NYU | TANDON SCHOOL OF ENGINEERING

Student Name	Net ID	N Number
Naveen Kunam	nk3090	N19068326
Geeta Chandra Raju, Bethala	gb2643	N16316570
Raichuri Vishnu Sai Rayalu	rr4302	N14436451

Term Project - Independent Problem

Teleoperation of a Robot in Gazebo with Image Feedback

Github link
https://github.com/bethalageetachandraraju/Teleop_Robot

Video Link
https://youtu.be/866OW6hLmJY

TABLE OF CONTENTS

INTRODUCTION 4

Problem Statement 4

Setup Summary 4

SETUP DESCRIPTION 5

Connection step with NYU VPN 5

GAZEBO WORLD CREATION 7

ROBOT MODELING 9

Joints and Links 12

Converting Robot model to URDF 13

ROS NAVIGATION AND SLAM 15

NETWORK EXPERIMENTS 22

CONCLUSION 26

FUTURE IMPROVEMENTS 27

GITHUB & VIDEO 27

REFERENCE RESEARCH PAPER 27

Introduction

We are a group of 3 students, Naveen Kumar, Geeta Chandra Raju Bethala and Raichuri Vishnu Sai Rayulu. We have chosen to solve an independent problem as our final term project of the course Simulation Tools for Mechatronics and Robotics.

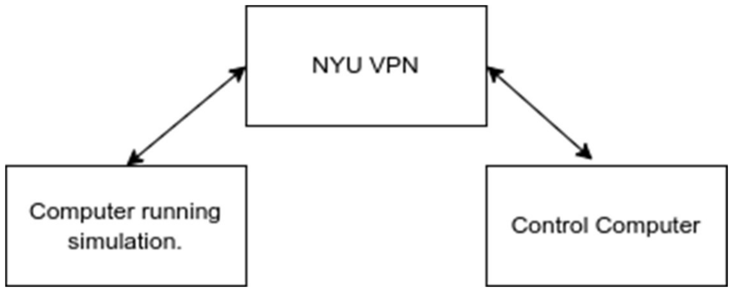
Problem Statement

The project aims to design and control a robot simulation in an environment using cloud telepresence applications. The robot should be able to travel autonomously, avoid obstacles, and transmit video visuals to the end user. The robot design will be converted into a URDF file, which will be used to build a movable robot model in the Gazebo simulation environment. The robot will be tested in an environment with different types of obstacles, such as a hospital, lab, or educational institute. It should be able to detect and avoid these obstacles while completing tasks. We want to address and understand how we can control a robot at any different locations using the internet and get a video feed from the camera mounted on the mobile robot over the internet from anywhere in the world.

Setup Summary

Here, we have three computers. One computer is running a Gazebo simulation. The Second computer is working as a control computer which has RViz, Teleop Control, and Image View node. RViz helps in ROS navigation, Teleop Control helps for manual control and Image View is the node which is subscribing image (compressed image) and showing on control computer. The third computer in NYU VPN. We connect the computer running simulation and the computer which controls the robot in simulation to NYU VPN. By doing so we bring the control computer and simulation computer on the same network, and by using ROS and setting up ROS Environment Variables we establish communication between both of the computers and are able to do tasks of interest easily through the control computer.

Setup Description



We have two computers running Ubuntu 20.04 having ROS Noetic installed with Gazebo 11.12 and RViz. Both computers are connected to NYU VPN to bring them on the same network.

Connection step with NYU VPN

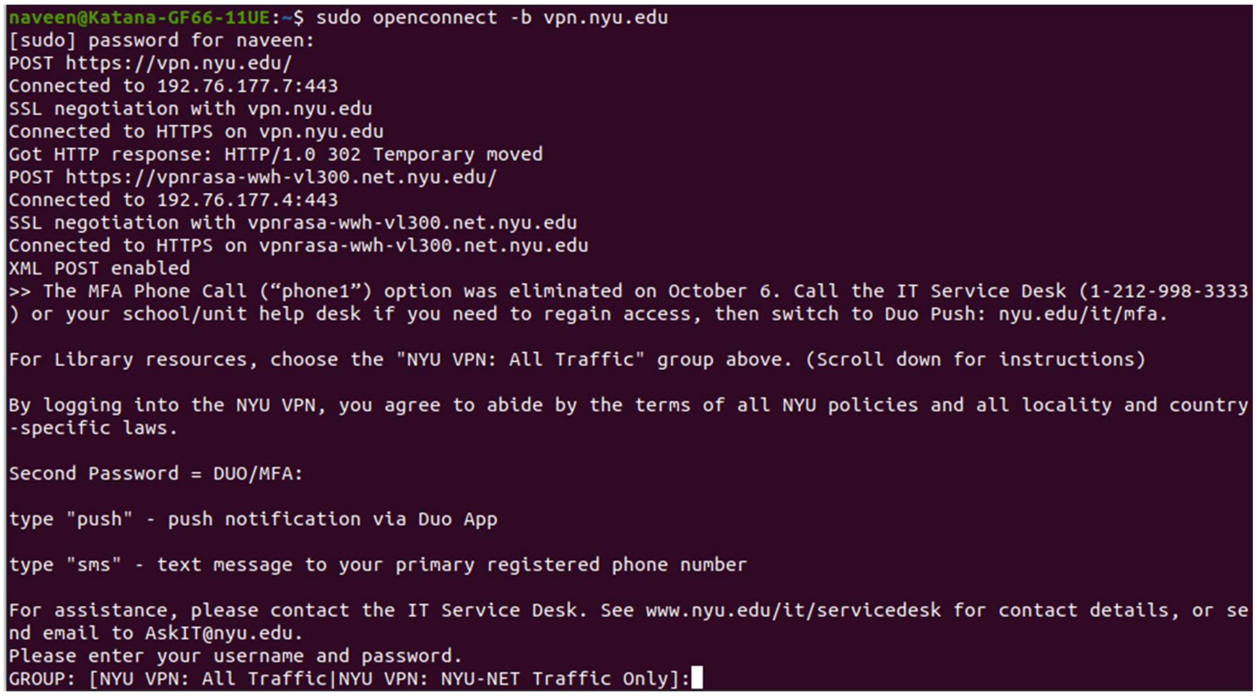
First, we need to install openconnect in our Linux system if it's not already installed. By copying and pasting the following command.

```
sudo apt install openconnect
```

After its installation, we need to connect to vpn.nyu.edu

```
sudo openconnect -b vpn.nyu.edu
```

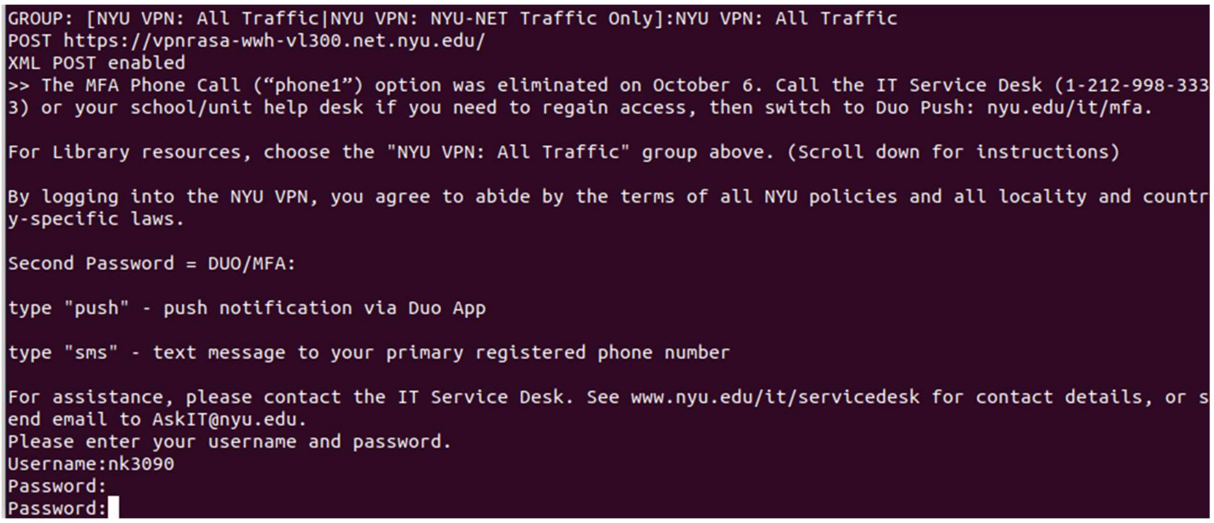
We got the following output after typing the above command



```
naveen@Katana-GF66-11UE:~$ sudo openconnect -b vpn.nyu.edu
[sudo] password for naveen:
POST https://vpn.nyu.edu/
Connected to 192.76.177.7:443
SSL negotiation with vpn.nyu.edu
Connected to HTTPS on vpn.nyu.edu
Got HTTP response: HTTP/1.0 302 Temporary moved
POST https://vpnrasa-wwh-vl300.net.nyu.edu/
Connected to 192.76.177.4:443
SSL negotiation with vpnrasa-wwh-vl300.net.nyu.edu
Connected to HTTPS on vpnrasa-wwh-vl300.net.nyu.edu
XML POST enabled
>> The MFA Phone Call (“phone1”) option was eliminated on October 6. Call the IT Service Desk (1-212-998-3333
) or your school/unit help desk if you need to regain access, then switch to Duo Push: nyu.edu/it/mfa.

For Library resources, choose the "NYU VPN: All Traffic" group above. (Scroll down for instructions)

By logging into the NYU VPN, you agree to abide by the terms of all NYU policies and all locality and country
-specific laws.

Second Password = DUO/MFA:

type "push" - push notification via Duo App

type "sms" - text message to your primary registered phone number

For assistance, please contact the IT Service Desk. See www.nyu.edu/it/servicedesk for contact details, or se
nd email to AskIT@nyu.edu.
Please enter your username and password.
GROUP: [NYU VPN: All Traffic|NYU VPN: NYU-NET Traffic Only]:
```

We need to choose “NYU VPN: All Traffic” and type this as a reply to the prompt.

After that it will ask our username i.e. our Net ID of NYU and Password and as a second password we need to put “push” so we get a notification on our phone and then we allow our computer to connect to NYU VPN using our credentials.



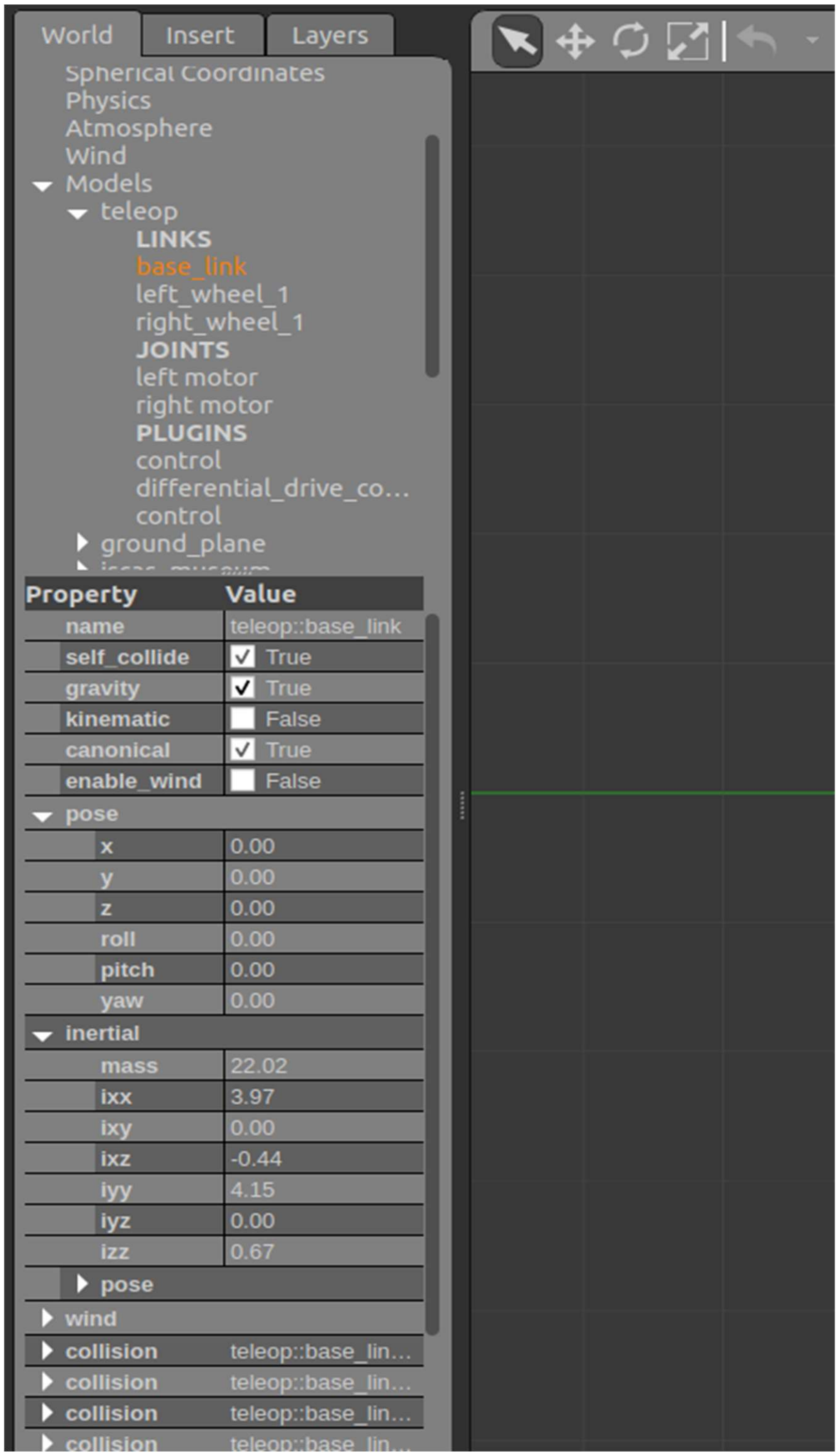
```
GROUP: [NYU VPN: All Traffic|NYU VPN: NYU-NET Traffic Only]:NYU VPN: All Traffic
POST https://vpnrasa-wwh-vl300.net.nyu.edu/
XML POST enabled
>> The MFA Phone Call (“phone1”) option was eliminated on October 6. Call the IT Service Desk (1-212-998-333
3) or your school/unit help desk if you need to regain access, then switch to Duo Push: nyu.edu/it/mfa.

For Library resources, choose the "NYU VPN: All Traffic" group above. (Scroll down for instructions)

By logging into the NYU VPN, you agree to abide by the terms of all NYU policies and all locality and countr
y-specific laws.

Second Password = DUO/MFA:

type "push" - push notification via Duo App

type "sms" - text message to your primary registered phone number

For assistance, please contact the IT Service Desk. See www.nyu.edu/it/servicedesk for contact details, or s
end email to AskIT@nyu.edu.
Please enter your username and password.
Username:nk3090
Password:
Password:
```

Gazebo World Creation

Gazebo is a robotics simulation software that allows users to design and simulate environments, robots, and other objects. In Gazebo, users can create custom worlds by designing and placing objects, adding terrain and other features, and specifying the physical properties of the objects and the environment.

In the Gazebo world, the model and its different parts contain three properties, collision, inertial and visual.
In order to see the property of a particular element we need to browse toward it from the left side and choose appropriate properties to see its data.
These values were given to it while we imported this robot from Fusion 360.

In Gazebo, we need to ensure that the geometric property that we are giving to visual should be same with collision and inertial properties of the model else model will not show physical properties like if it collides with something or anything.



Property	Value
name	teleop::base_link
self_collide	✓ True
gravity	✓ True
kinematic	False
canonical	✓ True
enable_wind	False
pose	
x	0.00
y	0.00
z	0.00
roll	0.00
pitch	0.00
yaw	0.00
inertial	
mass	22.02
ixx	3.97
ixy	0.00
ixz	-0.44
iyy	4.15
iyz	0.00
izz	0.67
pose	
wind	
collision	teleop::base_lin…
collision	teleop::base_lin…
collision	teleop::base_lin…
collision	teleop::base_lin…

Fig:- Properties of the model

To launch our custom world in Gazebo, use the roslaunch command:

```
roslaunch <package_name> <launch_file>.launch
```

Our world file is ISCAS Museum(China Software Technology Museum)
This lab closely resembles the real life environment where our robot will be used.

Robot Modeling

We designed our robot using Fusion-360

We took reference from various medical and Industrial assistive robots available in market, and came up with this design.

Overview of the procedure:

- We used the modeling tools in Fusion 360 to design the body and base of my telepresence robot. We used the Extrude, Revolve, Sweep, and other tools to create the desired shape and features of the robot. We used sketches, 3D shapes, and other modeling elements to create the structure of the robot.

- We kept in mind the requirements for an URDF file while designing the robot. The URDF file format is used to describe the kinematic structure of a robot, including its joints, links, and other mechanical elements. Therefore, we included these elements in my robot design in a way that is clear and easy to understand.

- We also consider how my robot would be controlled and powered. We included motors, servos, sensors, and other components in our design to enable telepresence functionality. We also took into account the size and weight of the robot, as well as any other constraints or requirements that may affect its performance or functionality.

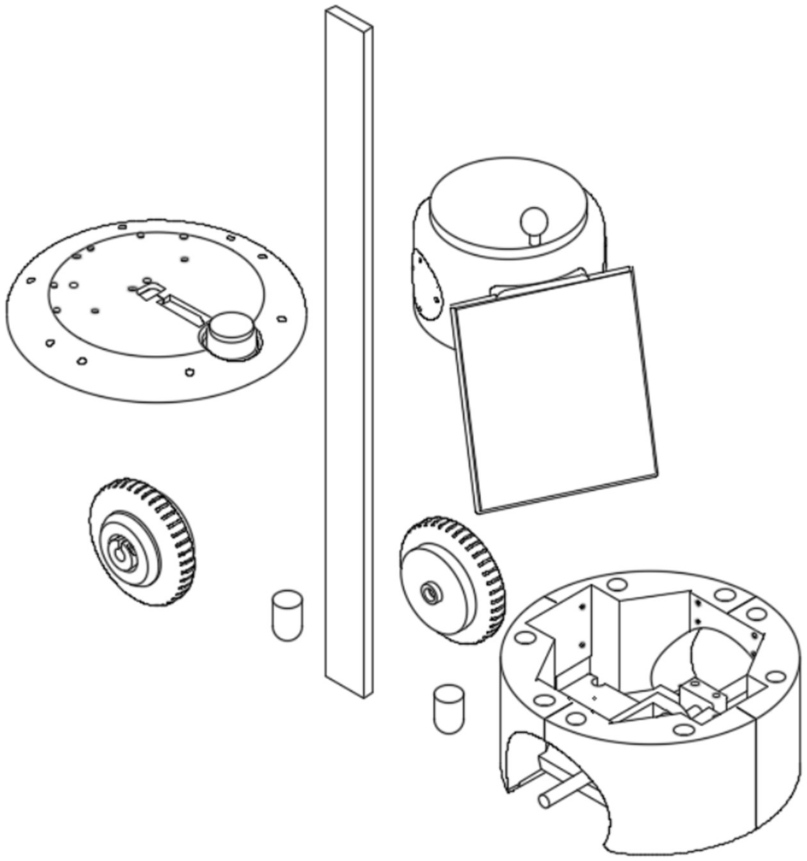


Fig :- Robot parts

We added sensors and modules to our design so that it will be able to access the plugin in ROS:

- Differential Motor
- Rotary to linear motion converter
- Lidar
- Camera
- LCD Display
- Microphones

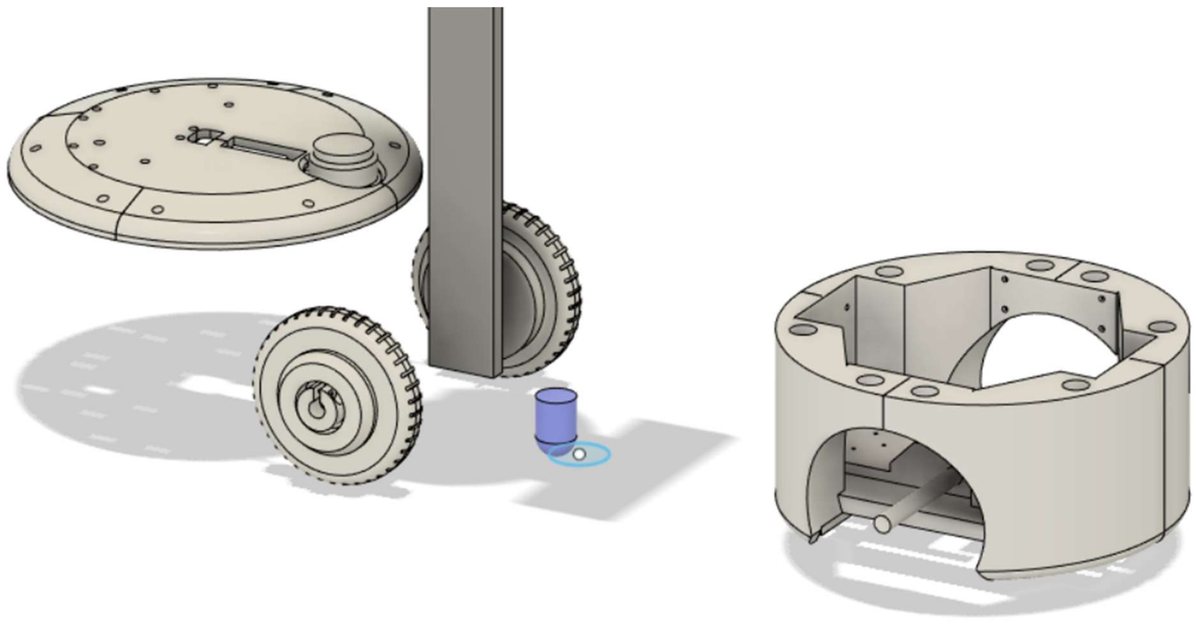


Fig :- Individual parts in Fusion 360

Base of the robot plays a crucial role as it consists motors, lidar, controllers and SBC(single board computer)

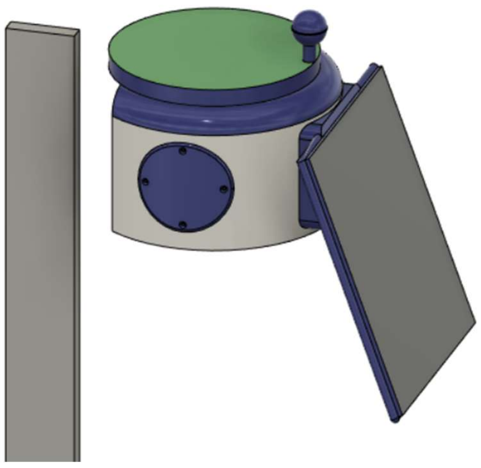


Fig :- Robot Head

The head part consists of a display, microphone and camera. This part is connected to the vertical slider in such a way that it can linearly adjust its height. And the camera is able to pan its view.

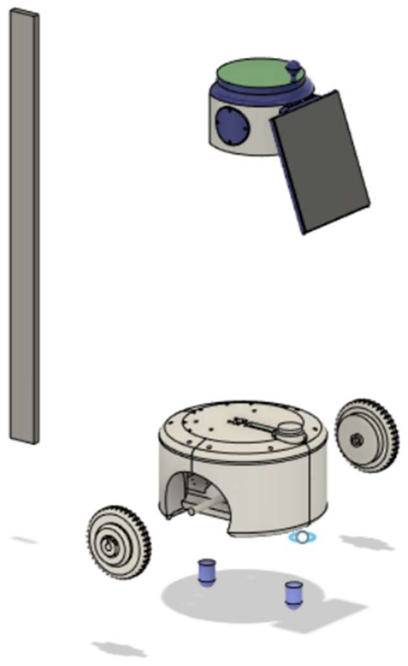


Fig :- Exploded view

This is the exploded view of our robot.

Joints and Links

We used a rotary joint for the wheels to motor, and lidar to its base.

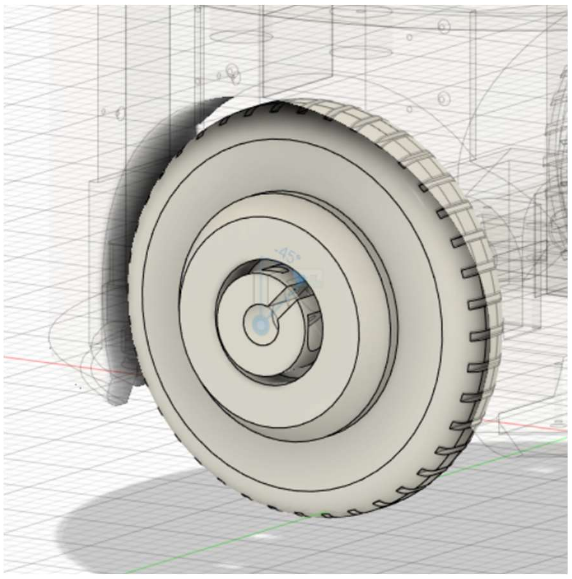


Fig :- Wheel Rotary joint

And Slider joint between head and vertical support bar.

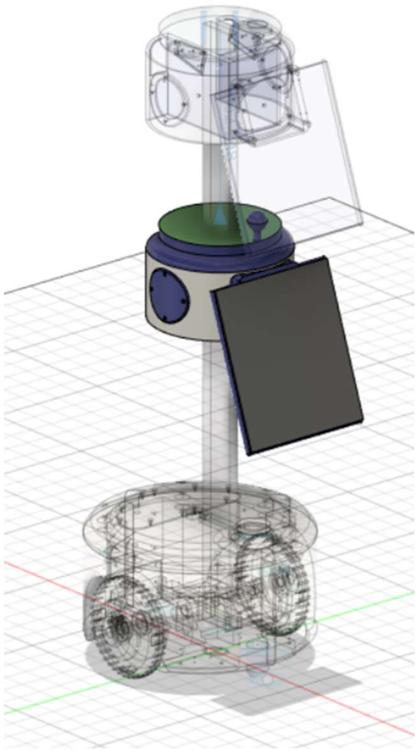


Fig :- Slider Joint for head

This link will give access to view our robot model:

https://a360.co/3GceJDz

Converting Robot model to URDF

- First we downloaded the open source repository for format conversion
- In windows command prompt:

```
cd <path to fusion2urdf>
Copy-Item ".\URDF_Exporter\" -Destination "${env:APPDATA}\Autodesk\Autodesk Fusion 360\API\Scripts\" -Recurse
```

- After running the following command the converter will be added to the Autocad Fusion
- Before running the script we need to define the base_link, which will be used as reference to generate the remaining model and joints.
- We went to the ADD-INS tab in Fusion 360 and selected the "fusion2urdf" script.
- We ran the script and waited a few seconds (or minutes) for a folder dialog to appear. we chose a location to save the URDF file (e.g. "Desktop/test").
- we defined the base component and renamed it as "base_link".
- We ran the script again, choosing the folder to save.
- The URDF file and .stl files were saved in the designated folder (e.g. "Desktop/test").
- we moved the folder containing the exported files (e.g. "Desktop/test") to our ROS environment.
- In our ROS environment, we placed the generated _description package directory in our own ROS workspace (e.g. "catkin_ws").
- We ran

```
Catkin_make
source devel/setup.bash
```

- To view our robot in RViz, we ran the command.

```
roslaunch teleop_description display.launch
```

- To simulate our robot in Gazebo, we ran the command.

```
roslaunch teleop_description gazebo.launch
```

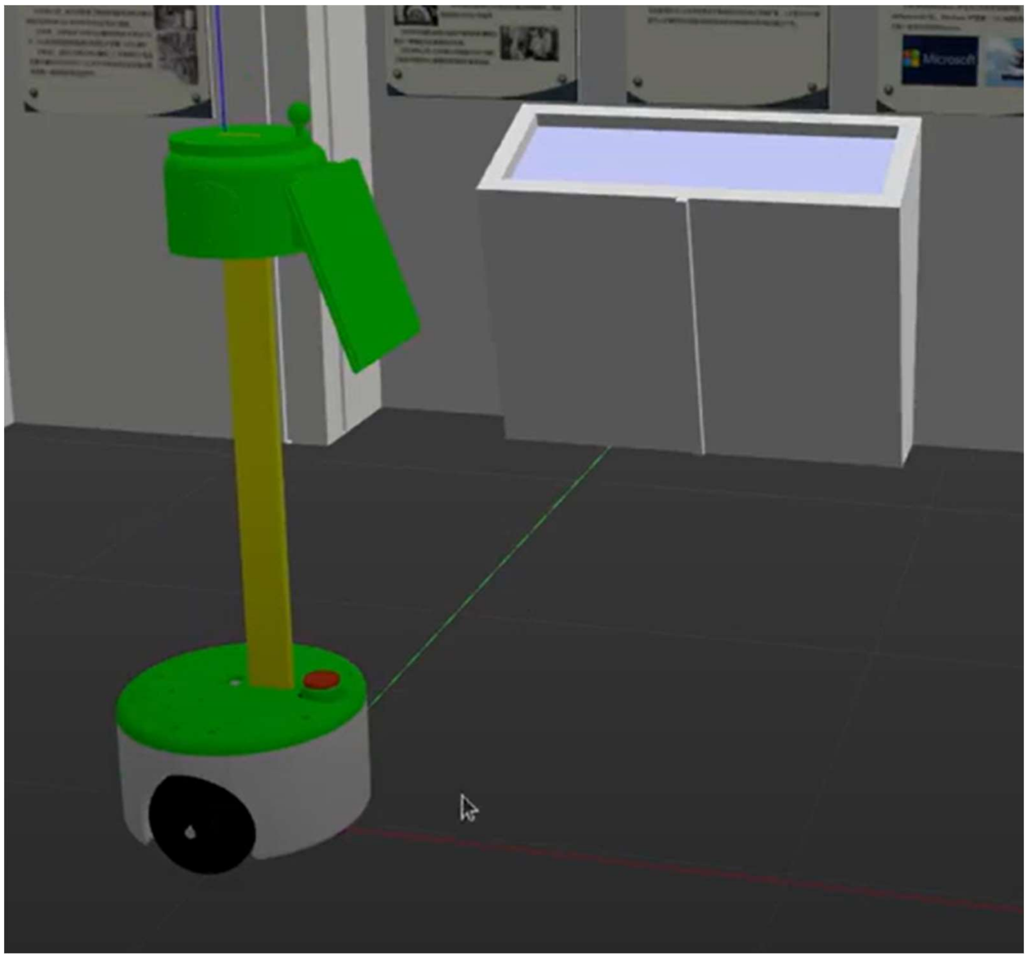


Fig :- Robot in gazebo

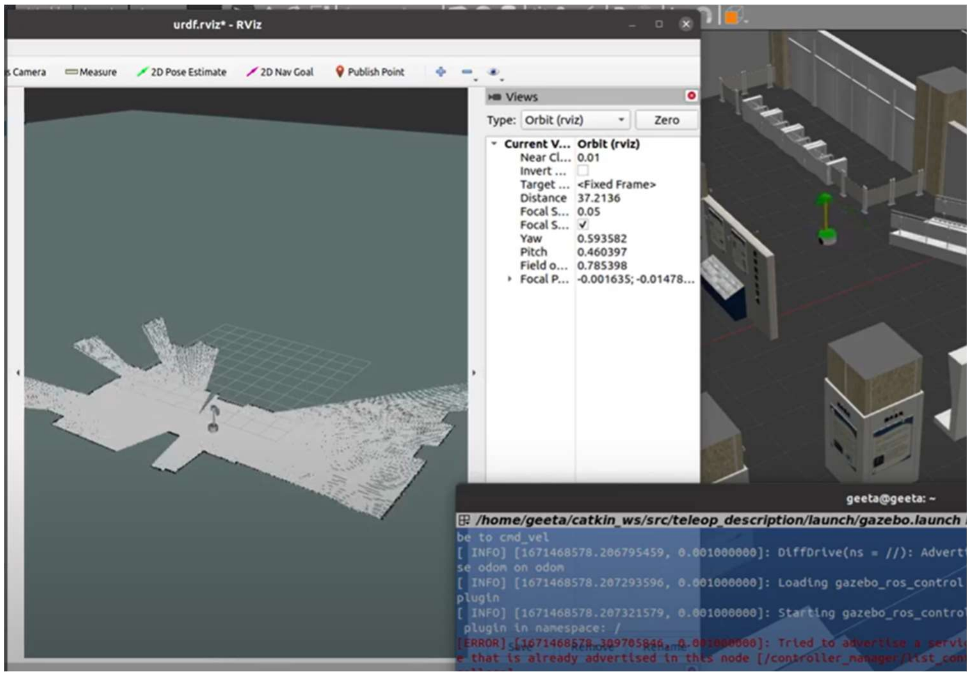
ROS Navigation and SLAM

To implement simultaneous localization and mapping (SLAM) with a lidar sensor using the Robot Operating System (ROS), we followed these steps:

- Setted up our ROS environment:
- Created a workspace and created a ROS package for our project.
- Connected our LiDAR sensor:
- Configured the LiDAR sensor in ROS:
- Created a ROS node that publishes the LiDAR data as a stream of point clouds.
- Set up the necessary launch files and parameters to configure the sensor's frame of reference, frequency of data acquisition, and other relevant settings.
- Created a map of the environment:

- Used the LiDAR data to build a map of the environment. Here we used the GMapping algorithm to generate map.
- The map is typically represented as an occupancy grid, where each cell in the grid represents the probability that the corresponding location in the environment is occupied by an obstacle.
- Localized the robot in the map:
- Used the LiDAR data and the map to estimate the pose (position and orientation) of the robot as it moves through the environment. This process is known as localization.
- There are several algorithms that can be used for localization, such as Adaptive Monte Carlo localization (AMCL) or robot_localization.
- Planned and executed a path for the robot:
- Used a path planning algorithm, such as the ROS Navigation Stack, to generate a safe and efficient path for the robot to follow based on the map and the robot's current pose.
- Executed the path using a robot controller, such as the ROS controller manager.
- Added a few dynamically moving obstacles and checked how our robot is tackling the situation and generating the alternate path.

Fig :- world



Current V...	Orbit (rviz)
Near Cl...	0.01
Invert ...	
Target ...	<Fixed Frame>
Distance	37.2136
Focal S...	0.05
Focal S...	✓
Yaw	0.593582
Pitch	0.460397
Field o...	0.785398
Focal P...	-0.001635; -0.01478...

```
geeta@geeta: ~
/home/geeta/catkin_ws/src/teleop_description/launch/gazebo.launch
be to cmd_vel
[ INFO] [1671468578.206795459, 0.001000000]: DiffDrive(ns = //): Advert
se odom on odom
[ INFO] [1671468578.207293596, 0.001000000]: Loading gazebo_ros_control
plugin
[ INFO] [1671468578.207321579, 0.001000000]: Starting gazebo_ros_contro
 plugin in namespace: /
[ERROR] [1671468578.309705846, 0.001000000]: Tried to advertise a servic
e that is already advertised in this node [/controller_manager/list_con
```

Fig :-Generating map

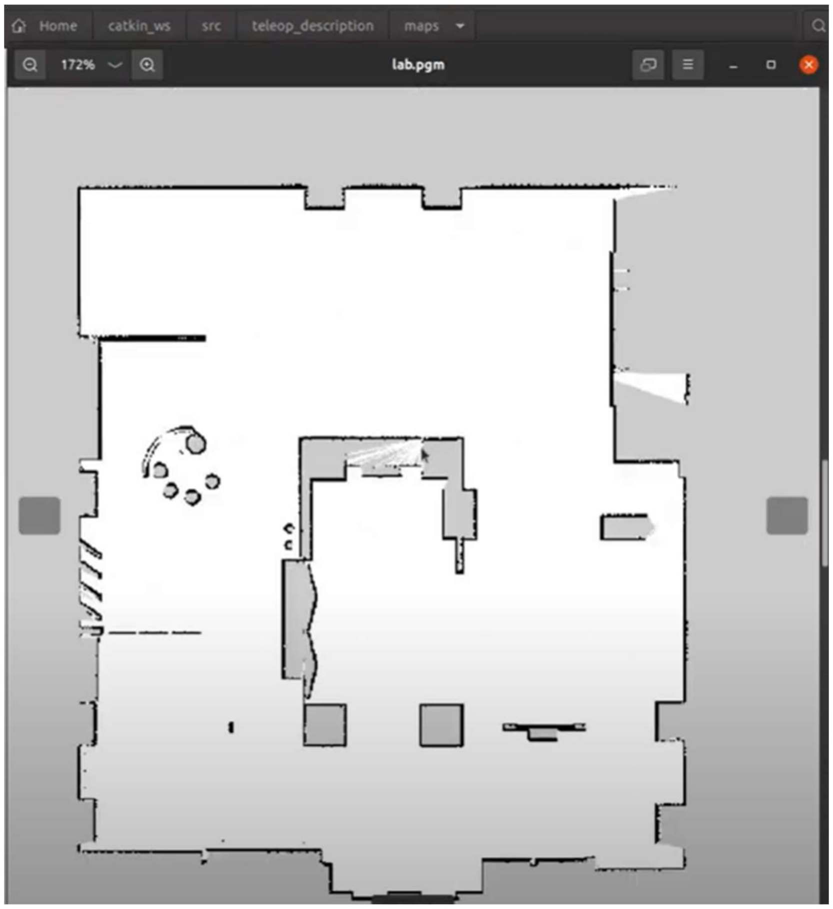
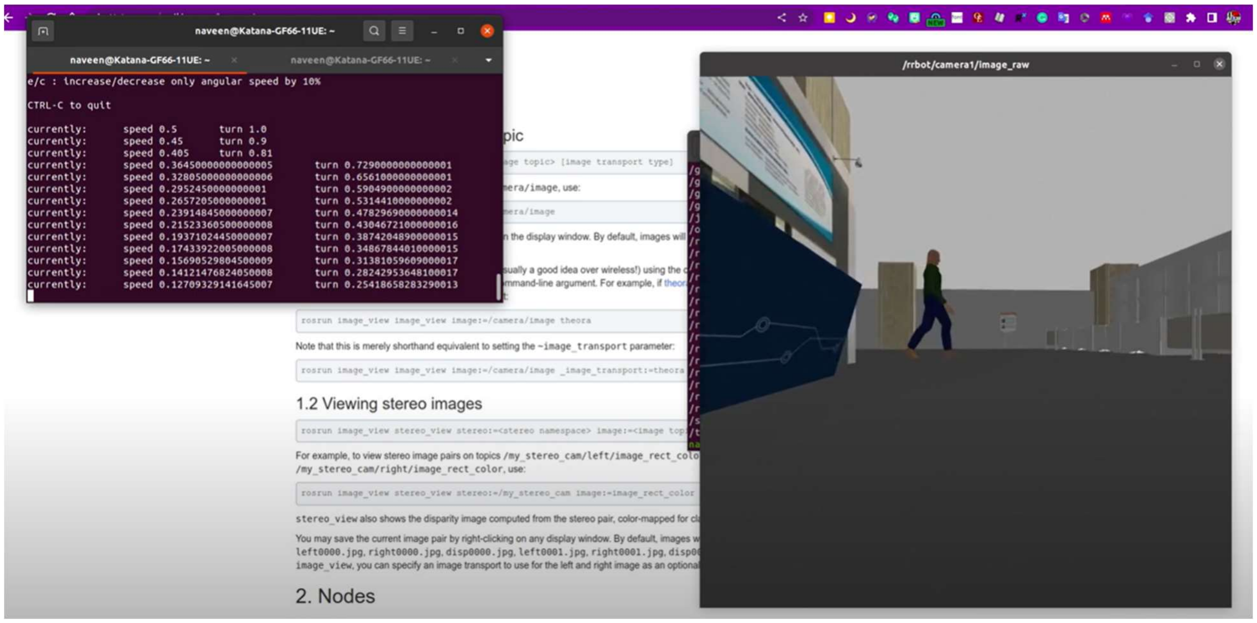


Fig :- Binary Occupancy Grid



```
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
currently:      speed 0.45      turn 0.9
currently:      speed 0.405     turn 0.81
currently:      speed 0.36450000000000005       turn 0.7290000000000001
currently:      speed 0.32805000000000006       turn 0.6561000000000001
currently:      speed 0.2952450000000001        turn 0.5904900000000002
currently:      speed 0.2657205000000001        turn 0.5314410000000002
currently:      speed 0.23914845000000007       turn 0.47829690000000014
currently:      speed 0.21523360500000008       turn 0.43046721000000016
currently:      speed 0.19371024450000007       turn 0.38742048900000015
currently:      speed 0.17433922005000008       turn 0.34867844010000015
currently:      speed 0.15690529804500009       turn 0.31381059609000017
currently:      speed 0.14121476824050008       turn 0.28242953648100017
currently:      speed 0.12709329141645007       turn 0.25418658283290013
```

pic

```
age topic> [image transport type]
```

era/image, use:

```
era/image
```

n the display window. By default, images will

sually a good idea over wireless!) using the
mmand-line argument. For example, if theora

```
rosrun image_view image_view image:=/camera/image theora
```

Note that this is merely shorthand equivalent to setting the ~image_transport parameter:

```
rosrun image_view image_view image:=/camera/image _image_transport:=theora
```

1.2 Viewing stereo images

```
rosrun image_view stereo_view stereo:=<stereo namespace> image:=<image top
```

For example, to view stereo image pairs on topics /my_stereo_cam/left/image_rect_colo
/my_stereo_cam/right/image_rect_color, use:

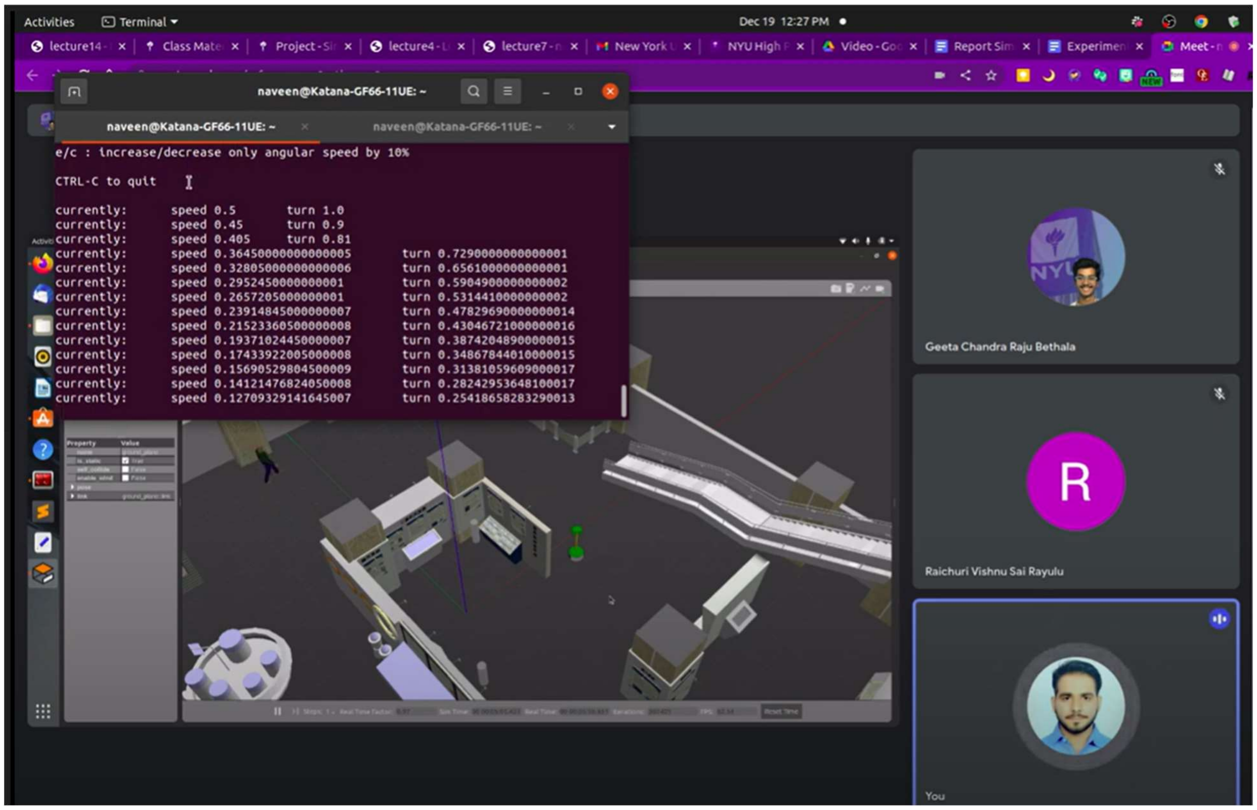
```
rosrun image_view stereo_view stereo:=/my_stereo_cam image:=image_rect_color
```

stereo_view also shows the disparity image computed from the stereo pair, color-mapped for cl

You may save the current image pair by right-clicking on any display window. By default, images w
left0000.jpg, right0000.jpg, disp0000.jpg, left0001.jpg, right0001.jpg, disp00
image_view, you can specify an image transport to use for the left and right image as an optiona

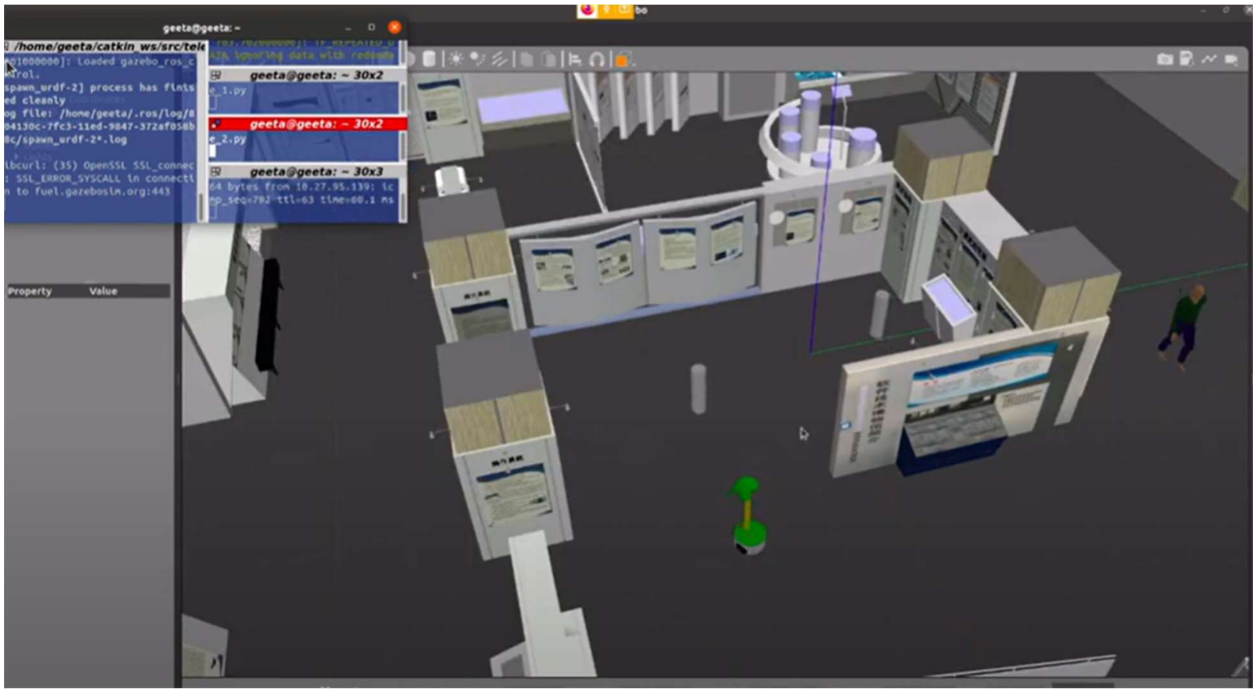
2. Nodes

Fig :-Camera Feed



```
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
currently:      speed 0.45      turn 0.9
currently:      speed 0.405     turn 0.81
currently:      speed 0.36450000000000005       turn 0.7290000000000001
currently:      speed 0.32805000000000006       turn 0.6561000000000001
currently:      speed 0.2952450000000001        turn 0.5904900000000002
currently:      speed 0.2657205000000001        turn 0.5314410000000002
currently:      speed 0.23914845000000007       turn 0.47829690000000014
currently:      speed 0.21523360500000008       turn 0.43046721000000016
currently:      speed 0.19371024450000007       turn 0.38742048900000015
currently:      speed 0.17433922005000008       turn 0.34867844010000015
currently:      speed 0.15690529804500009       turn 0.31381059609000017
currently:      speed 0.14121476824050008       turn 0.28242953648100017
currently:      speed 0.12709329141645007       turn 0.25418658283290013
```

Fig :-Tele-operating



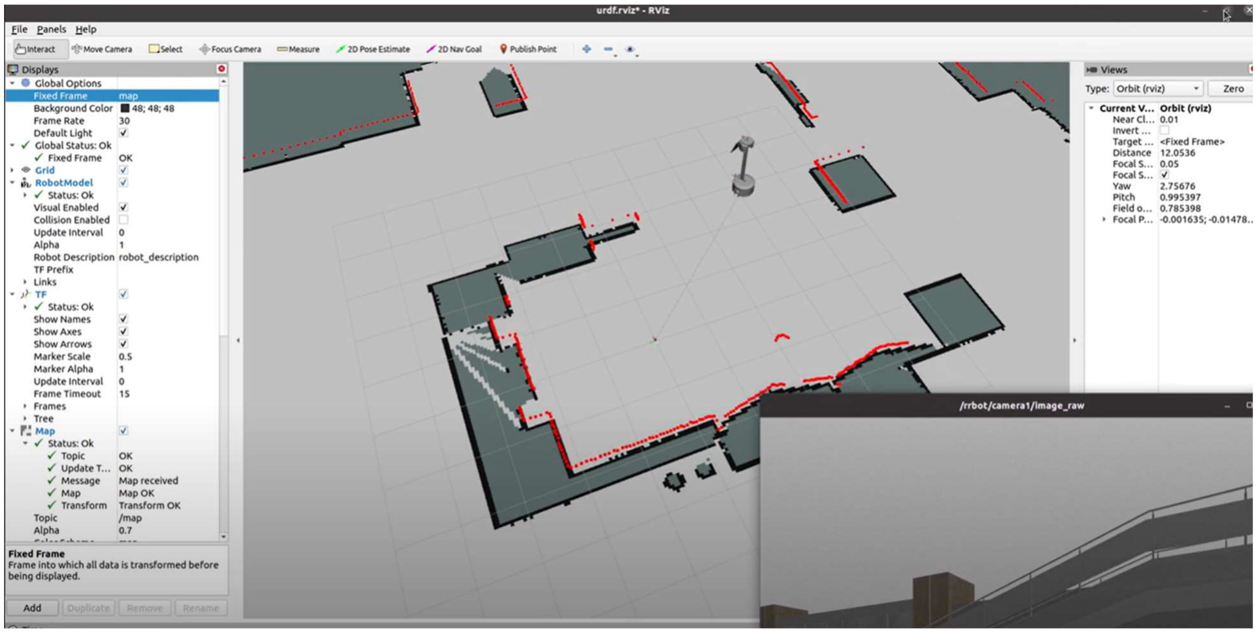
```
/home/geeta/catkin_ws/src/tele
01000000]: Loaded gazebo_ros_c
trol.
spawn_urdf-2] process has finis
ed cleanly
og file: /home/geeta/.ros/log/8
04130c-7fc3-11ed-9847-372af058b
8c/spawn_urdf-2*.log

ibcurl: (35) OpenSSL SSL_connec
: SSL_ERROR_SYSCALL in connecti
n to fuel.gazebosim.org:443
```

```
geeta@geeta: ~ 30x2
e_1.py
```

```
geeta@geeta: ~ 30x2
e_2.py
```

```
geeta@geeta: ~ 30x3
64 bytes from 10.27.95.139: ic
mp_seq=792 ttl=63 time=60.1 ms
```

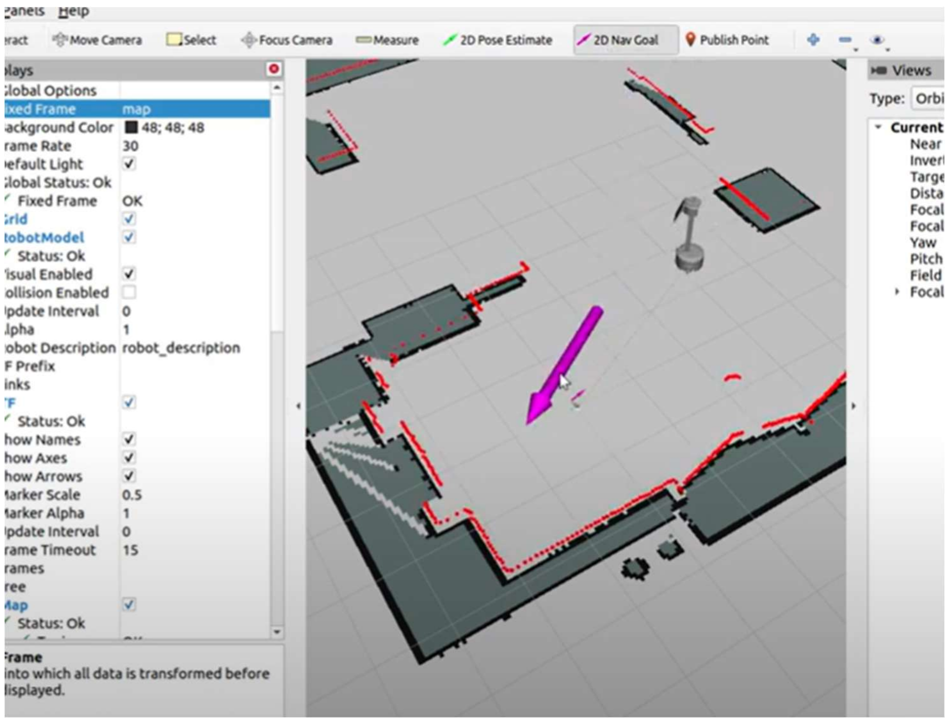
Fig :-Dynamic Obsticals



```
Global Options
  Fixed Frame           map
  Background Color      48; 48; 48
  Frame Rate            30
  Default Light         ✓
✓ Global Status: Ok
  ✓ Fixed Frame         OK
Grid                    ✓
RobotModel              ✓
  ✓ Status: Ok
  Visual Enabled        ✓
  Collision Enabled     ☐
  Update Interval       0
  Alpha                 1
  Robot Description     robot_description
  TF Prefix
  Links
TF                      ✓
  ✓ Status: Ok
  Show Names            ✓
  Show Axes             ✓
  Show Arrows           ✓
  Marker Scale          0.5
  Marker Alpha          1
  Update Interval       0
  Frame Timeout         15
  Frames
  Tree
Map                     ✓
  ✓ Status: Ok
    ✓ Topic             OK
    ✓ Update T...       OK
    ✓ Message           Map received
    ✓ Map               Map OK
    ✓ Transform         Transform OK
  Topic                 /map
  Alpha                 0.7

Fixed Frame
Frame into which all data is transformed before
being displayed.
```

```
Type: Orbit (rviz)
Current V...   Orbit (rviz)
  Near Cl...   0.01
  Invert ...   ☐
  Target ...   <Fixed Frame>
  Distance     12.0536
  Focal S...   0.05
  Focal S...   ✓
  Yaw          2.75676
  Pitch        0.995397
  Field o...   0.785398
  Focal P...   -0.001635; -0.01478..
```

Fig :-Lidar Scan



lays

lobal Options	
ixed Frame	map
ackground Color	■ 48; 48; 48
rame Rate	30
efault Light	✓
lobal Status: Ok	
✓ Fixed Frame	OK
rid	✓
obotModel	✓
✓ Status: Ok	
isual Enabled	✓
ollision Enabled	
pdate Interval	0
lpha	1
obot Description	robot_description
F Prefix	
inks	
F	✓
✓ Status: Ok	
how Names	✓
how Axes	✓
how Arrows	✓
arker Scale	0.5
arker Alpha	1
pdate Interval	0
rame Timeout	15
rames	
ree	
ap	✓
✓ Status: Ok	

rame
into which all data is transformed before
isplayed.

Fig :-Setting Navigation Goal

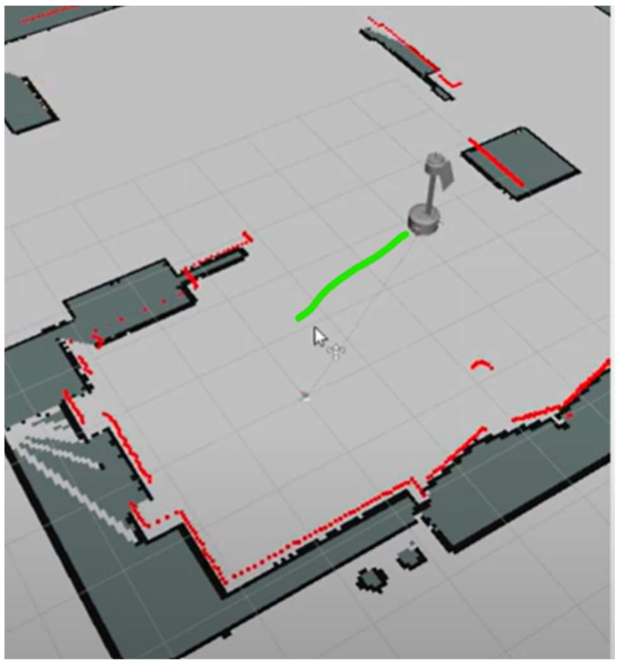


Fig :-Path generated to reach the given goal

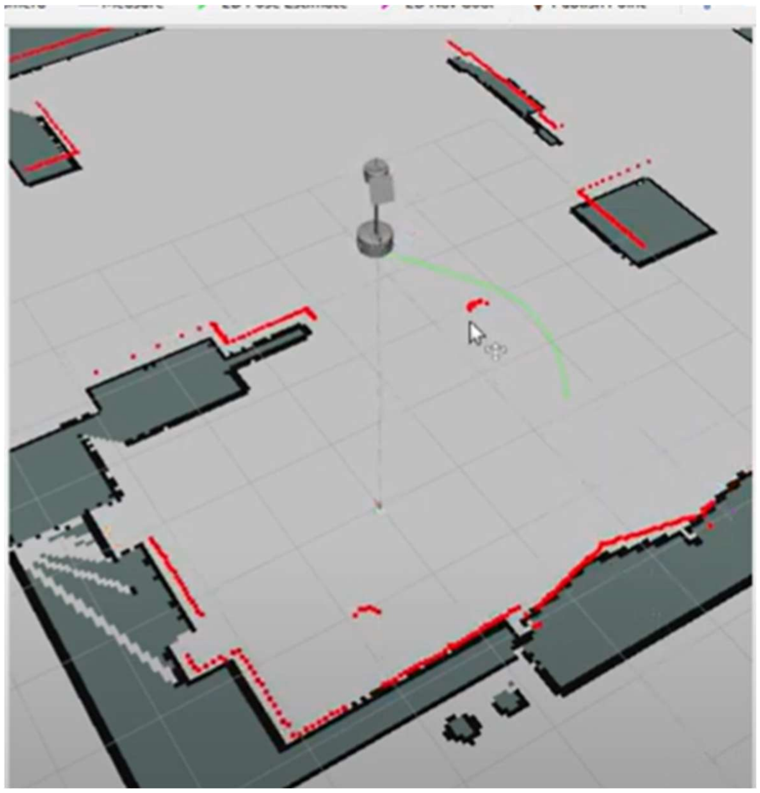


Fig :-Robot generated a new map to avoid dynamic obstacles

ROS Environment Variables
There are several ROS environment variable of which we used is
ROS_ MASTER_URI
ROS_MASTER_URI is a necessary setting that specifies the location of the master for nodes to connect to. It should be set to the XML-RPC URI of the master. It is important to be careful when using localhost, as it can cause unexpected issues with nodes launched remotely.

```
export ROS_MASTER_URI=http://mia:11311/
```

Light Detection and Ranging

LiDAR sensors are devices that use lasers to scan and create a 2D image of an environment. They do this by emitting a laser pulse and measuring the time it takes for the pulse to return, which allows them to calculate the distance to the reflection. One 360 degree LiDAR is mounted on the robot body. LiDAR scans are transmitted to a control center with a total of 720 scan points. The LiDAR data is used for positioning, emergency braking, and obstacle avoidance by the system and is also presented to the operator to aid in navigation. Research has shown that users are able to navigate more accurately and quickly using sensors that display the surrounding environment rather than a fixed camera on a robot. The LiDAR data will be displayed on a map.

Network Experiments

Latency
Latency in computer networking refers to the amount of time it takes for a data packet to travel from one point to another. The lower the latency, the faster the data can be transmitted. One way to measure latency is by calculating the round-trip time (RTT) of a data packet, which is the time it takes for the packet to reach its destination and return. A low RTT indicates a faster network connection with low latency.

Packet size
In computer networking, a packet is a unit of data that is transmitted over a network. It is similar to a physical package that is sent through the mail, as it includes a source and destination address, as well as the content being transferred. Packets are often used to transmit large amounts of data, as they can be divided into smaller units for easier transmission and then reassembled at the destination. When packets reach their intended destination, they are reassembled into a single file or block of data.

These two factors play a crucial role in teleoperation, hence keeping an eye over these factors and implementing methods to improve these factors will make our robot run smoothly.

These are the following experiments performed and tabulated the results

The below data came from running a ping command, which sends Internet Control Message Protocol (ICMP) echo request packets to a target host and measures the time it takes for the host to respond with an echo reply. The data includes the sequence number of the packet (icmp_seq), the time to live (ttl) of the packet, and the time it took for the host to respond in milliseconds (time).

The target host has an IP address of 10.27.95.139 and is responding to the ping requests consistently, with a relatively stable ttl value of 63. The response times vary somewhat, with some responses taking longer than others. This could be due to various factors such as network congestion, distance between the host and the sender, and the load on the host. Overall, it appears that the host is responding in a timely manner and that the network connection is functioning properly.

- This is the data from the master PC ie where the robot is present and when there is no ROS or gazebo running

icmp_seq	ttl	time (ms)
19	63	39.9
20	63	34.6
21	63	33.6
22	63	75.5
23	63	32.0
166	63	51.0
167	63	83.1
168	63	54.6
169	63	64.1

```
 232 packets transmitted, 230 received, 0.862069% packet loss, time 231327ms
rtt min/avg/max/mdev = 28.059/78.719/471.116/65.131 ms
```

When we use a computer, we can send messages to other computers using something called the Internet. Sometimes, we want to make sure that the other computer is working and can hear us, so we use a special command called "ping". When we use ping, we send a message called a "packet" to the other computer and wait for it to send the packet back to us. This helps us see how fast the other computer is responding and if it's working correctly.

In this case, we sent 232 packets to another computer and we received 230 of them back. That means that the other computer was able to hear us and respond most of the time. We also saw that only a small number of packets, or messages, didn't make it back to us. That's called "packet loss" and it happens sometimes when we use the Internet.

We also measured how long it took for the packets to go back and forth between the computers. This is called the "round-trip time" or "rtt". We saw that it took a minimum of 28.059 milliseconds, an average of 78.719 milliseconds, a maximum of 471.116 milliseconds, and had a standard deviation of 65.131 milliseconds. This tells us that the other computer was able to respond pretty quickly most of the time, but sometimes it took a little longer.

Overall, it looks like the other computer is working properly and can hear us when we send messages using the Internet.

- This is the data from the controller PC ie where the users controls the robot and when there is no ROS or gazebo running

icmp_seq	**ttl**	**time (ms)**
1	63	109
2	63	41.8
3	63	52.3
4	63	175
5	63	174
6	63	59.3
7	63	73.3
8	63	41.5
9	63	43.6

It looks like the target host has an IP address of 10.27.20.68 and is responding to the ping requests consistently, with a relatively stable ttl value of 63.It is worth noting that the response times for icmp_seq 4 and 5 are significantly higher than the others, which could indicate a temporary issue with the network or the host. Overall, it appears that the host is responding in a timely manner and that the network connection is functioning properly.

```
96 packets transmitted, 95 received, 1.04167% packet loss, time 95140ms
rtt min/avg/max/mdev = 30.478/92.777/463.384/77.036 ms
```

The data shows the number of packets transmitted (96), the number of packets received (95), and the percentage of packets lost (1.04167%). The data also includes the round-trip time (rtt) in milliseconds, which is the time it takes for a packet to be transmitted to the target host and for the host to respond. The rtt is measured in min/avg/max/mdev, which stands for the minimum, average, maximum, and standard deviation of the rtt values.

In this case, it appears that 95 out of 96 packets were received successfully, which means that the network connection is functioning properly. The packet loss rate is relatively low at 1.04167%. The rtt values also appear to be within an acceptable range, with a minimum of 30.478 ms, an average of 92.777 ms, a maximum of 463.384 ms, and a standard deviation of 77.036 ms. This suggests that the target host is responding in a timely manner and that there is relatively stable network performance.

- **Data when navigation with camera feed**

icmp_seq	**ttl**	**time (ms)**
273	63	130
274	63	90.8
275	63	106
276	63	361
277	63	166
278	63	219
279	63	120
280	63	92.7
281	63	143
282	63	236

It looks like the target host has an IP address of 10.27.95.139 and is responding to the ping requests consistently, with a stable ttl value of 63. The response times vary somewhat, with some responses taking longer than others. This could be due to various factors such as network congestion, distance between the host and the sender, and the load on the host. It is worth noting that the response time for icmp_seq 276 is significantly higher than the others, which could indicate a temporary issue with the network or the host. Overall, it appears that the host is responding in a timely manner and that the network connection is functioning properly.

Conclusion

1. By doing these experiments, we concluded that we are able to control the robot through the internet with a moderate latency which is not a big concern because we are making the robot move at walking speed.
2. We also learned the potential of ROS for modern-day simulation and ROS Navigation Stack to avoid both static and dynamic obstacles.

Future Improvements

1. We can use a more powerful wifi dongle or router.
2. Instead of NYU VPN, we could use our own server which has low load which helps in smooth control and reduction of latency.
3. We could contact some big wireless companies like Verizon and can do robot control research and something like that.

Github & Video

Github link which contains all folders and launch files
https://github.com/bethalageetachandraraju/Teleop_Robot
Video Link: https://youtu.be/866OW6hLmJY

Reference Research Paper

1. **Teleoperation Methods and Enhancements Techniques for Mobile robots: A Comprehensive Survey. MD Moniruzzaman, Alexander Rassau, Douglas Chai, Syed Mohammad Shamsul Islam, 2022**

   This Survey paper indicates that the most common control mechanism for teleoperated mobile robots is supervisory control, followed by direct teleoperation. Multimodal approaches, which use multiple sensors and input tools to provide better situational awareness, are not yet widely used. These approaches require high computational and bandwidth capabilities and can increase cognitive workload, but they are useful in complex and dynamic environments. Direct teleoperation, which uses a simple wired or wireless connection, is effective for UGVs, UAVs, and ROVs, but it is best suited for short-distance operations. Video feed-based teleoperation, in which the teleoperator uses a remote control GUI to view a video feed of the environment, is also used to provide situational awareness. Even Though, direct teleoperation is more forgiving of latency, intermittency, and data loss than multimodal or supervisory control methods.

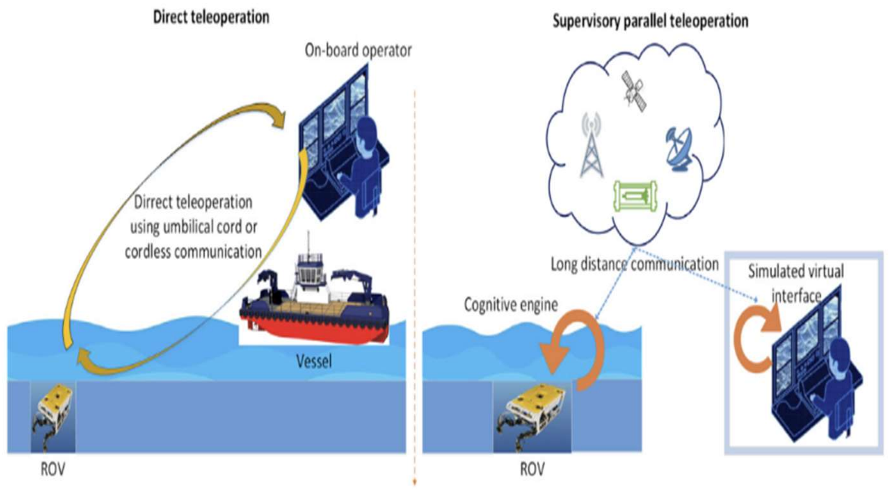

Direct teleoperation
On-board operator
Dirrect teleoperation using umbilical cord or cordless communication
Vessel
ROV
Supervisory parallel teleoperation
Long distance communication
Simulated virtual interface
Cognitive engine
ROV

Fig :- Teleoperating ROV

This paper inspired us to take up one of the methods of teleoperations that is supervised parallel teleoperation over a VPN network to establish communication between the mobile robot and the teleoperator.

2. **High Latency Unmanned Ground Vehicle Teleoperation Enhancement Through Video Transformation, MD Moniruzzaman, Alexander Rassau, Douglas Chai, Syed Mohammad Shamsul Islam, 2022**

   This paper introduces a model that is particularly effective for teleoperation over long distances and with high latency, such as when operating at high speeds. However, it can also be customized for other teleoperation scenarios. In addition, the simulations described in the paper can be used to train operators to perform tasks in high latency environments and can provide visual and control data for artificial intelligence research. The video transformation methods described in the paper have demonstrated significant improvements in reducing the impact of high latency.