate (m) | y-coordinate (m) | % error in x | % error in y |
|---|---|---|---|---|---|
| Actual Values | | 20 | 20 | - | - |
| Measured Global Coordinates | Trial 1 | 21.67 | 19.56 | 8.35 | -2.2 |
| | Trial 2 | 20.47 | 20.34 | 2.35 | 1.7 |
| | Trial 3 | 19.92 | 19.54 | -0.4 | -2.3 |

Table: 2

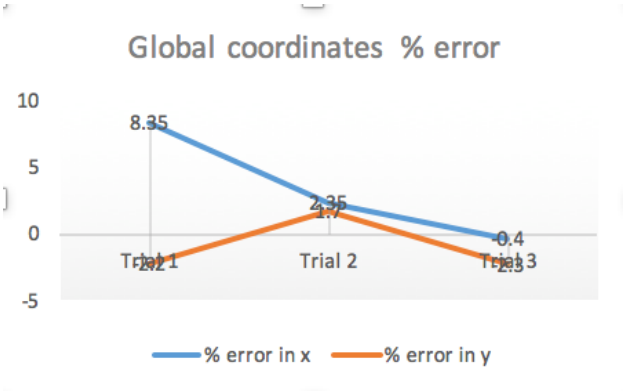



Figure: 11

## Wrench detection

We use the depth image from Kinect RGB-D camera as the input for the purpose of wrench detection.

To simplify the task, we assume the wrench is placed on a planar surface.

Then after three steps, we can get the position of each wrench, and the relative size order of all wrenches.

Step 1: Use Least Square Fit Approach to get the equation of the planar surface.

The equation of a plane can be expressed as:

$$a_1 x + a_2 y + a_3 z + a_4 = 0$$

Define $A$ as

$$A = [a_1, a_2, a_3, a_4]^T$$

Since only 3 coefficient is required, we can force $|A| = 1$

And use homogeneous coordinates to represent 3D point:

$$X = [x, y, z, 1]^T$$

The equation becomes:

$$A^T X = 0$$

Now consider an arbitrary point $X_0 = [x_0, y_0, z_0, 1]^T$ and the concerned plane $A^T X = 0$

The distance from $X_0$ to the plane can be written as

$$d_0 = |A^T X_0| / l$$

where

$$l = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

Notice $[\frac{a_1}{l}, \frac{a_2}{l}, \frac{a_3}{l}]$ is the plane's normal

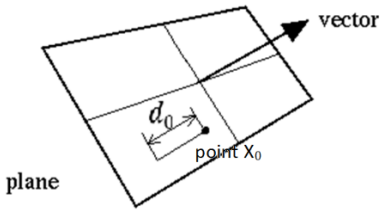



Figure: 12

Then the problem becomes to find the value of A to minimize $\sum d_i^2 = \sum \left( \left| A^T X_i \right| / l \right)^2$

Notice that $l$ is identical for every points, than the problem becomes to minimize $\sum \left| A^T X_i \right|^2$

$$\sum \left| A^T X_i \right|^2 = \sum A^T X_i \left( A^T X_i \right)^T = \sum A^T X_i X_i^T A = A^T \left( \sum X_i X_i^T \right) A = A^T \left( \bar{X} \bar{X}^T \right) A$$

where $\bar{X}$ is a $4 \times n$ matrix, containing all the points need to be fitted

$$\bar{X} = \left[ X_1 X_2 X_3 \ldots \right]$$

Let $\mathrm{L} = \bar{X} \bar{X}^T$ , $L$ is a $4 \times 4$ matrix, then

$$A^T \left( \bar{X} \bar{X}^T \right) A = A^T L A$$

Based on the knowledge of linear algebra

$$\min \left( A^T L A \right) = \min \left( \left| \lambda \right| \right)$$

$\lambda$ is the eigenvalue of $L$, and its corresponding eigenvector is the value of $A$ to achieve minimum.

Step 2: Compare the pixels from depth image. Find the difference between the points/pixels of the planar surface to the other set of points. Use this obtained difference to find out the location of the rest of the points/pixels with respect to the planar surface.

For a certain point $X_i$ from the depth image,

if $A^T X_i = 0$ then the point is on the plane

if $A^T X_i > 0$ then the point is above the plane along the direction of the normal vector

if $A^T X_i < 0$ then the point is below the plane along the direction of the normal vector

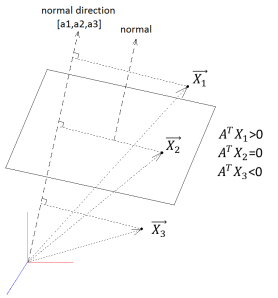



Figure: 13

For every pixel in depth image:

if (it is above the plane)

set its value as 1

else

set its value as 0

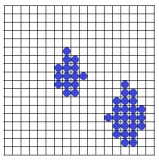


Figure: 14

In Figure 14, all pixels with value 1 are represented with blue dots, while other pixels are remain blank. The blue dot means that at that position under world coordinate system, there is something above the planar surface, indicating that some part of some wrench is at that position.

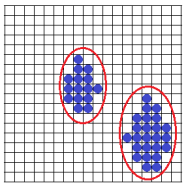
Step3: After finding out the pixels or points of the wrenches, the points need to be grouped to represented different wrenches. And we can also get the relative size order of all the wrenches by calculating the area of each wrench image.

The algorithm of grouping points is described as below:

For every point with value 1

If it has a neighbor with value 1, then they belong to same object.

The way to calculate the size of wrenches are simply as counting the number of pixels of each group.



size of object 1 is 15

size of object 2 is 24

Figure: 15

Results of wrench detection:

The original depth image from Kinect is:

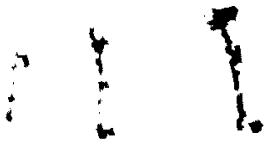


Figure: 16

The result after Step 1 and Step 2.

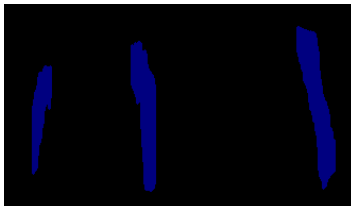


Figure: 17

The result after Step 3, notice 0 indicates the largest wrench, while 2 indicates the smallest one.

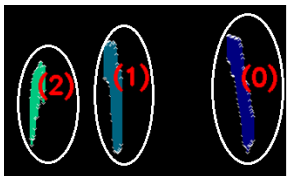


Figure: 18

The following figure shows the result for detecting the smallest wrench of 5 wrenches.

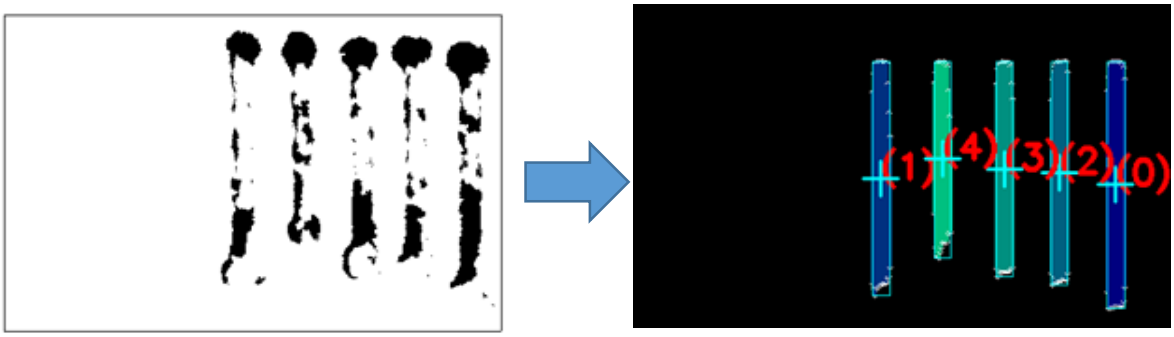


Figure: 19

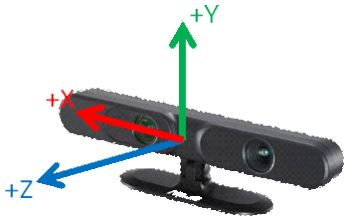


Figure: 20

### 3D Point Cloud

A point cloud is a set of data points in a three dimensional coordinate system where the position of each data point is defines by x, y, z coordinates.

Packages used for complete the task:

1) Openni2_launch

Openni2_launch package contains launch files for using OpenNI-compliant devices in ROS such as Asus Xtion, Xtion Pro, and multiple version of the Primesense cameras. [11]

2) Rtabmap_ros

The 3D point cloud is obtained using rtabmap_ros. RTAB-Map is a RGB-D SLAM approach based package that get rgb image and depth image from the Asus Xtion sensor and creates a 3D point cloud that can viewed in rviz. [9]

Procedure implemented are the following:

1. The color and depth images from the Asus Xtion camera are obtained by running the launch files of openni2_launch package
2. The 3D point cloud is obtained in rviz using the following command [10]

roslaunch rtabmap_ros rgbd_mapping.launch rtabmap_args:="--delete_db_on_start" rviz:=true rtabmapviz:=false

## Experimental Results

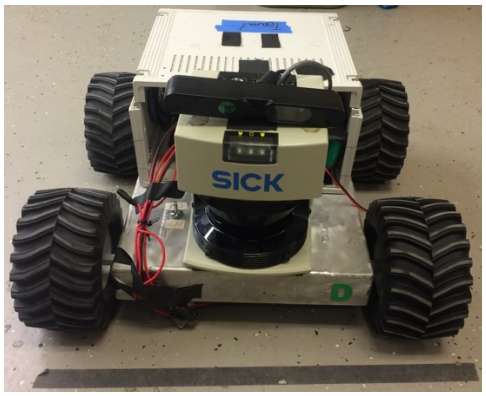



Figure: 21: Initialization - Tele-operated robot is set ready for navigation to the wrench board.

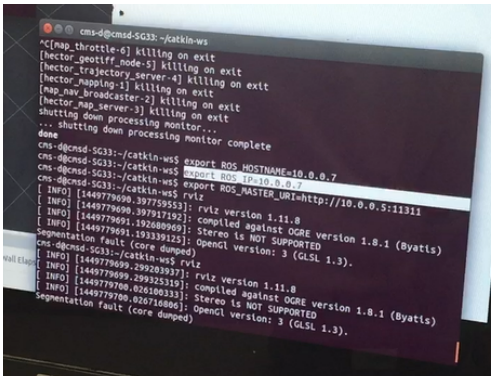



Figure: 22: ROS Networking – The above picture shows how the Master Client communication is setup on the client computer.

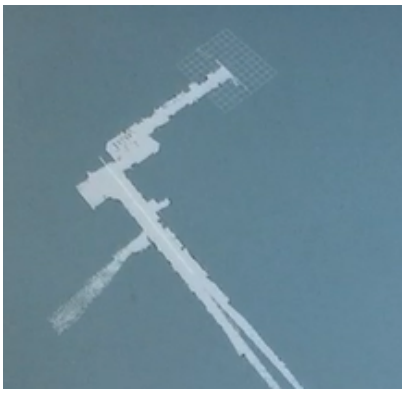


Figure: 23: Mapping - 2-Dimensional map created in rviz software using hector mapping as the robot navigates to the destination.

Figure: 24: Wrench Detection - Robot is positioned to detect the smallest wrench on the board.

Figure: 25: Wrench detection - The above picture shows the wrench detection package run on the client computer. The smallest wrench is shown with the largest number on the left window. Therefore, we can conclude that the smallest wrench is the second wrench from the right in the above case.

```
the smallest wrench of all the 5 wrenches is the number 4 wrench, the position i
s @(48,25,556), defining the center of the kinect is (0,0,0) position, and left
is the positive x direction, up is the positive y direction, forward is the posi
tive z direction
```

Figure : 26: Local Coordinate Determination – The local coordinates of the wrench are shown in the terminal when the wrench detection package is run. The three values correspond to x, y, z distance of the wrench in mm from the camera. In the above case (x, y, z) = (48 mm, 25 mm, 556 mm).

| | Measured value/ Program value | x-coordinate (mm) | y-coordinate (mm) | z- coordinate (mm) | % error in x | % error in y | % error in z |
|---|---|---|---|---|---|---|---|
| Trial 1 | Actual value | 15 | 20 | 495 | 13.33 | 15 | 1.41 |
| | Program value | 17 | 23 | 502 | | | |
| Trial 2 | Actual value | 30 | 25 | 530 | 40 | 24 | 1.51 |
| | Program value | 42 | 31 | 538 | | | |
| Trial 3 | Actual value | 20 | 20 | 620 | 20 | 40 | 1.94 |
| | Program value | 24 | 28 | 632 | | | |

Table: 3

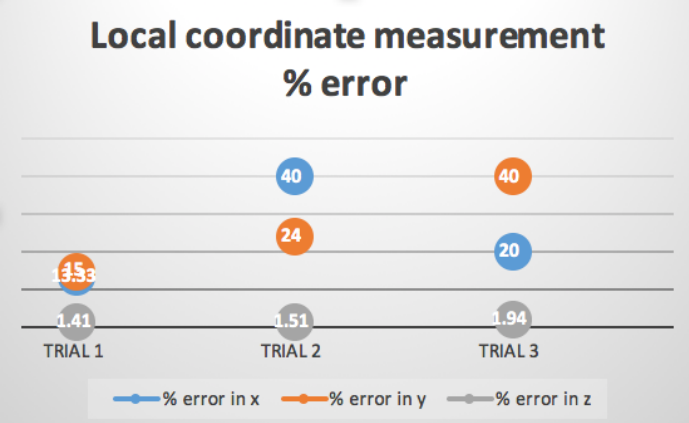



Figure: 27

## Conclusions and Future Work

A tele-operational robot for object detection has been successfully developed. The robot was successfully controlled wirelessly with the help of the Logitech wireless gaming controller by the Master computer from the Server Room using the ROS master slave and networking package. A 2D map of the travelled area was successfully created which also showed the distance travelled by the robot and also has a global coordinate system which told the position of the robot. The Asus Xiton was used to obtain the RGB and the depth image and was displayed in master computer using the rviz-ROS package. The same camera was used to successfully to detect the smallest wrench from a set of hanging wrenches. Using the code made by the team we were able to tell the local wrench coordinates with respect to the camera on the UGV.

There is scope for betterment in different aspects of the robot such as Using a faster internet and Wi-Fi with a larger range. Increasing the speed of GeoTIFF creation for mapping of the environment. There is also scope for adding a manipulator to the UGV.

We have finish the tele-operation and target (wrench) detection in this project. But in order to get fully autonomous operation, future work is needed to autonomous navigation by mapping or visual input. Possible solution including obstacle detection and avoidance by visual input, or use the 2D map we created. In order to get wrench operation, robotic hand is needed to actually manipulate the wrench. So we need to build shared coordinate to achieve autonomous manipulation.

## References


[1] Michele Nash-Hoff, 2015, "What Is the Importance of Unmanned Vehicles to Our Economy? <"http://www.industryweek.com/emerging-technologies/what-importance unmanned vehicles-our-economy>

[2] 2015, “The UAV – The Future Of The Sky.”< http://www.theuav.com>

[3] 2015, “Google Self-Driving Car Project.” Google Self-Driving Car Project, <https://www.google.com/selfdrivingcar/>

[4] Ghose, By. "Self-Driving Cars: 5 Problems That Need Solutions." *LiveScience*. TechMedia Network, 14 May 2015. Web. 16 Dec. 2015. <http://www.livescience.com/50841-future-of-driverless-cars.html>

[5] "6 Simple Things Google's Self-Driving Car Still Can't Handle." *Gizmodo*. Web. 16 Dec. 2015.<http://gizmodo.com/6-simple-things-googles-self-driving-car-still-cant-han-1628040470>

[6] "Wiki." *Hector_mapping*. Web. 16 Dec. 2015. <http://wiki.ros.org/hector_mapping>

[7] "Wiki." *Sicktoolbox_wrapper*. Web. 16 Dec. 2015. <http://wiki.ros.org/sicktoolbox_wrapper>

[8] "Wiki." *Sicktoolbox*. Web. 16 Dec. 2015. <http://wiki.ros.org/sicktoolbox>

[9] "Wiki." *Rtabmap_ros*. Web. 16 Dec. 2015. <http://wiki.ros.org/rtabmap_ros>

[10] "Wiki." *Rtabmap_ros/Tutorials/HandHeldMapping*. Web. 16 Dec. 2015. <http://wiki.ros.org/rtabmap_ros/Tutorials/HandHeldMapping>

[11] "Wiki." *Openni2_launch*. Web. 16 Dec. 2015. <http://wiki.ros.org/openni2_launch>

[12] "Product Description." *Sabertooth 2X25 V2 Regenerative Dual Motor Driver*. Web. 16 Dec. 2015. <https://www.dimensionengineering.com/products/sabertooth2x25>

[13] "Simulating Arduino Mega2560 in Proteus. - Dms." Dms. 5 Jan. 2015. Web. 16 Dec. 2015. <http://dostmuhammad.com/simulating-arduino-mega2560-in-proteus/>

[14] "Safety Laser Scanners." *SICK – Safety Laser Scanners – Overview*. Web. 16 Dec. 2015. <http://www.sick.com/group/EN/home/products/product_portfolio/optoelectronic_protective_devices/Pages/safetylaserscanners.aspx>

[15] "Wiki." *ROS/NetworkSetup*. Web. 16 Dec. 2015. <http://wiki.ros.org/ROS/NetworkSetup>

## Appendix

1) “Team 1 - Master Computer Screen”: This video shows the screen of the master computer during the competition.
2) “Team 1 - Unmanned Ground Vehicle”: This video shows the Unmanned Ground Vehicle as it travels to reach the Wrenches (using tele-operation) for object detection.