LIDAR MAPPING

TEAM MEMBERS:

ABHISHEK KUMAR RAWAT

HIMANSHU PAL

MANROOP SINGH

MD. REHAN SHAKOOR

MENTORS:

LAKSHAY MADAN

ASLEESHA KOMARAPU

Objective:

To perform 3D lidar mapping using ALTM (Airborne Laser Terrain Mapping) technology.

Project work is divided into three phases.

Phase 1

Getting the lidar data online or creating data through limulator software.

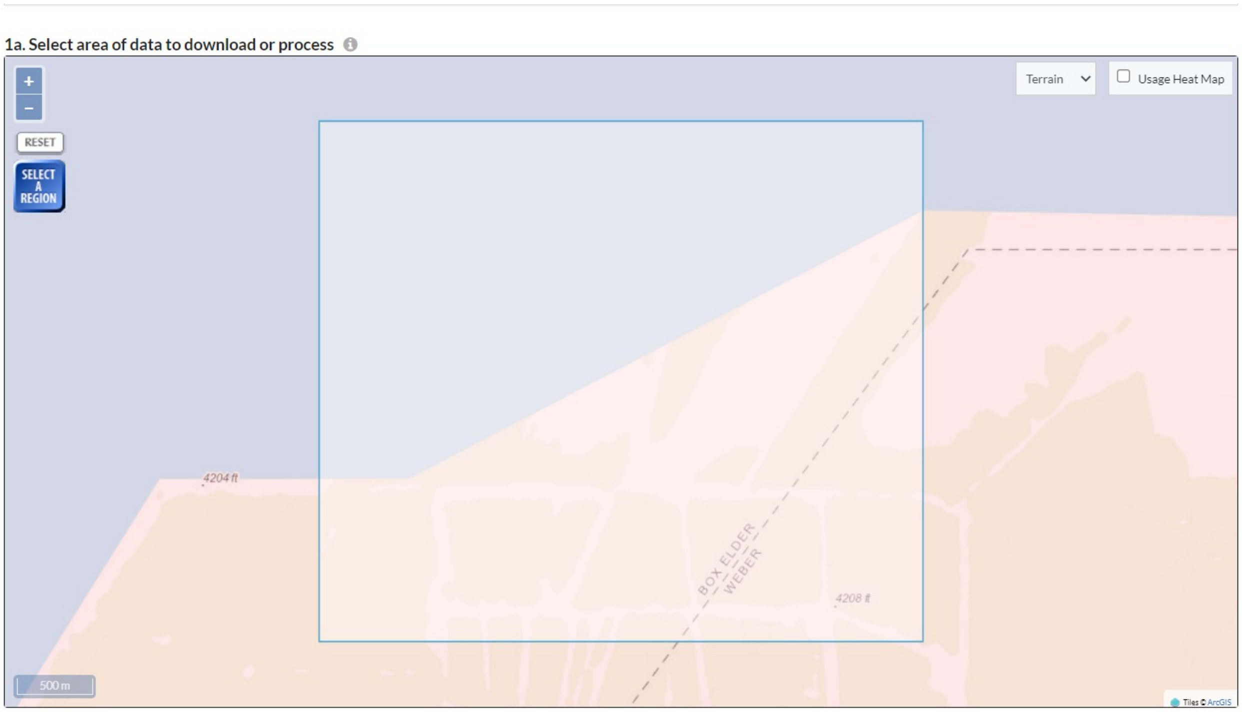
The lidar datasets have been downloaded from
https://portal.opentopography.org/datasets
.LAS OR .LAZ(compressed version of .LAS) format.

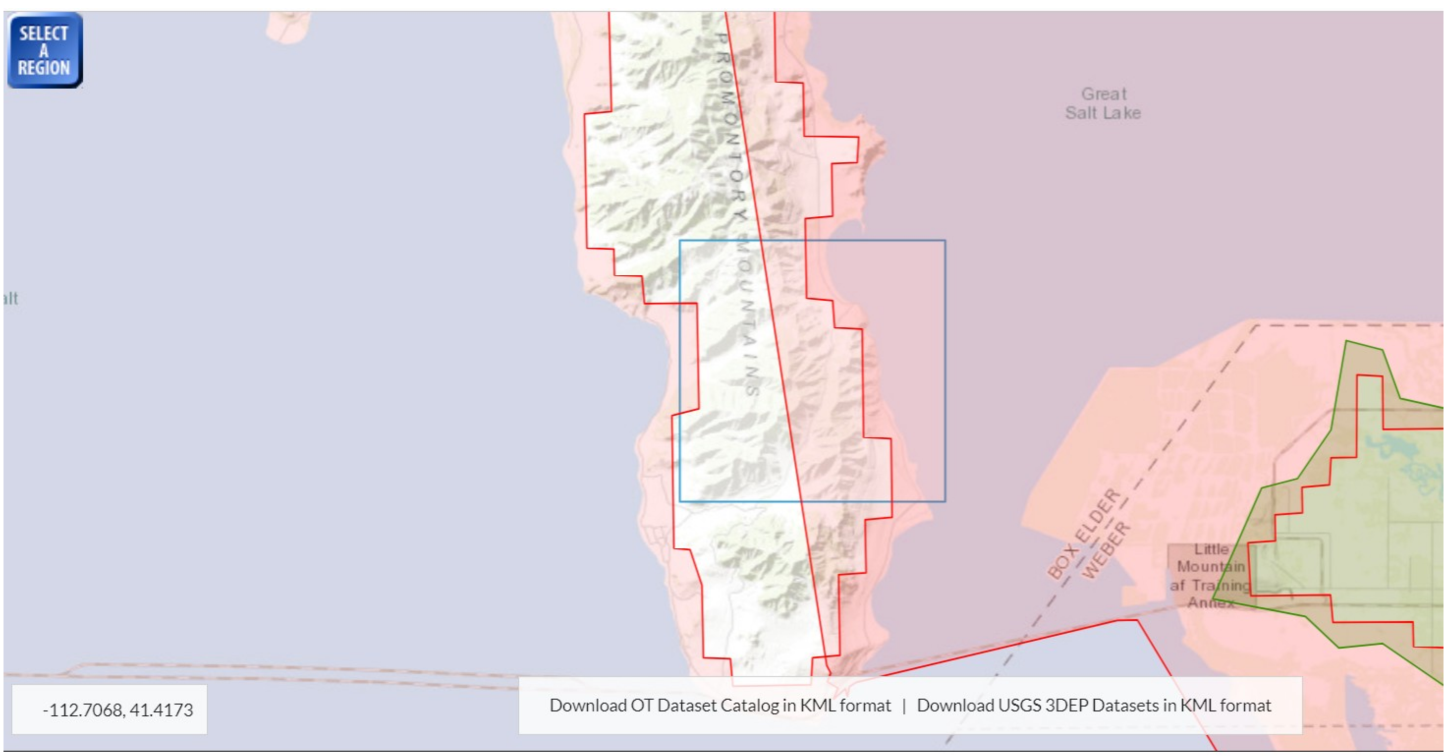
The data is collected by selecting a particular area from topographic map
by assigning the required format and the extracting the required amount of features.

We have to select the features of the dataset i.e.

1. The grid resolution
2. The grid format
3. 3D point cloud visualization
4. Noise feature
5. Hydrologic terrain analysis product(tauDEM)

These are some of images of area selection and dataset generation:





PHASE 2:

Plotting the point cloud data and creating terrain in MATLAB using the image processing toolbox and lidar toolbox.

Now, the terrain generation is done in two formats. Format 1 is reading the data and plotting it in heat map format and in Format 2 we will read the data and visualizing it based on the classification point attributes.

Steps for image processing for format 1:

1. Now in the process of plotting , we first create an object that points towards our lidar data using the method:

   lasFileReader()

   It creates the properties of data:

```
lasreader =
  lasFileReader with properties:

        FileName: 'C:\Users\MANROOP SINGH\Downloads\points.laz'
           Count: 749608
      LasVersion: '1.2'
         XLimits: [2.5088e+05 2.5147e+05]
         YLimits: [9.4412e+05 9.4465e+05]
         ZLimits: [-2.8700 35.8700]
   GPSTimeLimits: [0 sec    0 sec]
      NumReturns: 1
      NumClasses: 3
```

2. After the object is created we create an another object that reads the reads our point cloud data. The method used for this is:
   ReadPointCloud()

   It generates the properties:

```
ptcloud =
  pointCloud with properties:

     Location: [749608×3 single]
        Count: 749608
      XLimits: [2.5088e+05 2.5147e+05]
      YLimits: [9.4412e+05 9.4465e+05]
      ZLimits: [-2.8700 35.8700]
        Color: []
       Normal: []
    Intensity: [749608×1 uint8]
```

3. Now after the dataset is ready we plot the dataset and create the terrain by using the method:
   ptshow()

   The generated plots are:

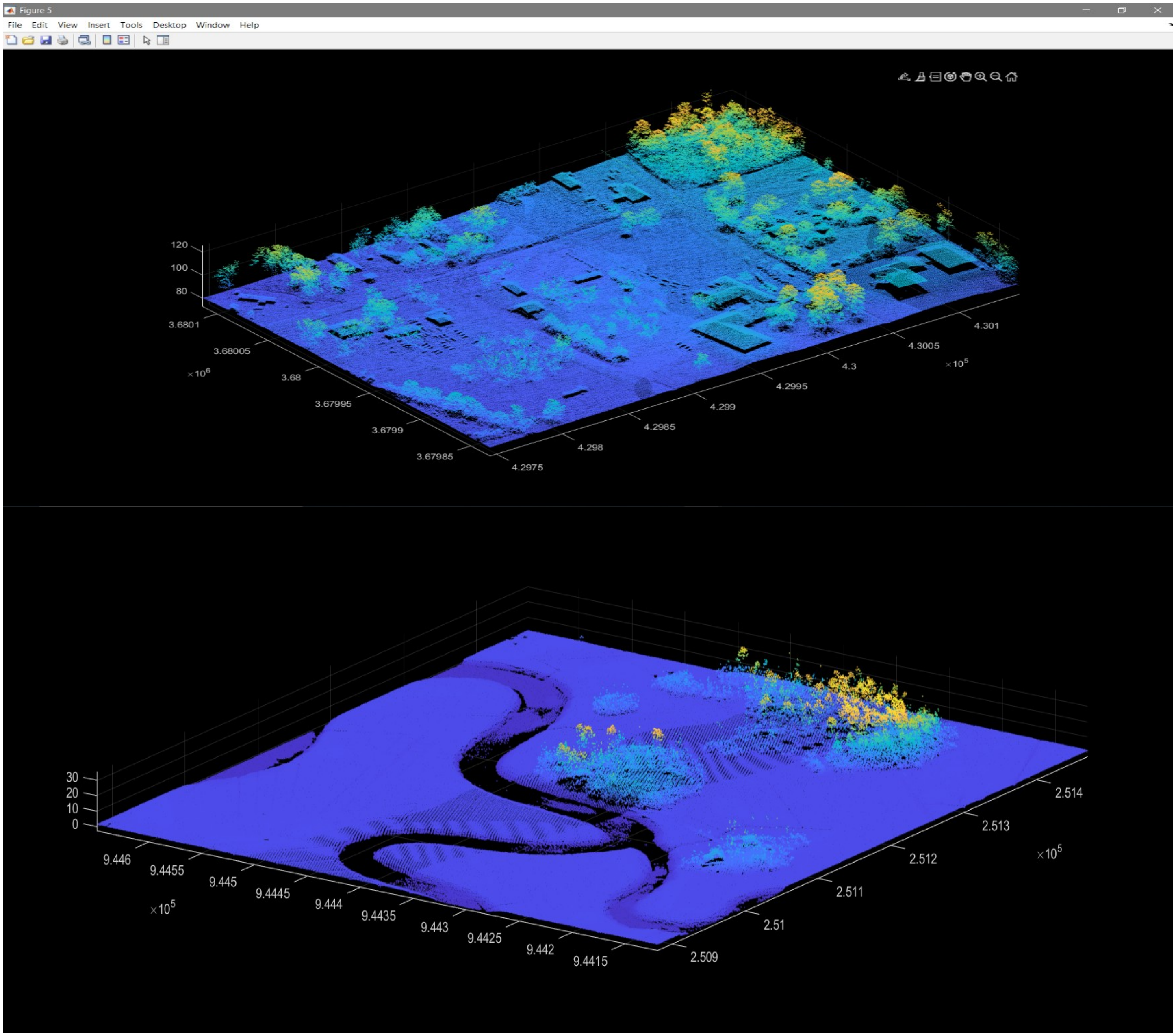


Figure 5
File Edit View Insert Tools Desktop Window Help
120
100
80
3.6801
3.68005
×10^6
3.68
3.67995
3.6799
3.67985
4.2975
4.298
4.2985
4.299
4.2995
4.3
4.3005
4.301
×10^5
30
20
10
0
9.446
9.4455
9.445
9.4445
9.444
9.4435
9.443
9.4425
9.442
9.4415
×10^5
2.509
2.51
2.511
2.512
2.513
2.514
×10^5

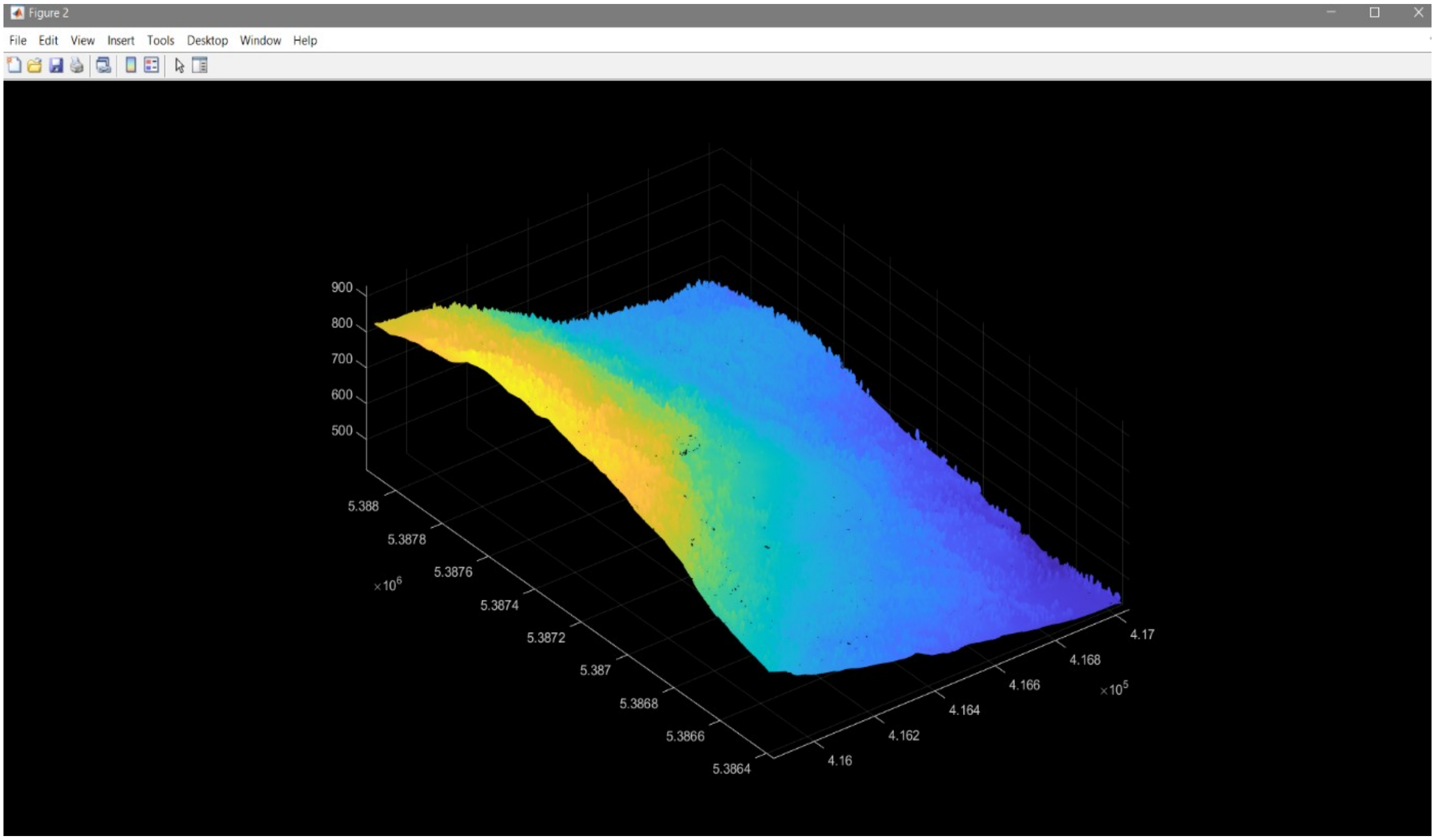
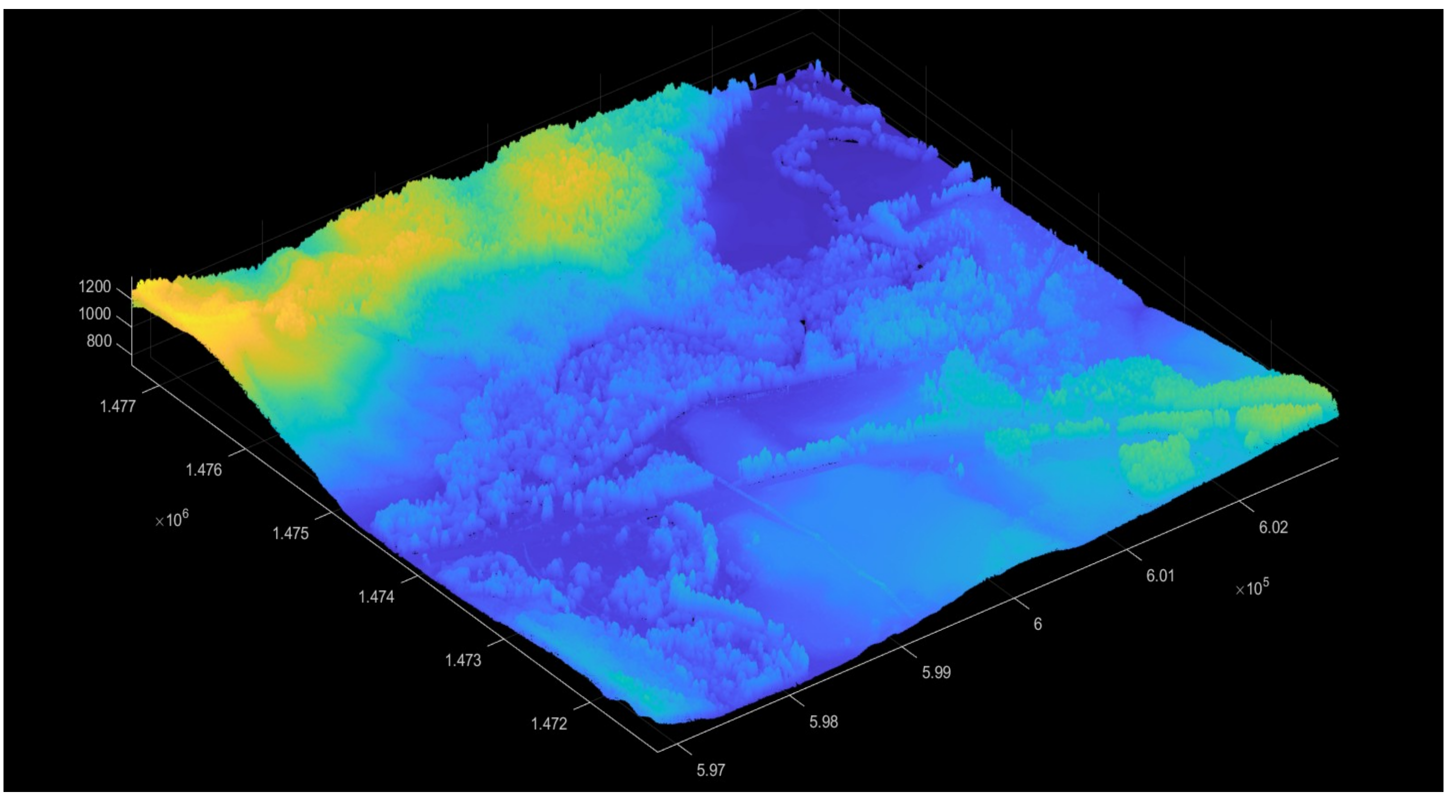
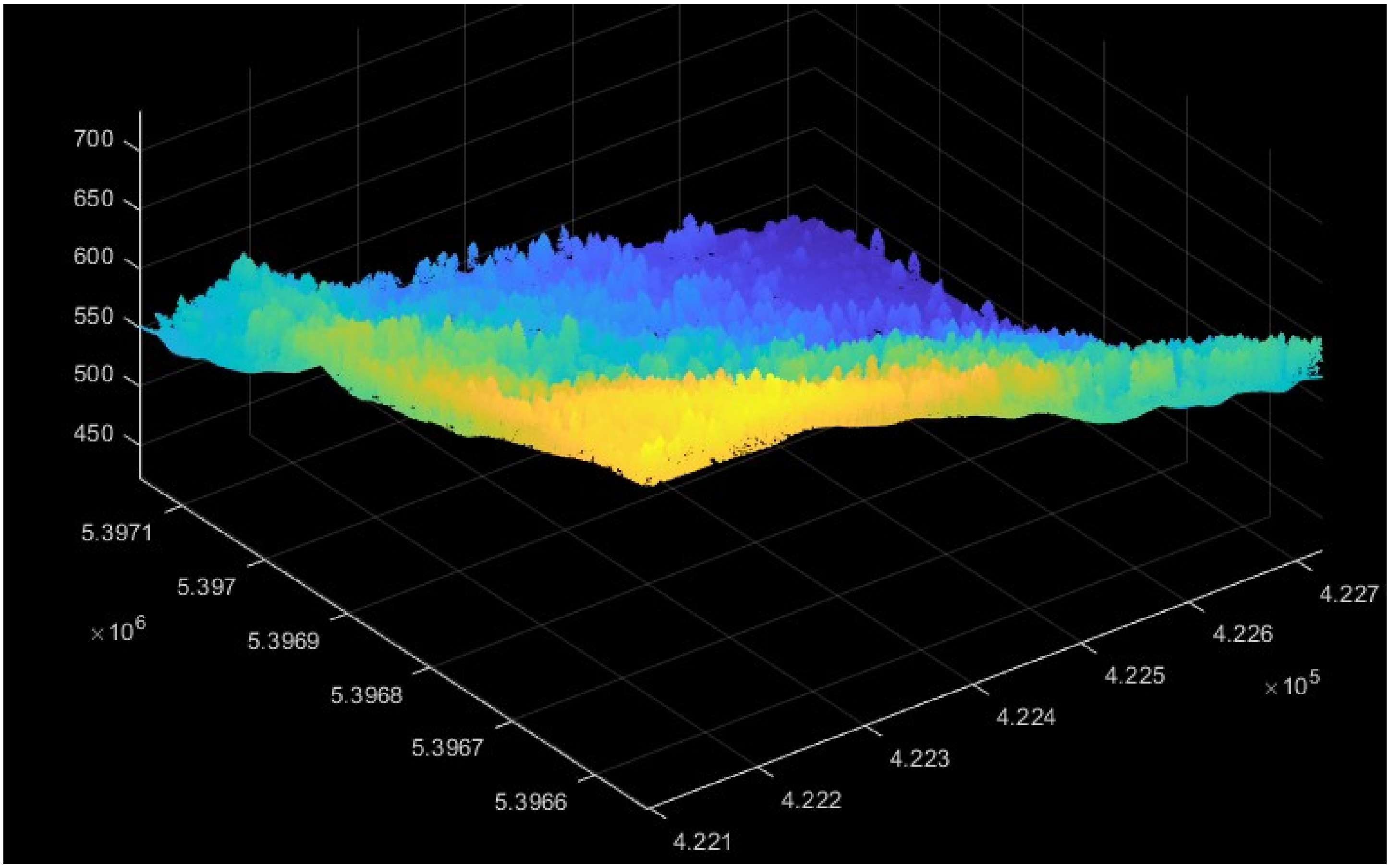


Figure 2
File Edit View Insert Tools Desktop Window Help
900
800
700
600
500
5.388
5.3878
5.3876
5.3874
5.3872
5.387
5.3868
5.3866
5.3864
×10^6
4.16
4.162
4.164
4.166
4.168
4.17
×10^5


1200
1000
800
1.477
1.476
1.475
1.474
1.473
1.472
×10^6
5.97
5.98
5.99
6
6.01
6.02
×10^5



Steps for image processing for format 2:

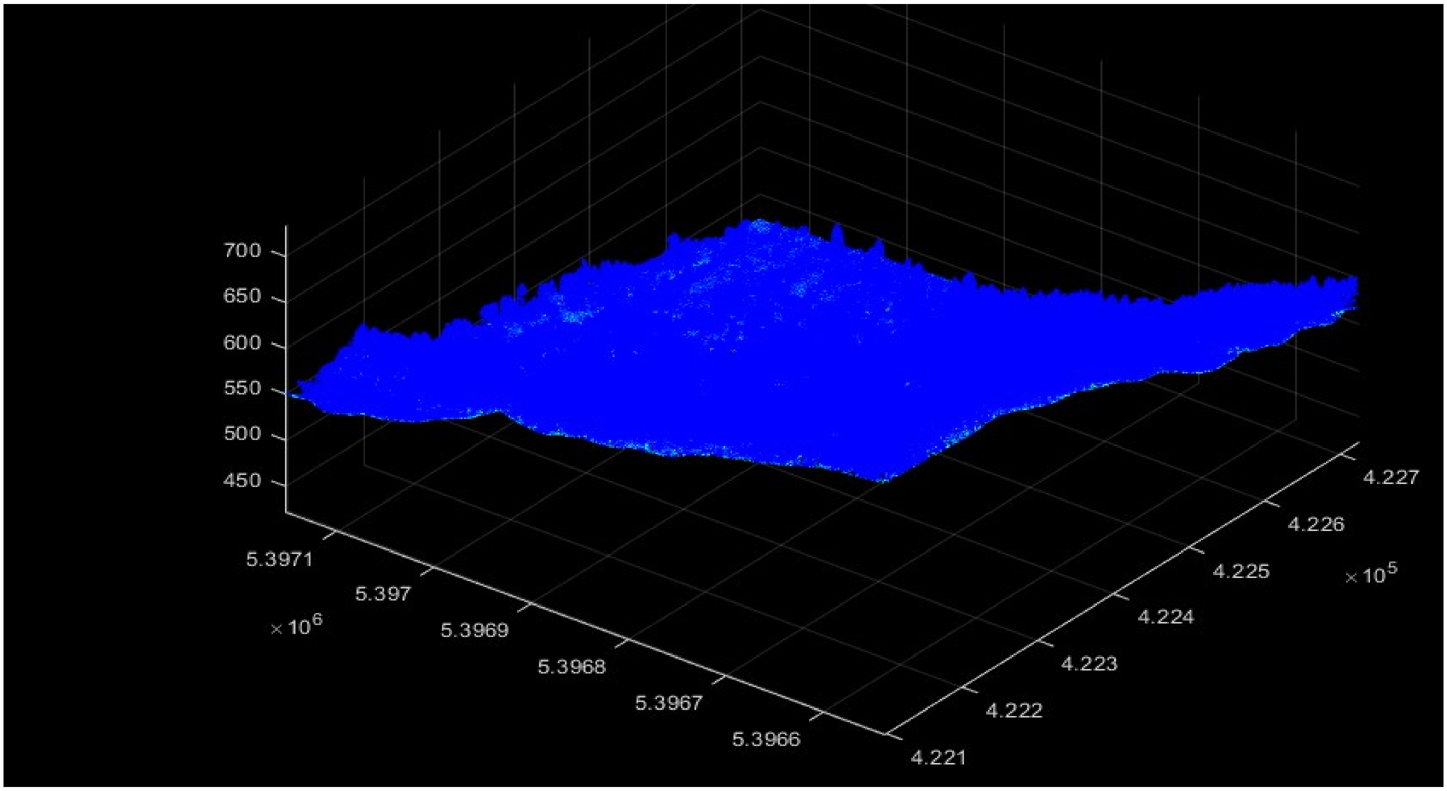
1. The first step is same creating the object that points towards our point cloud data using the lasFileReader() method.

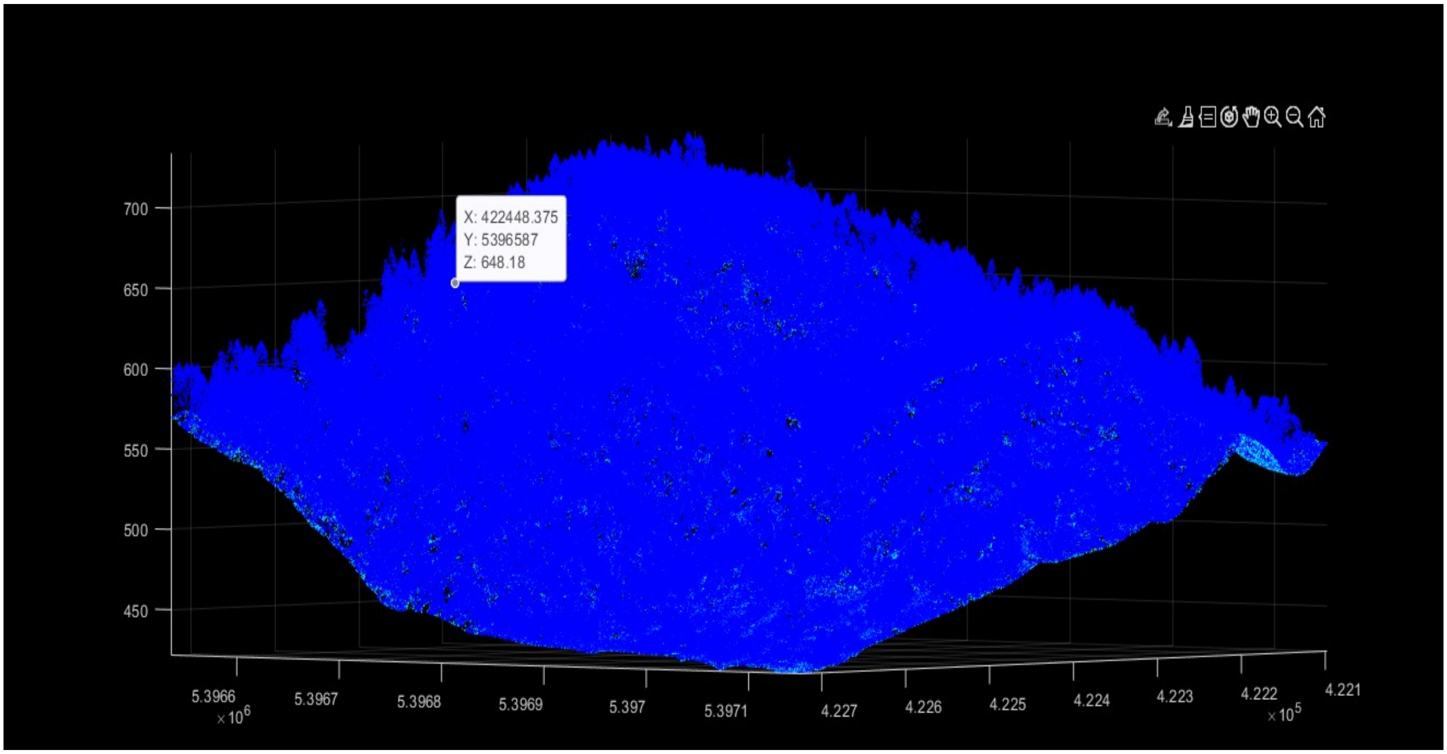
2. After that the point cloud data is read and the associated attributes are added from the LAZ file using the readPointCloud() method.

3. After this the points are coloured on the basis of classified attributes using the reshape() method.

4. After the points are classified they are plotted and terrain is generated using ptshow() method.

Some plots generated are :





X: 422448.375
Y: 5396587
Z: 648.18
700
650
600
550
500
450
5.3966
$\times 10^6$
5.3967
5.3968
5.3969
5.397
5.3971
4.227
4.226
4.225
4.224
4.223
4.222
4.221
$\times 10^5$

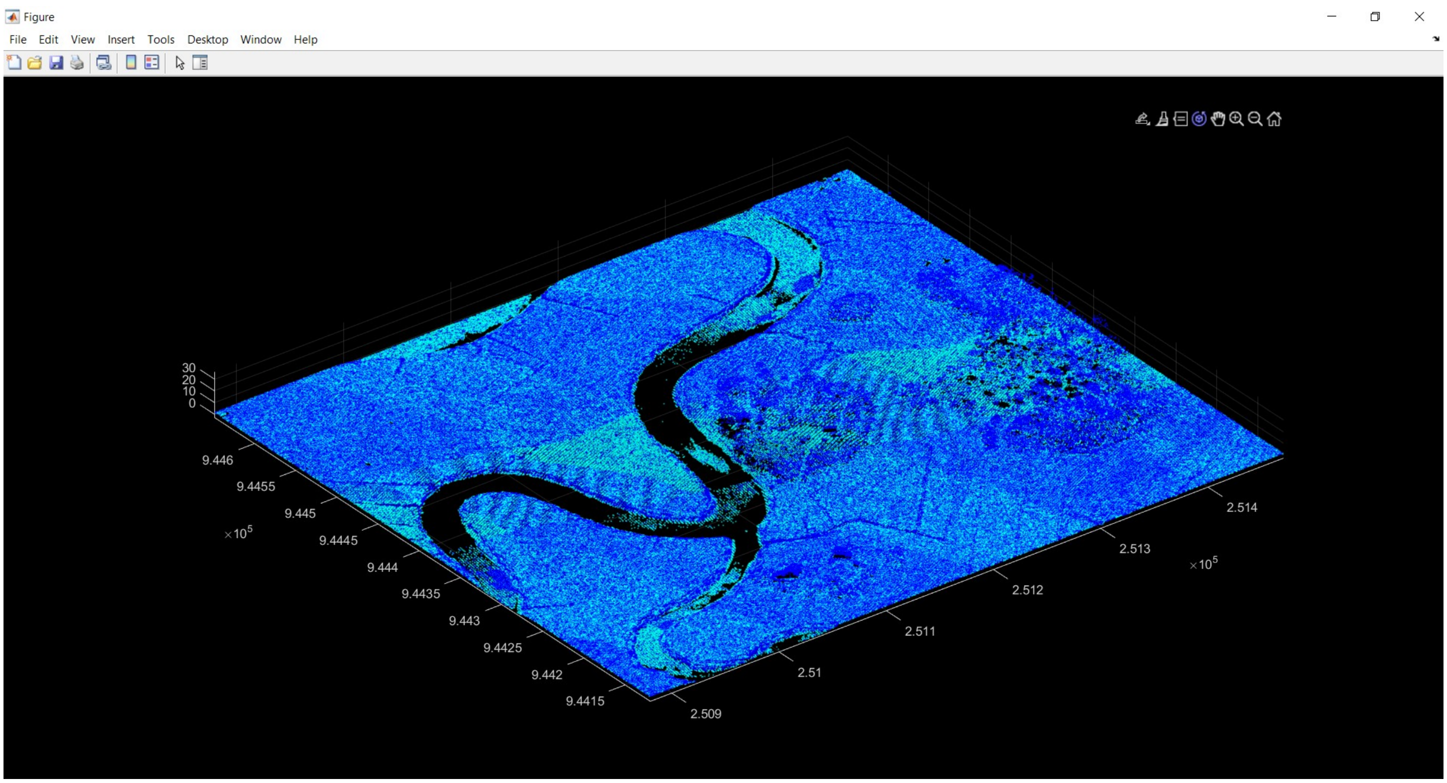


Figure
File Edit View Insert Tools Desktop Window Help
30
20
10
0
9.446
9.4455
9.445
$\times 10^5$
9.4445
9.444
9.4435
9.443
9.4425
9.442
9.4415
2.509
2.51
2.511
2.512
2.513
2.514
$\times 10^5$

Phase 3 :

Creating a UAV simulation by fitting a lidar sensor on quadcopter using MATLAB[UAV toolbox, ROS toolbox, Image processing toolbox], Simulink & ROS gazebo.

Steps for creating simulation in MATLAB & Simulink:

1. First a UAV simulation scenario is defined using the the method uavScenario().The uavScenario object generates a simulation scenario consisting of static meshes, UAV platforms & sensors in a 3-D environment
   The following features are shown:

   a. Simulation update rate, specified as a positive scalar in Hz.
   b. Scenario origin in geodetic coordinates, defined as a 3-element vector of scalars in the form [latitude longitude altitude].

2. After that we use addInertialFrame() method to define a new inertial frame in the UAV scenario scene.

3. Then we create a new static mesh to UAV scenario using the method assMesh(). Now it will add a new static mesh to the UAV scenario scene by specifying the mesh type, geometry, and color.

   a. UAV scenario, specified as uavScenario object.
   b. Mesh type is specified as “cylinder”, “surface”, “terrain”, “polygon” or “custom”.

4. Afer the mesh is created we use the load(“buildingData.nat”) to add building polygons. This method will create a datatype with field : buildingData: {1x11 cell}.

5. We will the load() method again and will use the parameter “flightData.mat”. It will help us to specify the path of the UAV. It contains a struct data type with fields : orientation : [1x3x30 double] and position : [1x3x30 double].

6. Now we will create a uavPlatform object using the method uavPlatform(). The uavPlatform object represents an unmanned aerial vehicle (UAV) platform in agiven UAV scenario. It also simulates the lidar sensor readings for the platform.

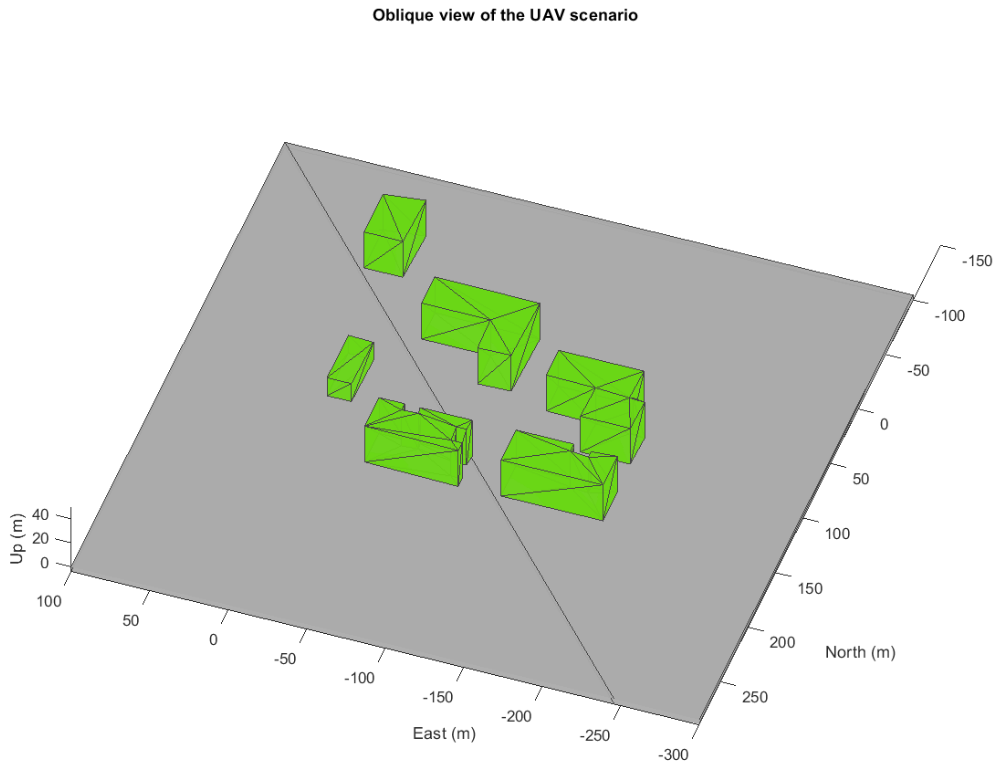
7. After the platform and buiding all are created we use uavLidarPointCloudGenerator() method to create point cloud data from meshes.

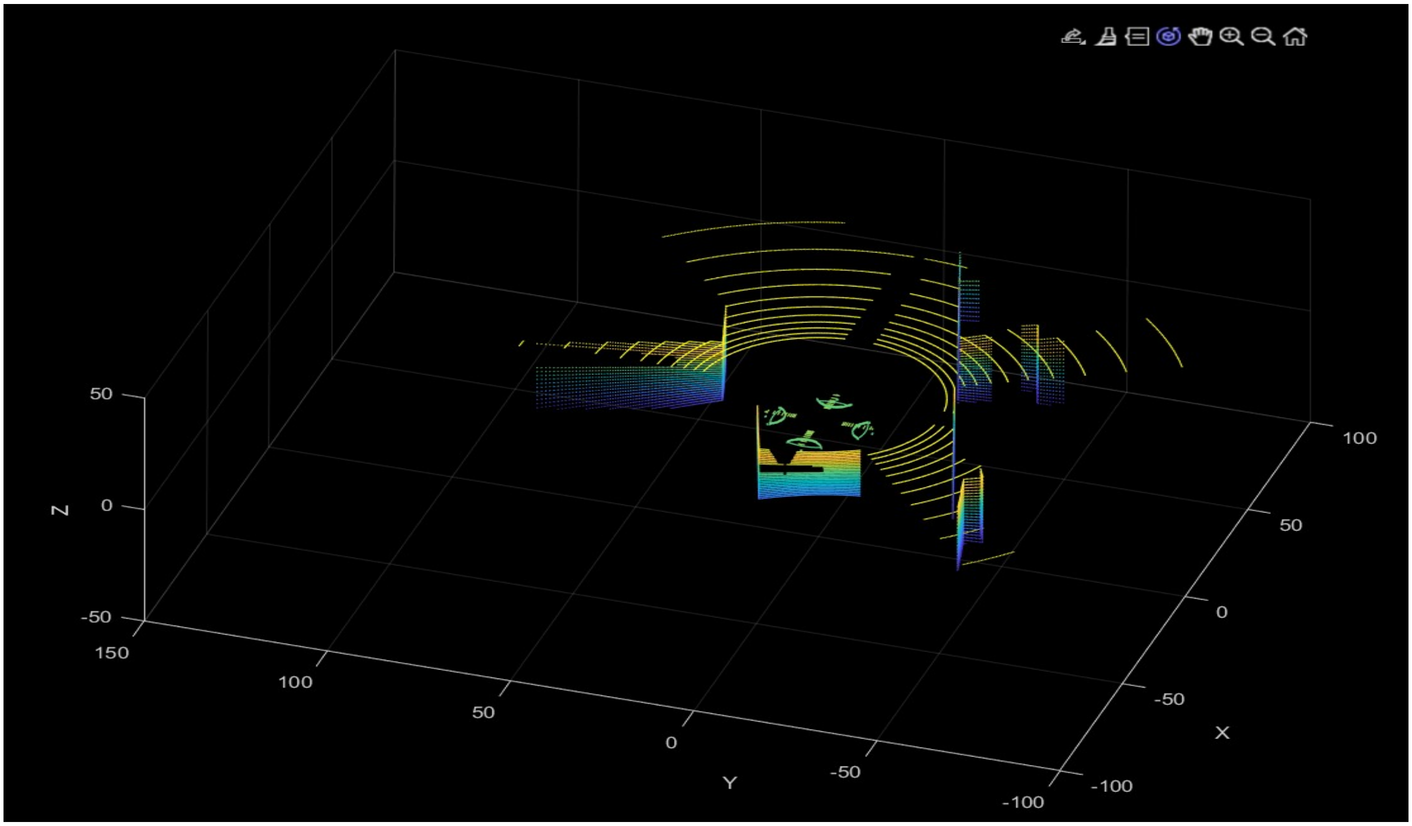
8. Now the main thing is to add lidar sensor for our UAV scenario using uavSensor() method. It will create a rigidly attached UAV sensor to the UAV platform, specified as a uavPlatform object.

9. Now our simulation is ready and we have to create an object to store our 3-D point cloud. The pointCloud object creates point cloud data from a set of points in a 3-D coordinate system. The points generally represent the x, y and z geometric coordinates of a sample surface or an environment.

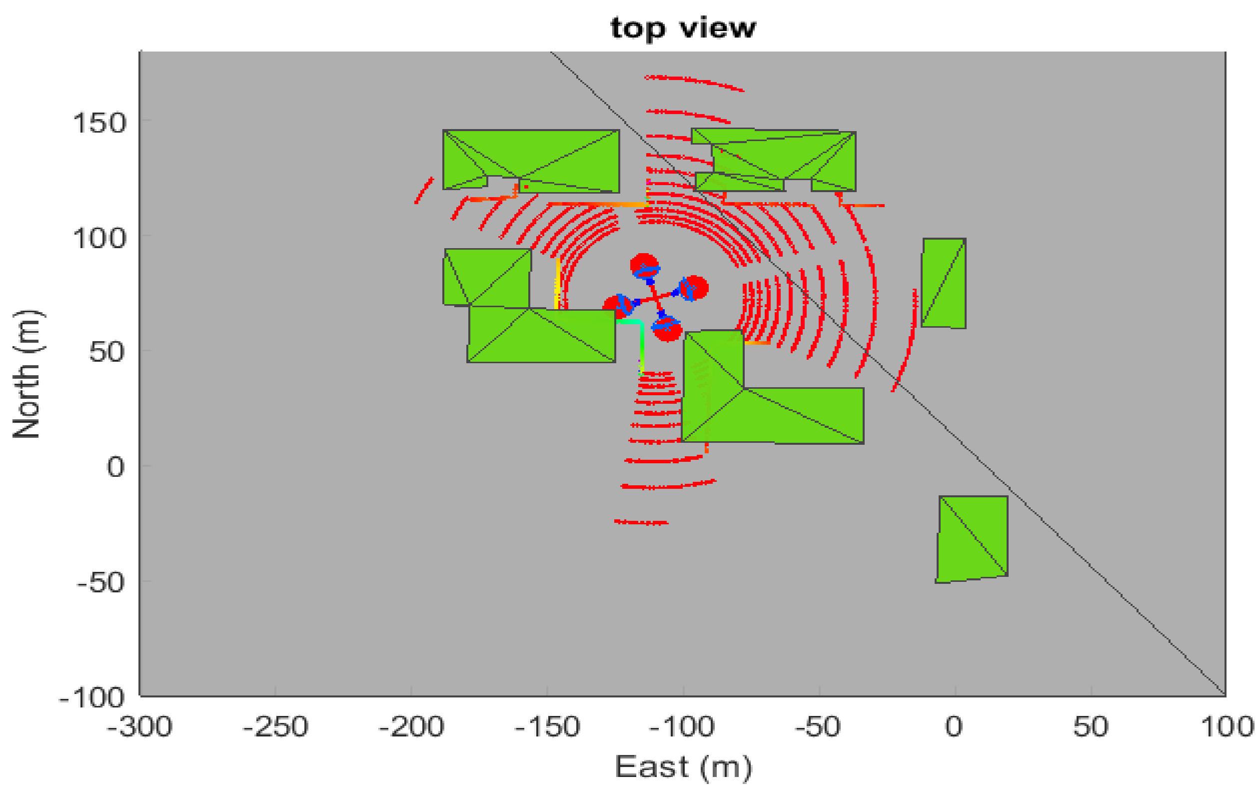
10. Now all our code is ready to create the simulation we have to use the scatter3 object for 3-D scatter plot Pcplayer object to visualize streaing 3-D point cloud data.

This is some of the work:

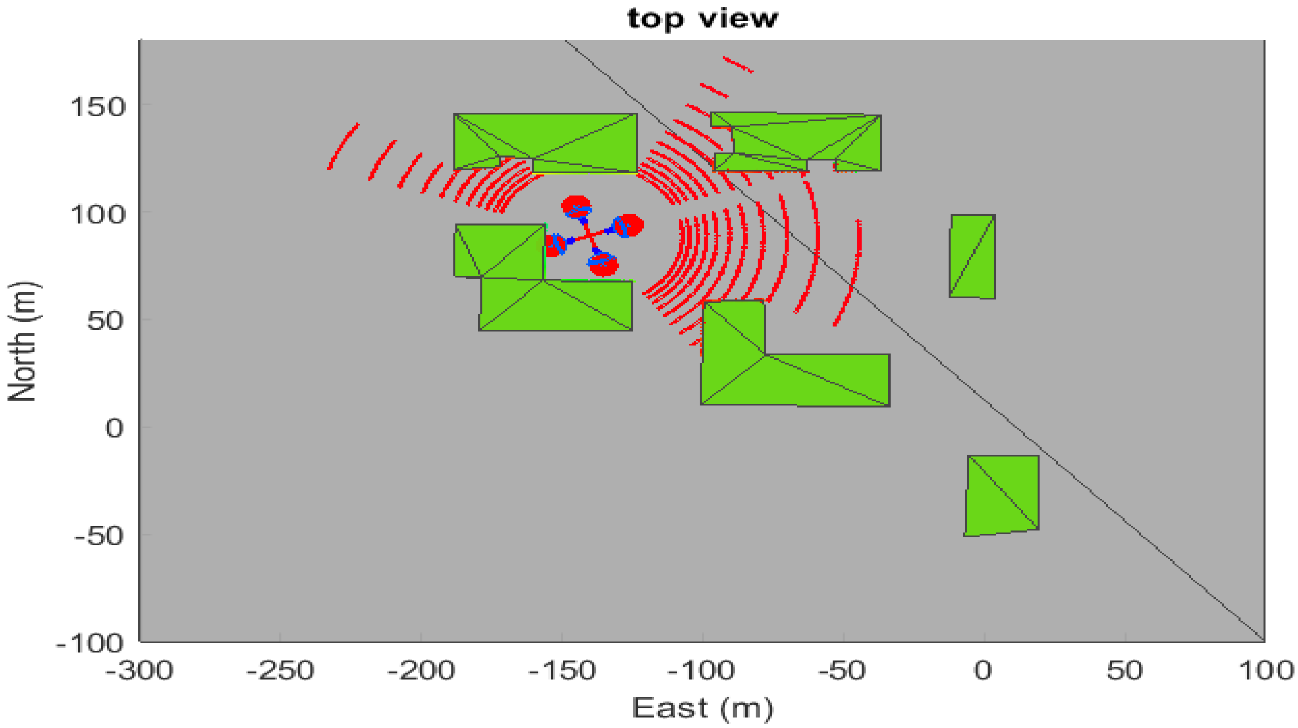


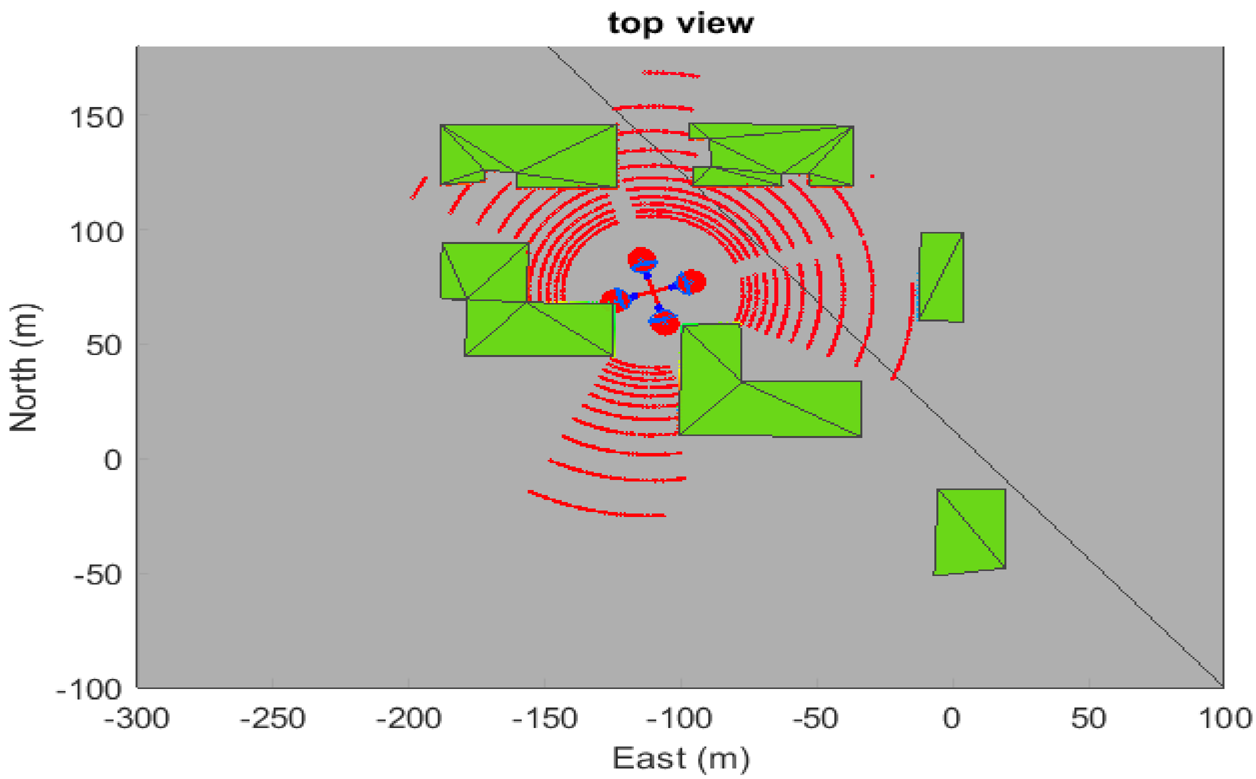


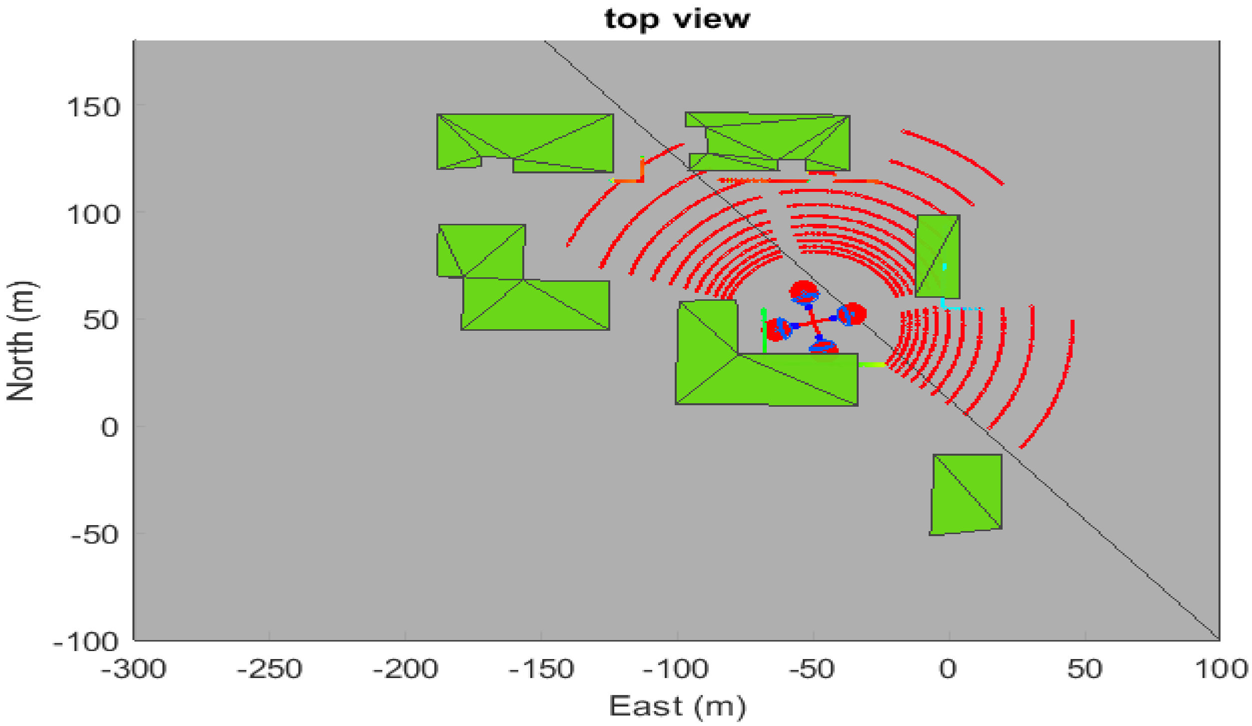
50
0
-50
Z
150
100
50
0
Y
-50
-100
100
50
0
-50
X
-100

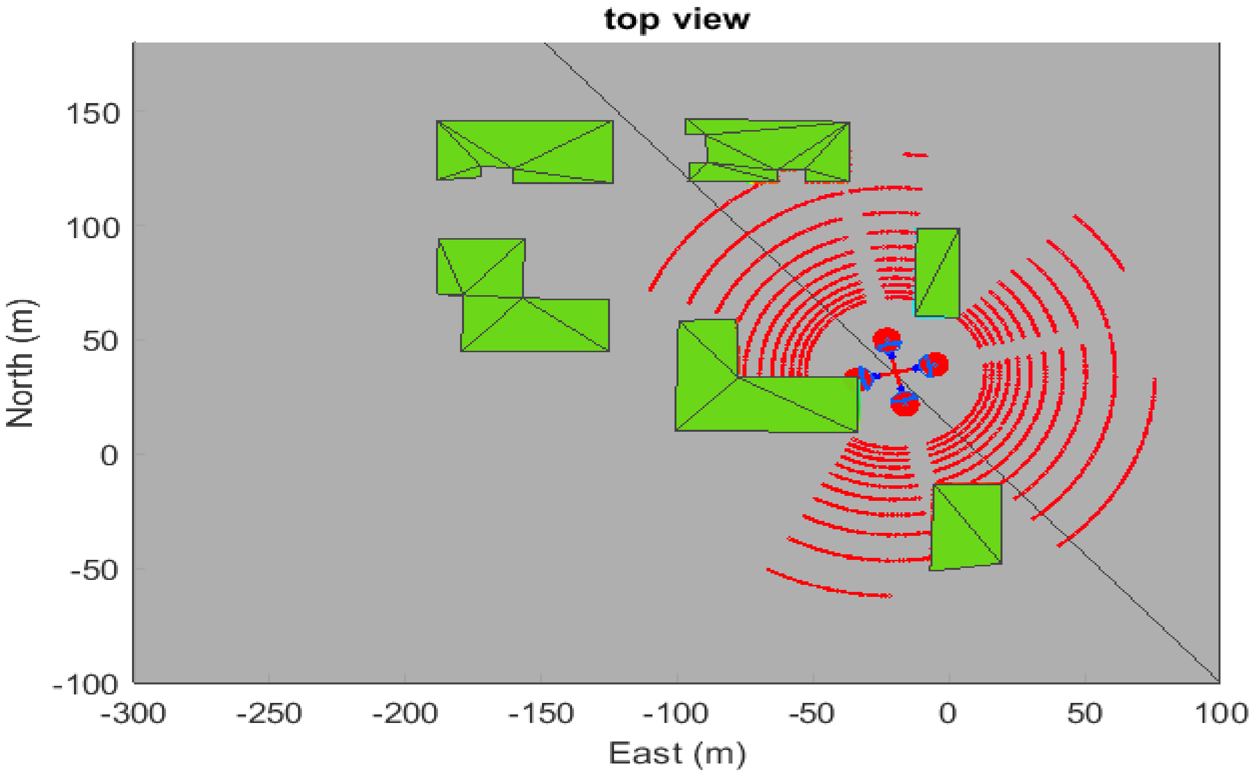


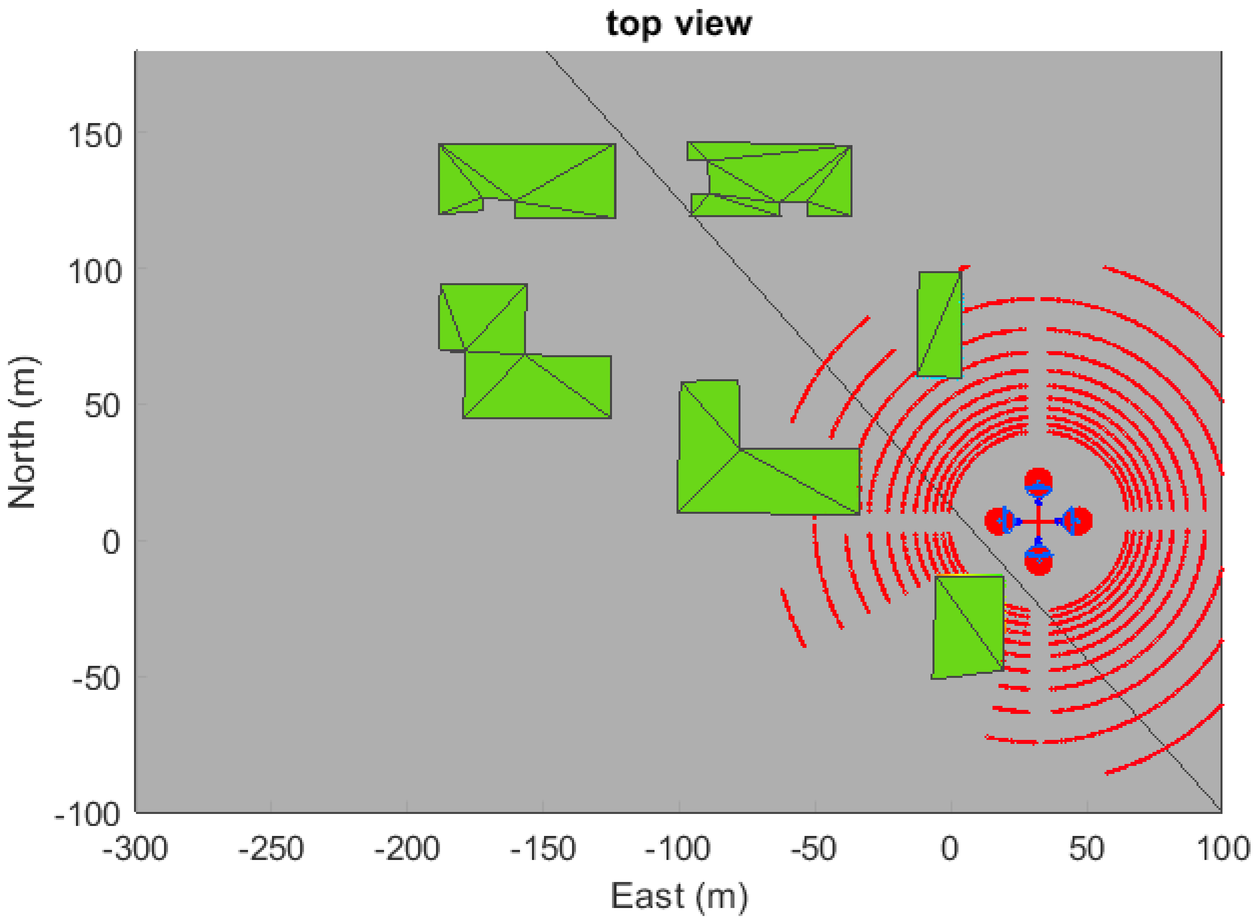
top view
150
100
50
0
-50
-100
North (m)
-300
-250
-200
-150
-100
-50
0
50
100
East (m)

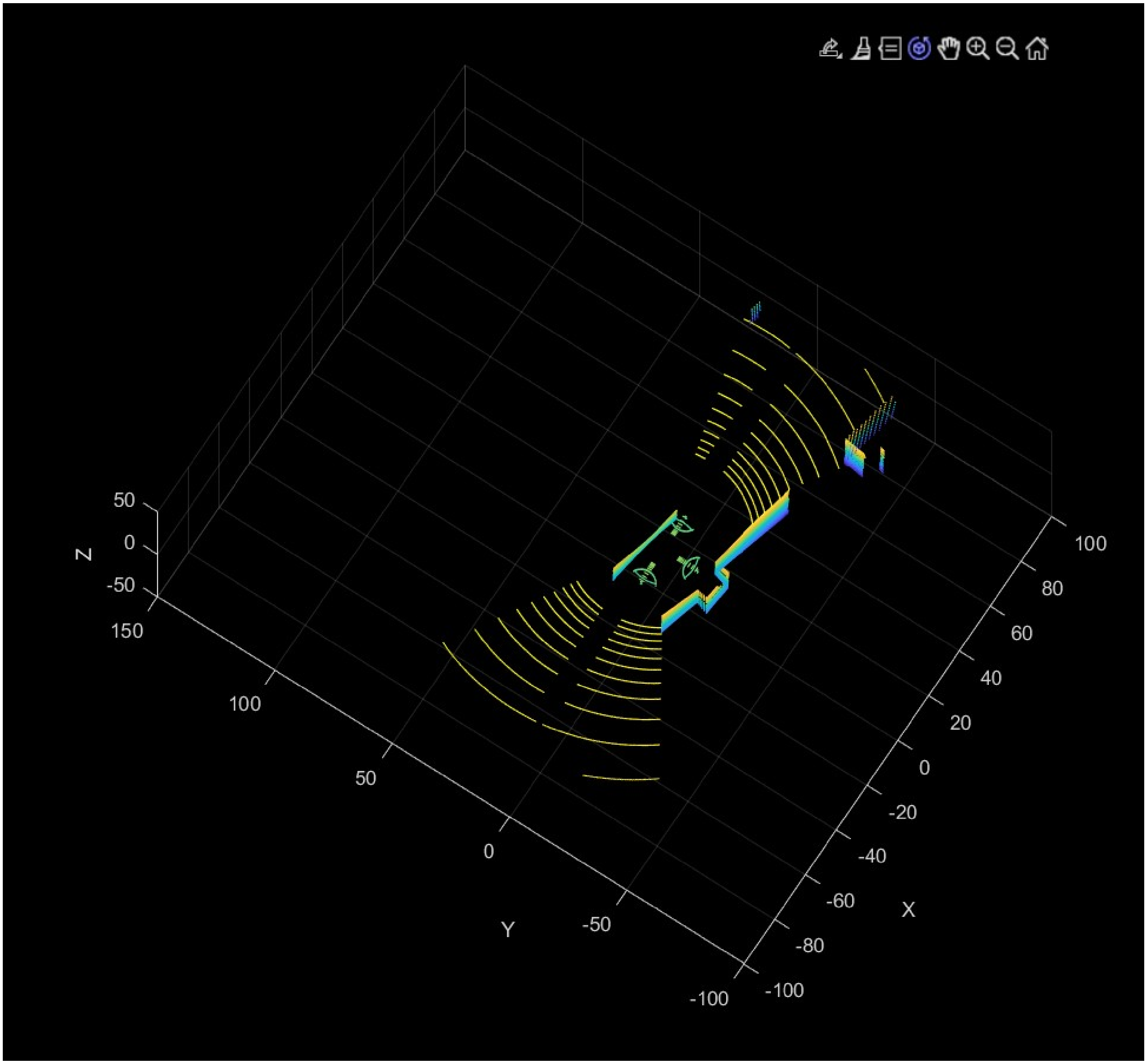

top view
North (m)
East (m)
150
100
50
0
-50
-100
-300
-250
-200
-150
-100
-50
0
50
100


top view
North (m)
East (m)
150
100
50
0
-50
-100
-300
-250
-200
-150
-100
-50
0
50
100

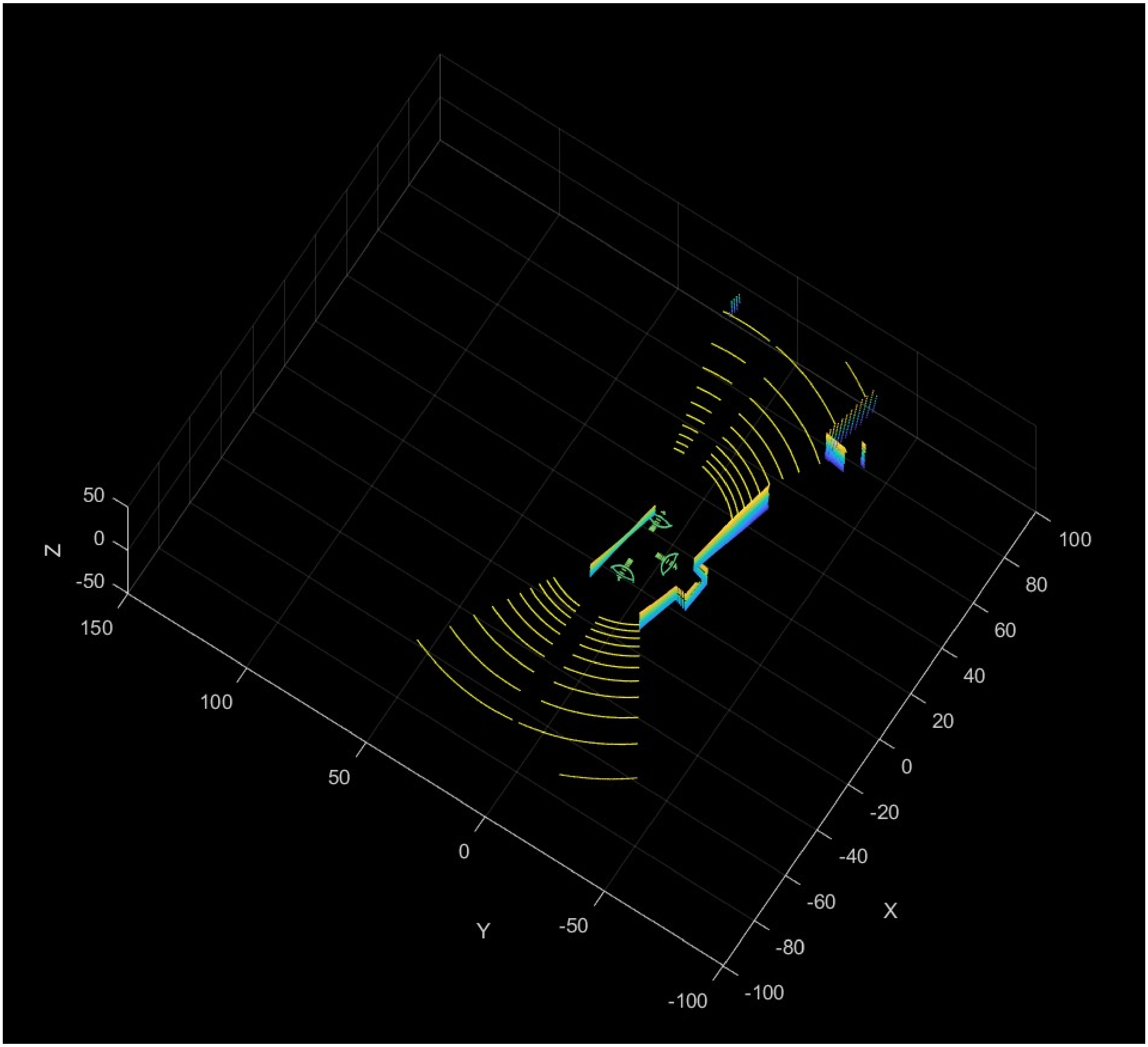

top view
150
100
50
0
-50
-100
North (m)
-300
-250
-200
-150
-100
-50
0
50
100
East (m)


top view
150
100
50
0
-50
-100
North (m)
-300
-250
-200
-150
-100
-50
0
50
100
East (m)

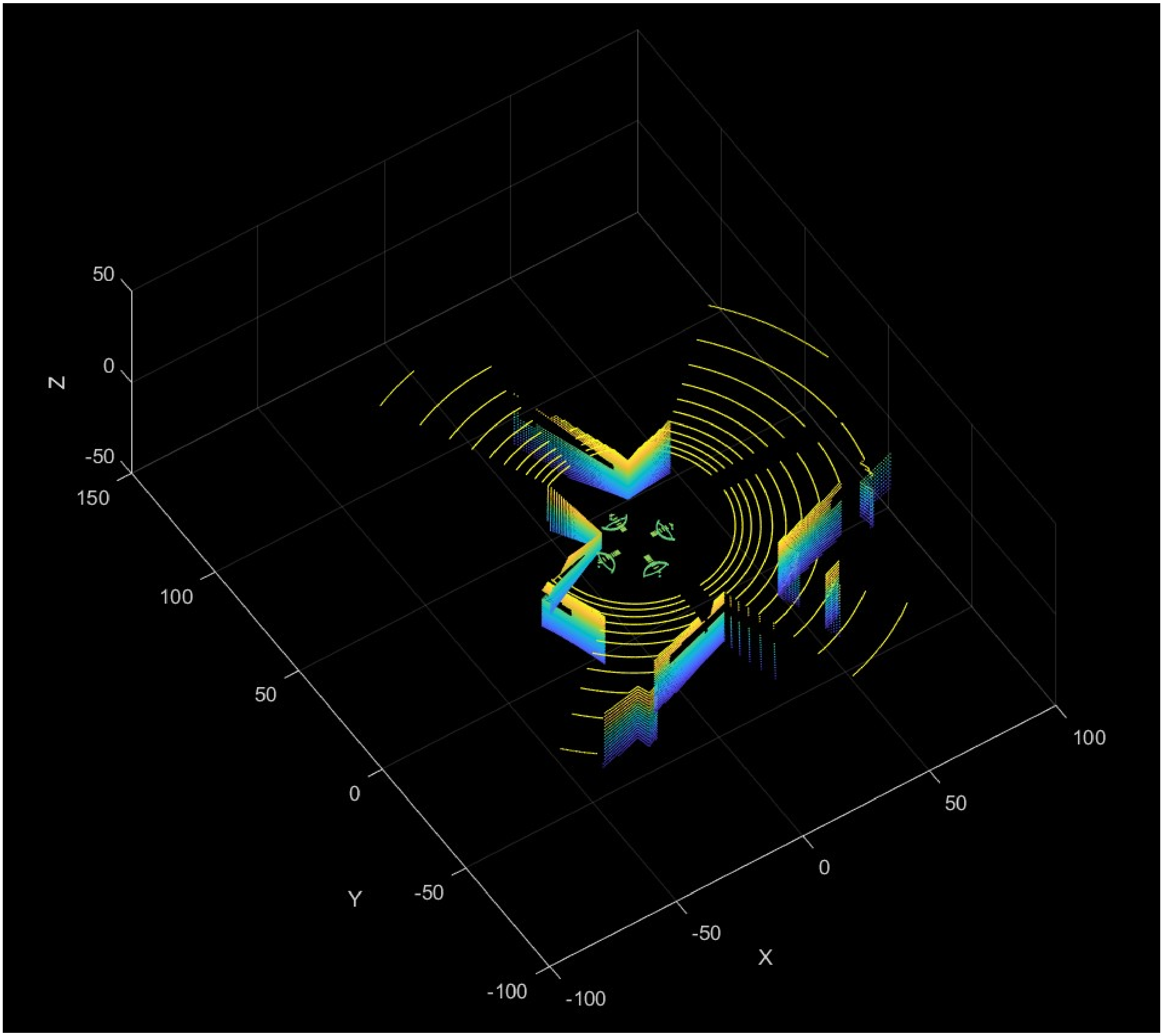

top view
150
100
50
0
-50
-100
North (m)
-300
-250
-200
-150
-100
-50
0
50
100
East (m)

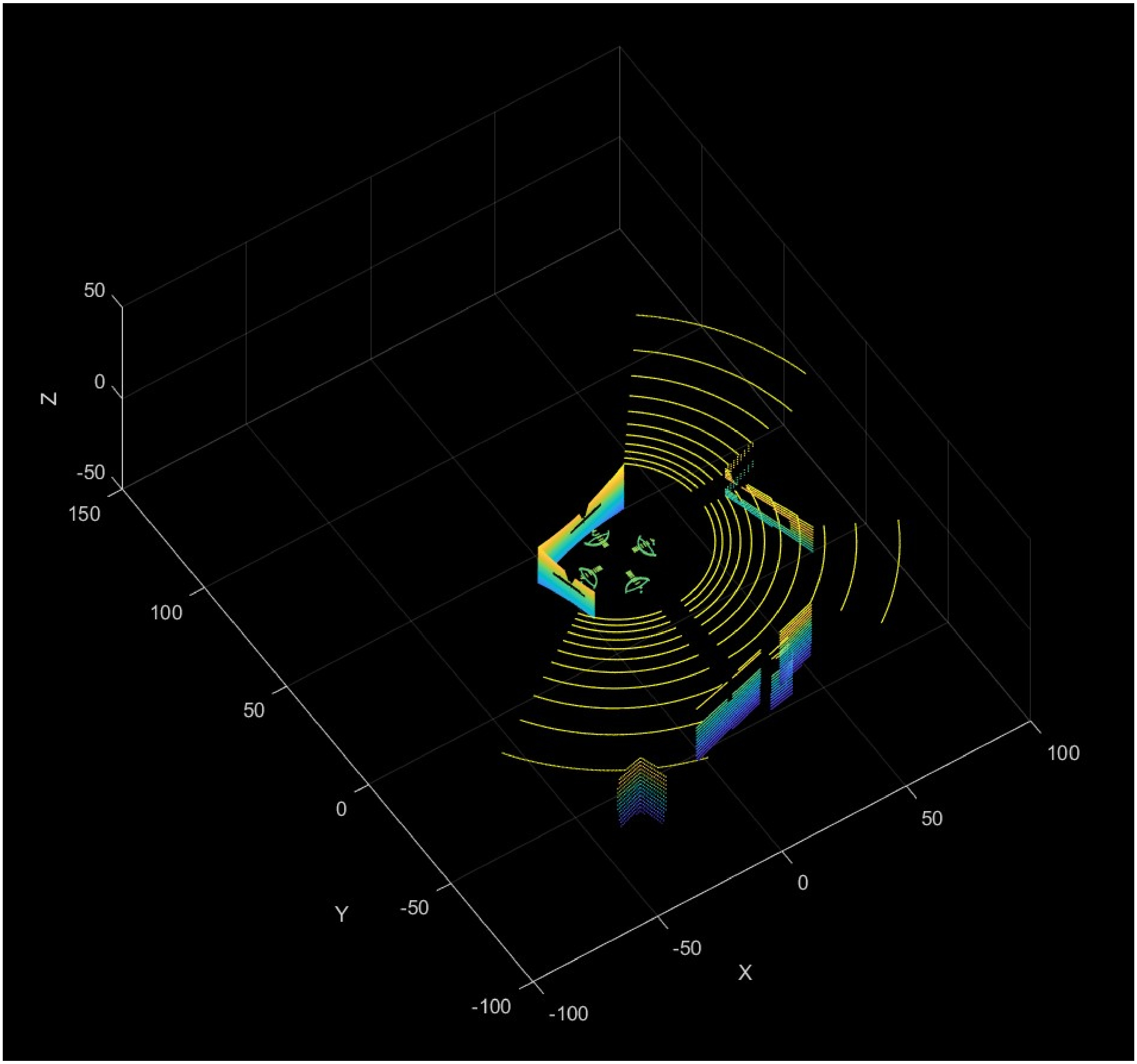


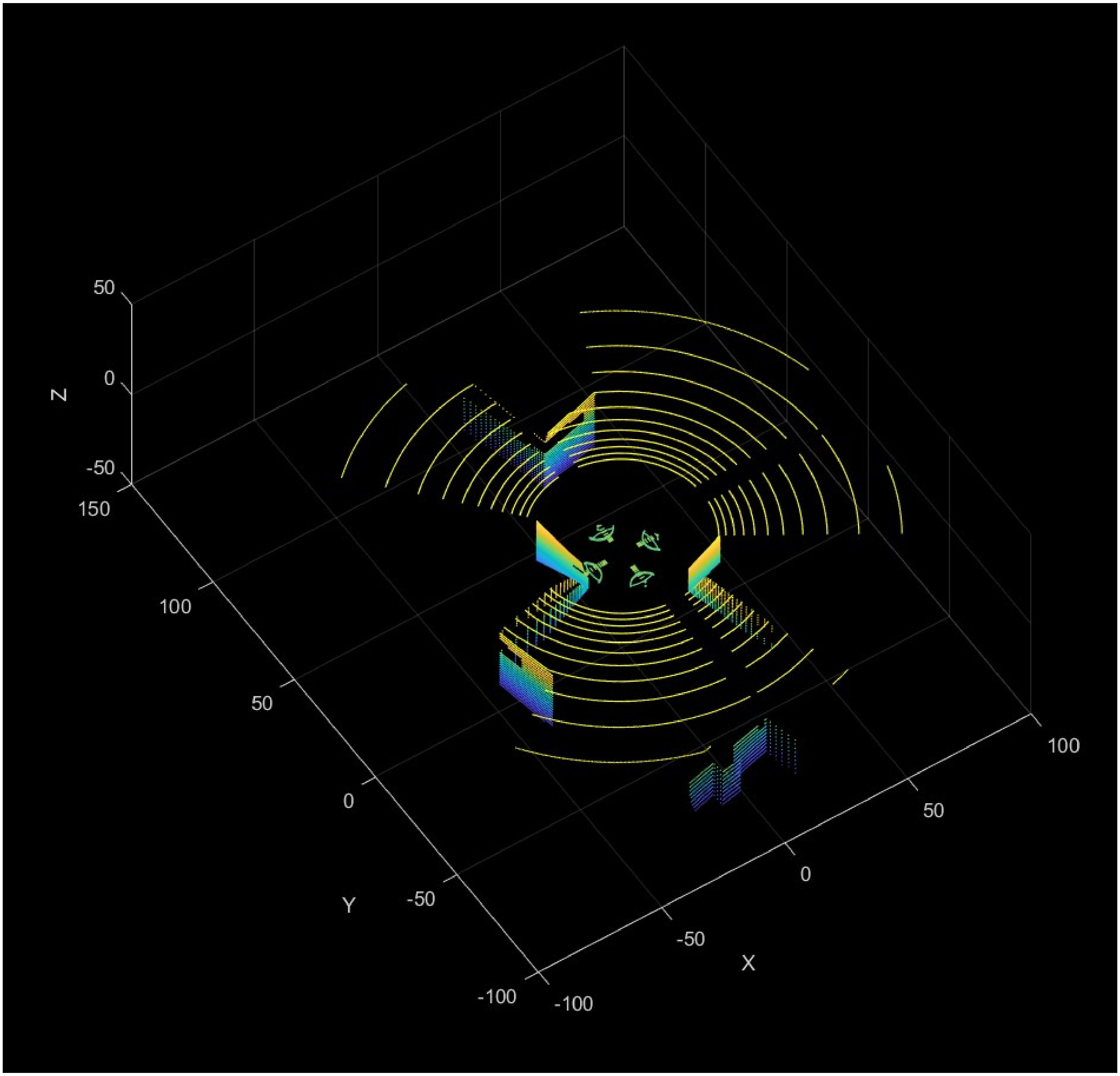
50
0
-50
Z
150
100
50
0
-50
-100
Y
100
80
60
40
20
0
-20
-40
-60
-80
-100
X

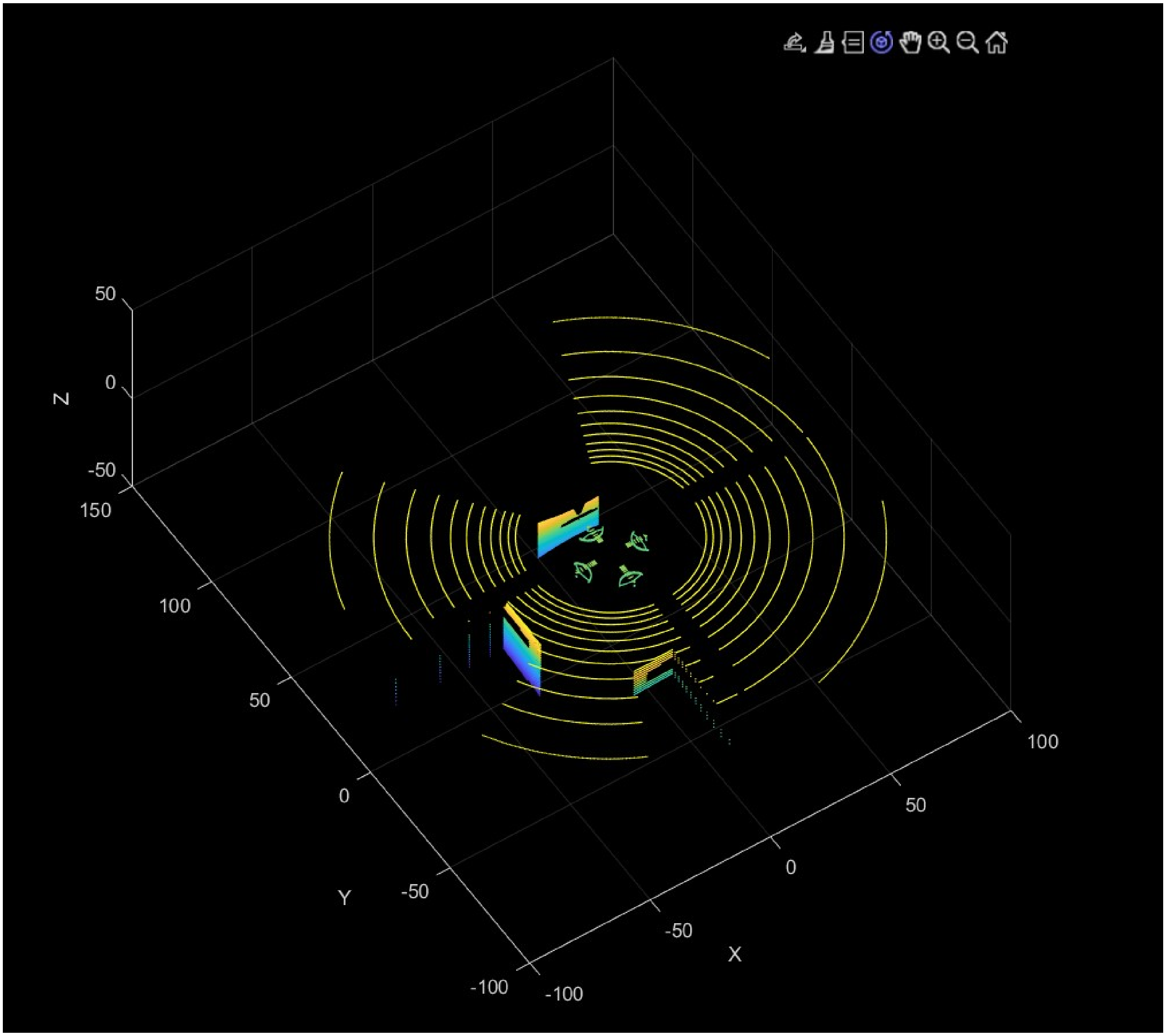


50
0
-50
Z
150
100
50
0
-50
-100
Y
100
80
60
40
20
0
-20
-40
-60
-80
-100
X

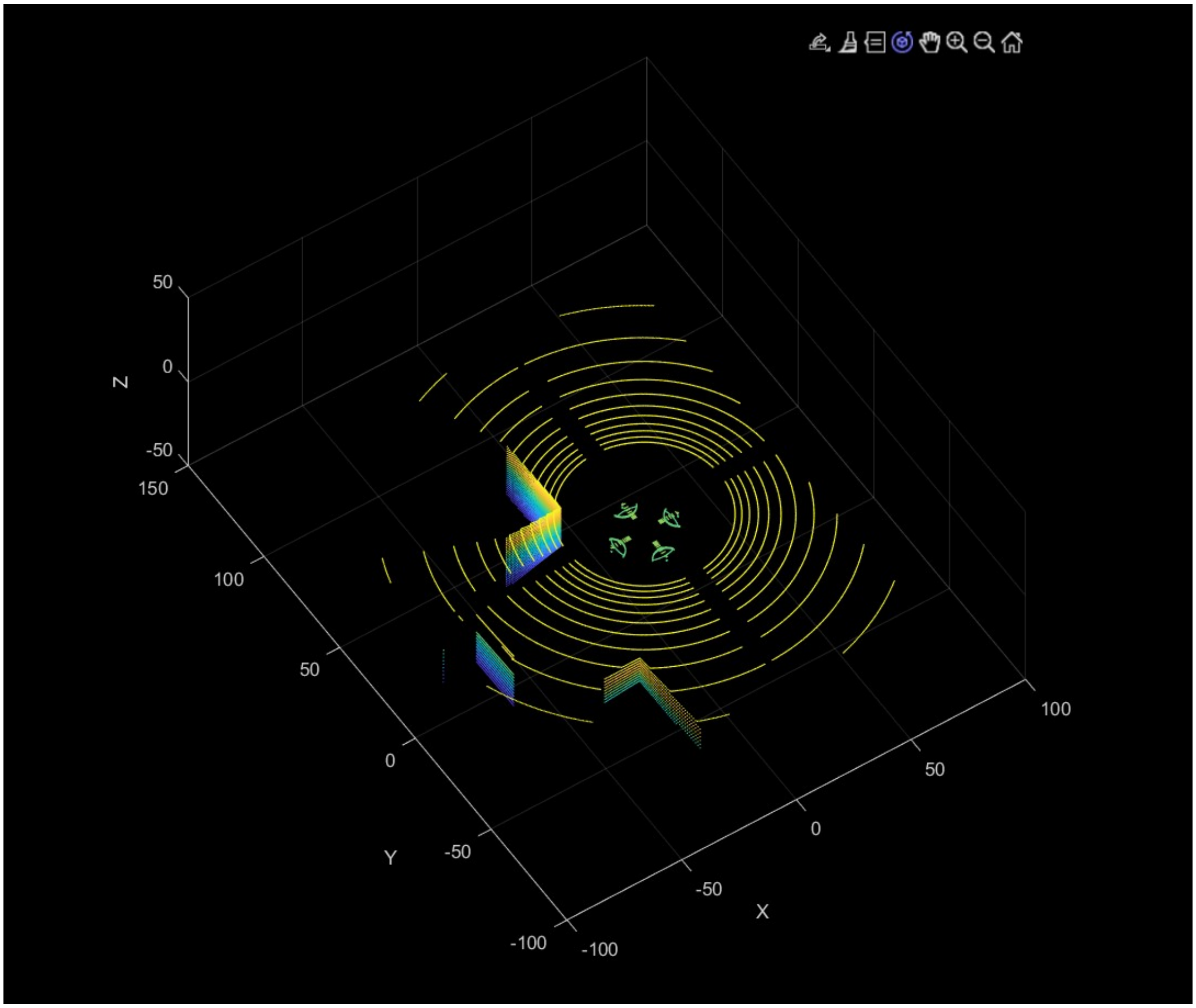

50
0
Z
-50
150
100
50
0
Y
-50
-100
-100
-50
X
0
50
100


50
0
Z
-50
150
100
50
0
Y
-50
-100
-100
-50
0
X
50
100


50
0
Z
-50
150
100
50
0
Y
-50
-100
-100
-50
X
0
50
100



50
0
Z
-50
150
100
50
0
Y
-50
-100
-100
-50
0
X
50
100


50
0
Z
-50
150
100
50
0
Y
-50
-100
-100
-50
0
X
50
100

Steps for UAV simulation in ROS Gazebo:

1. First installing the ros and primary UAV packages.

Install ROS and primary packages

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'

$ sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116
$ sudo apt-get update
```

2.Installing ROS indigo using the code:

Install ROS Indigo

```
$ sudo apt-get -y install ros-indigo-desktop-full

$ sudo rosdep init
$ rosdep update
```

3. Setting up the environment variables:

Setup environment variables

```
$ sudo sh -c 'echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc'

$ source ~/.bashrc
```

4. Getting rosintall and other dependencies:

Get rosinstall and some additional dependencies

```
$ sudo apt-get -y install python-rosinstall

$ sudo apt-get -y installros-indigo-octomap-msgs

$ sudo apt-get -y install ros-indigo-joy

$ sudo apt-get -y install ros-indigo-geodesy

$ sudo apt-get -y install ros-indigo-octomap-ros

$ sudo apt-get -y install unzip
```

5. Installing Gazebo 2 into the ubuntu server

Installed Gazebo 2.x,if it's not installed

```
$ sudo apt-get install libsdformat1

$ sudo apt-get install gazebo2
```

6. Creating and setting up the catkin workspace

Create the catkin workspace

$ WORKSPACE=~/ros/catkin_ws

$ source ~/.bashrc

Set up the workspace

$ mkdir -p $WORKSPACE/src

$ cd $WORKSPACE/src

$ catkin_init_workspace

$ cd $WORKSPACE

$ catkin_make

$ sh -c "echo 'source $WORKSPACE/devel/setup.bash' >> ~/.bashrc"

7. installing the mav comm package, glog catkin package , simple catkin package, and the glog catkin package

Install the mav comm package

$ cd $WORKSPACE/src

$ git clone https://github.com/PX4/mav_comm.git

Install the glog catkin package

$ cd $WORKSPACE/src

$ git clone https://github.com/ethz-asl/glog_catkin.git

Install the catkin simple package

$ cd $WORKSPACE/src

$ git clone https://github.com/catkin/catkin_simple.git

8. Now we have to install quadrotor package and the rotor package

Install the ROS Quadrotor Simulator package

$ cd $WORKSPACE/src

$ git clone https://github.com/wilselby/ROS_quadrotor_simulator

Install rotors simulator

RotorS is a UAV gazebo simulator developed by the Autonomous Systems Laboratory at ETH Zurich.

$ cd $WORKSPACE/src

$ git clone https://github.com/wilselby/rotors_simulator

$ cd rotors_simulator

9.Now compiling the workspace, installing xbox controller and verifying the modal

Compile the workspace

$ cd $WORKSPACE

$ source devel/setup.bash

$ catkin_make

Install Xbox 360 Controller

Install the integrated Ubuntu Xbox driver

$ sudo apt-add-repository ppa:rael-gc/ubuntu-xboxdrv

$ sudo apt-get update && sudo apt-get install ubuntu-xboxdrv

Model Verification

$ cd /tmp/

$ check_urdf kit_c.urdf

$ urdf_to_graphiz kit_c.urdf

$ roslaunch quad_description quad_rviz.launch

If it’s showing an error then, code

```
$ export ROS_HOSTNAME=localhost
$ export ROS_MASTER_URI=http://localhost:11311
```

The quad_world launch file is executed with the following command which displays the quadrotor model in Gazebo.

$ roslaunch quad_gazebo quad_world.launch

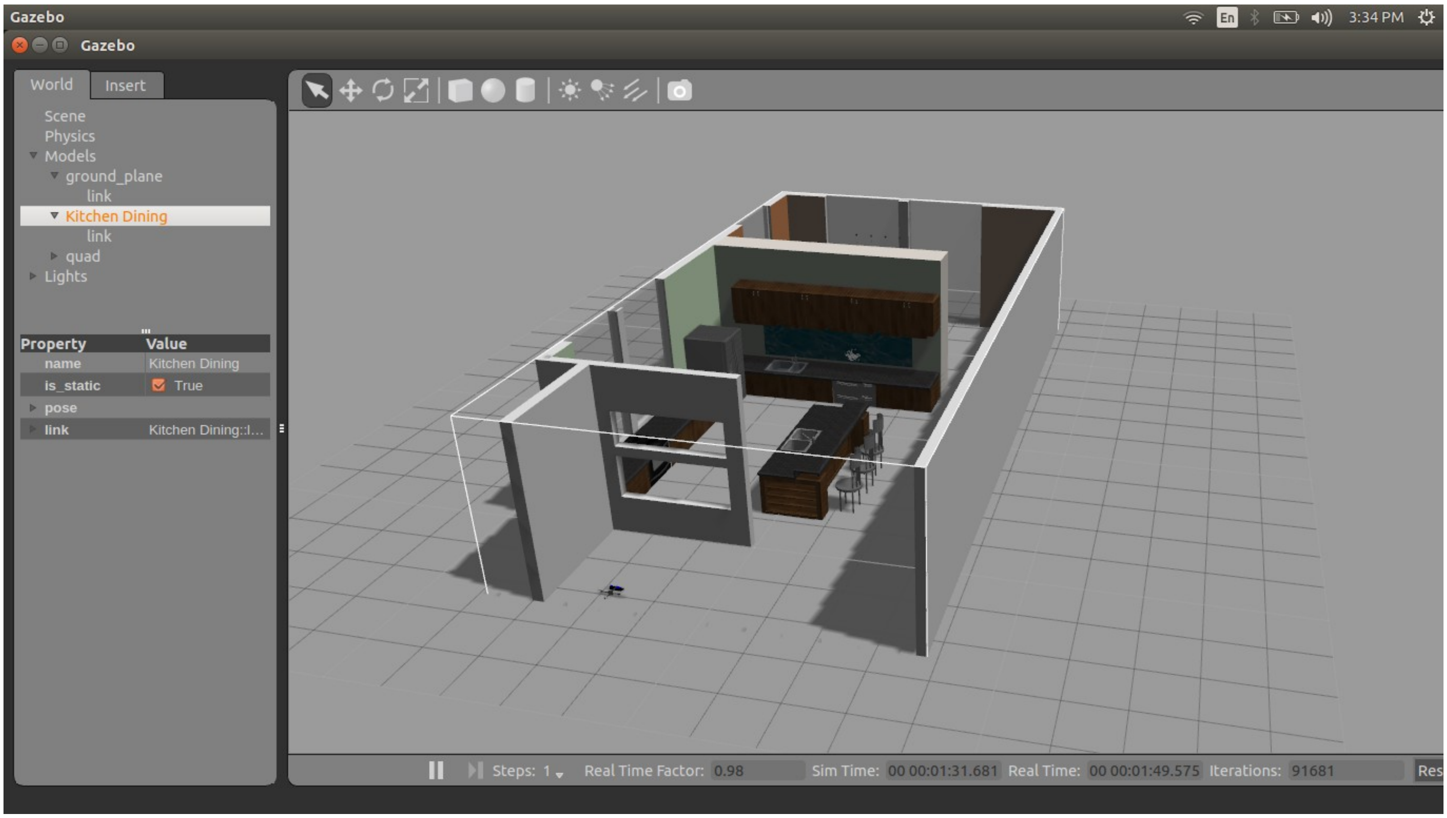
10. Now 3-D mapping and navigation

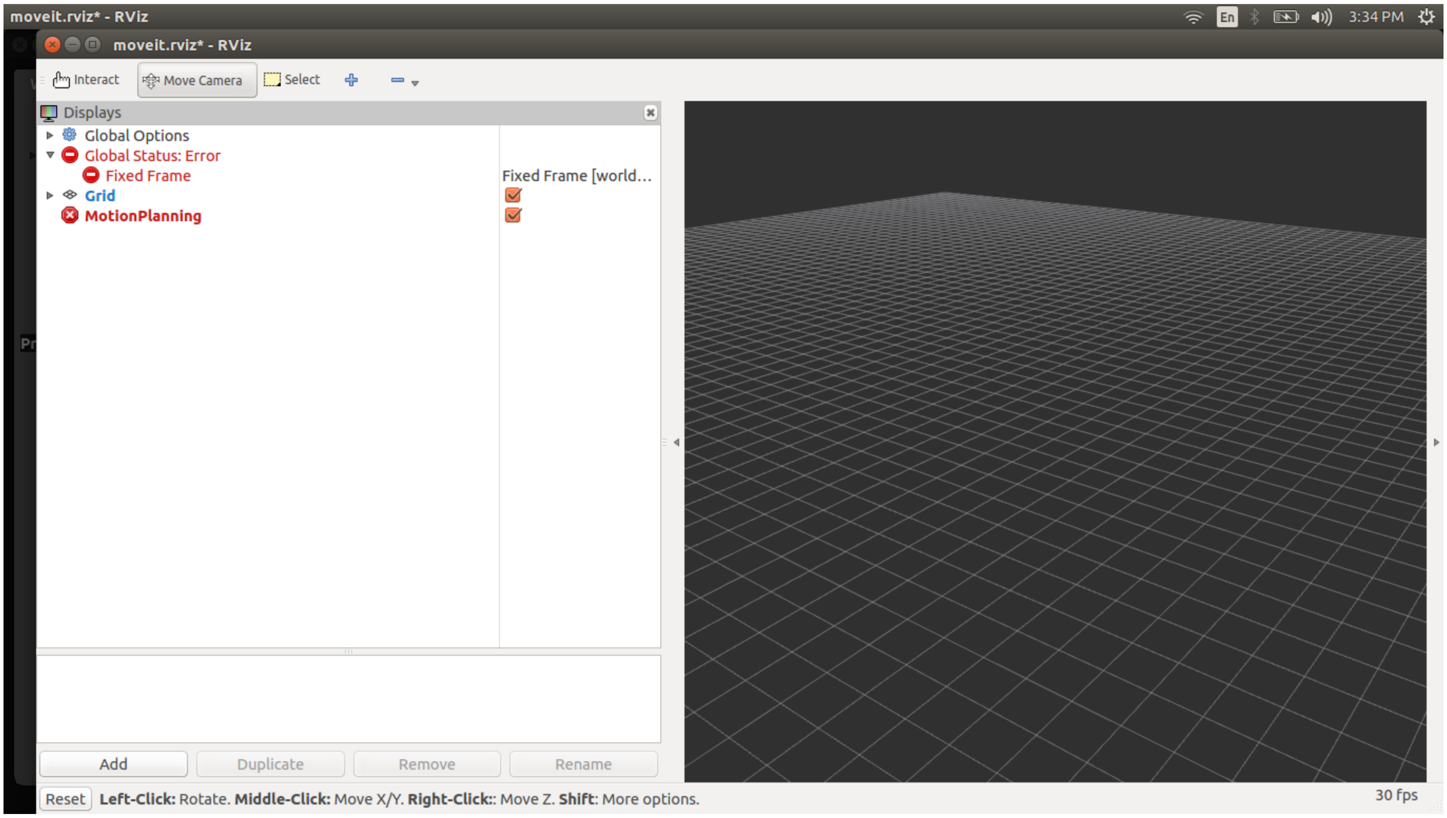
3D Mapping and Navigation

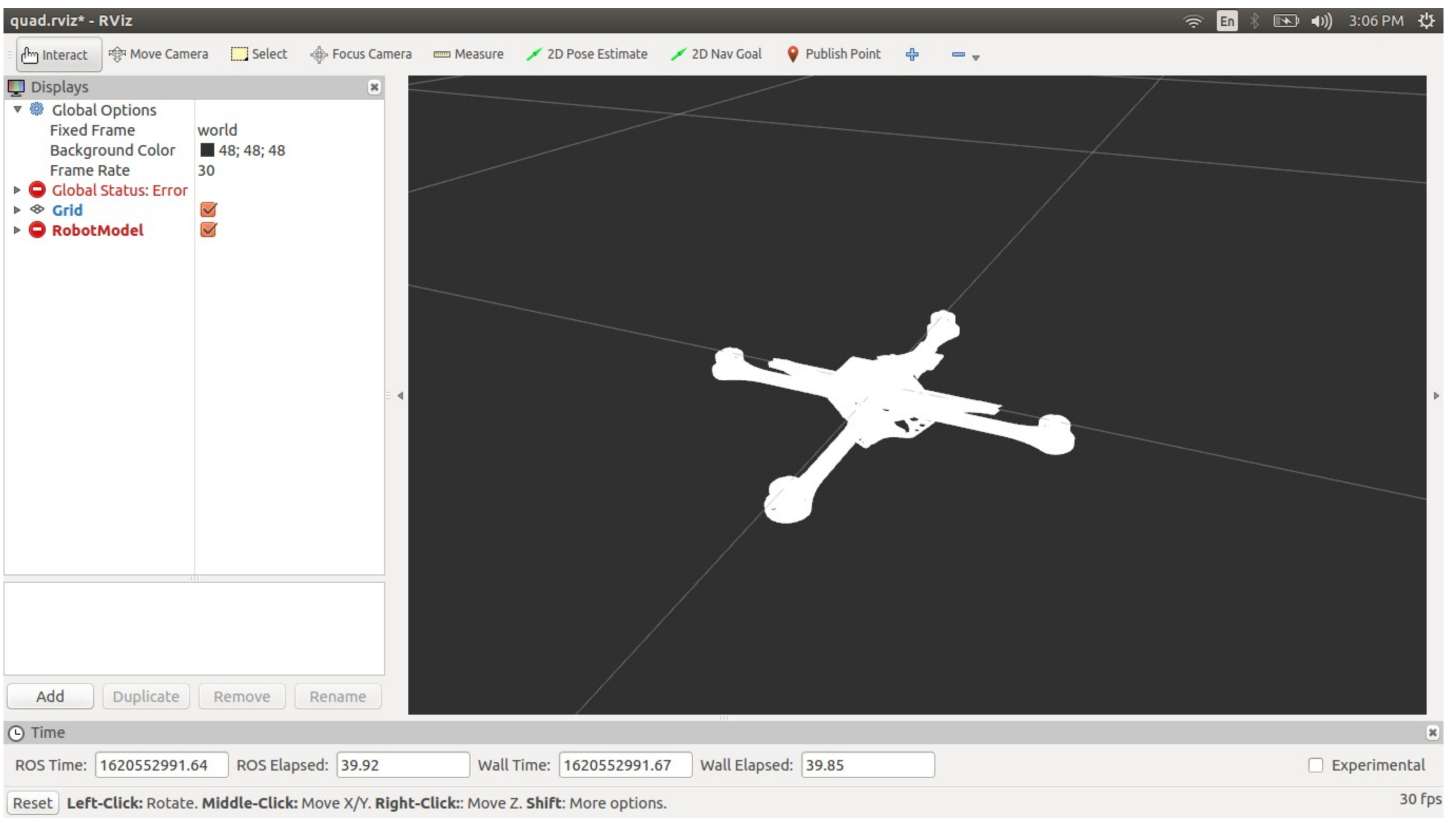
```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```

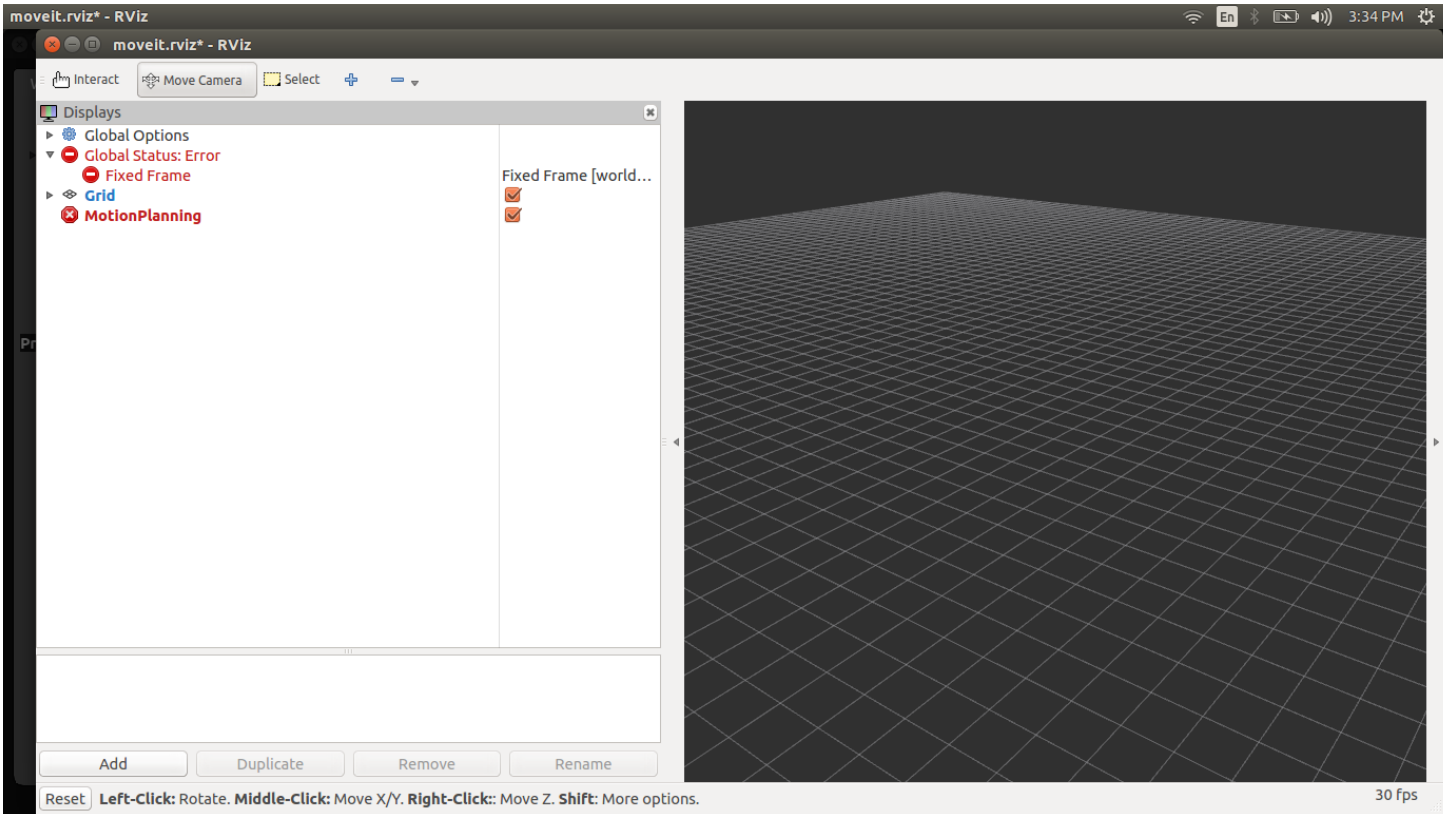
The 3D navigation simulation can be launched with the following command

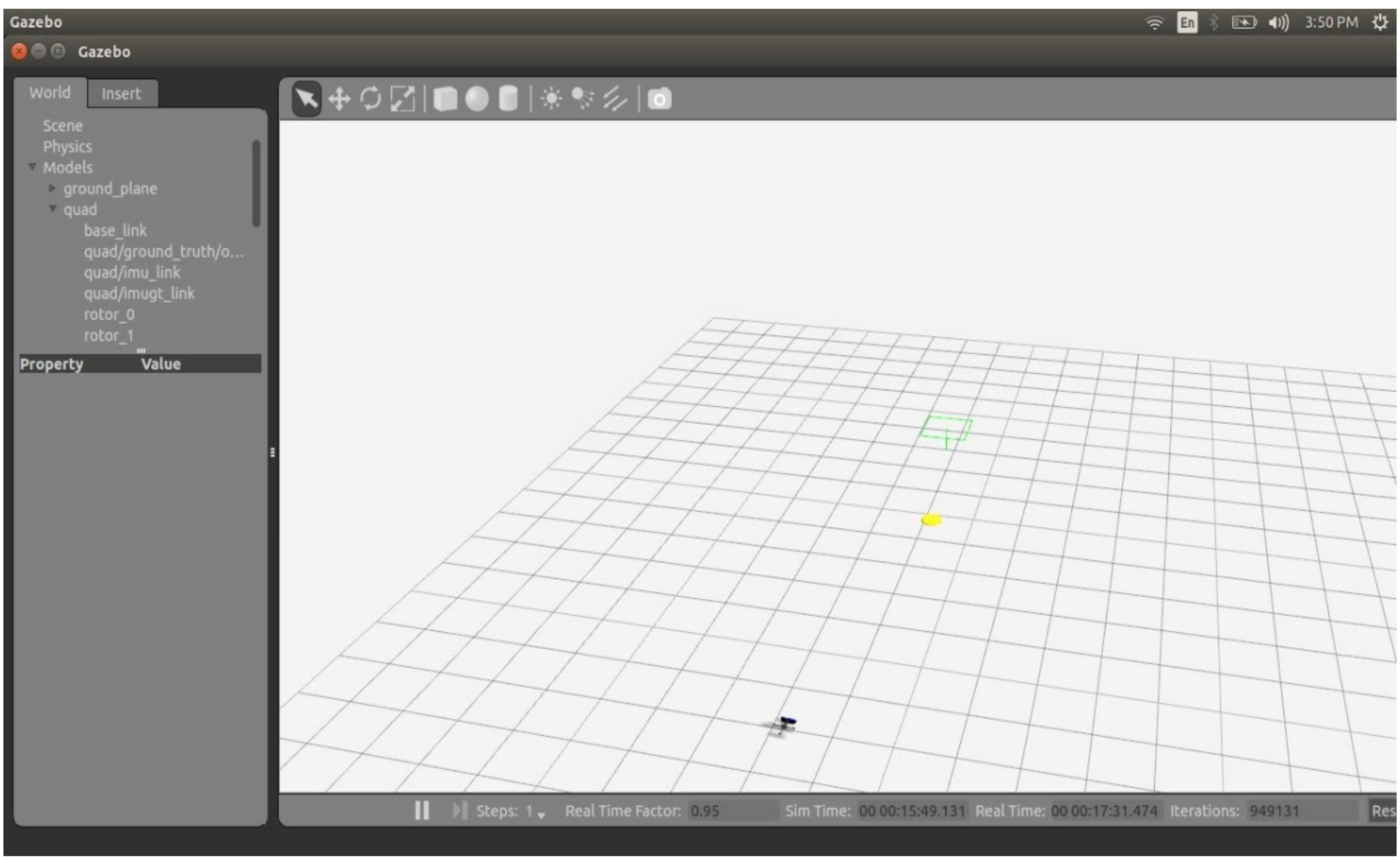
```
$ roslaunch quad_3dnav quad_3dnav.launch
```

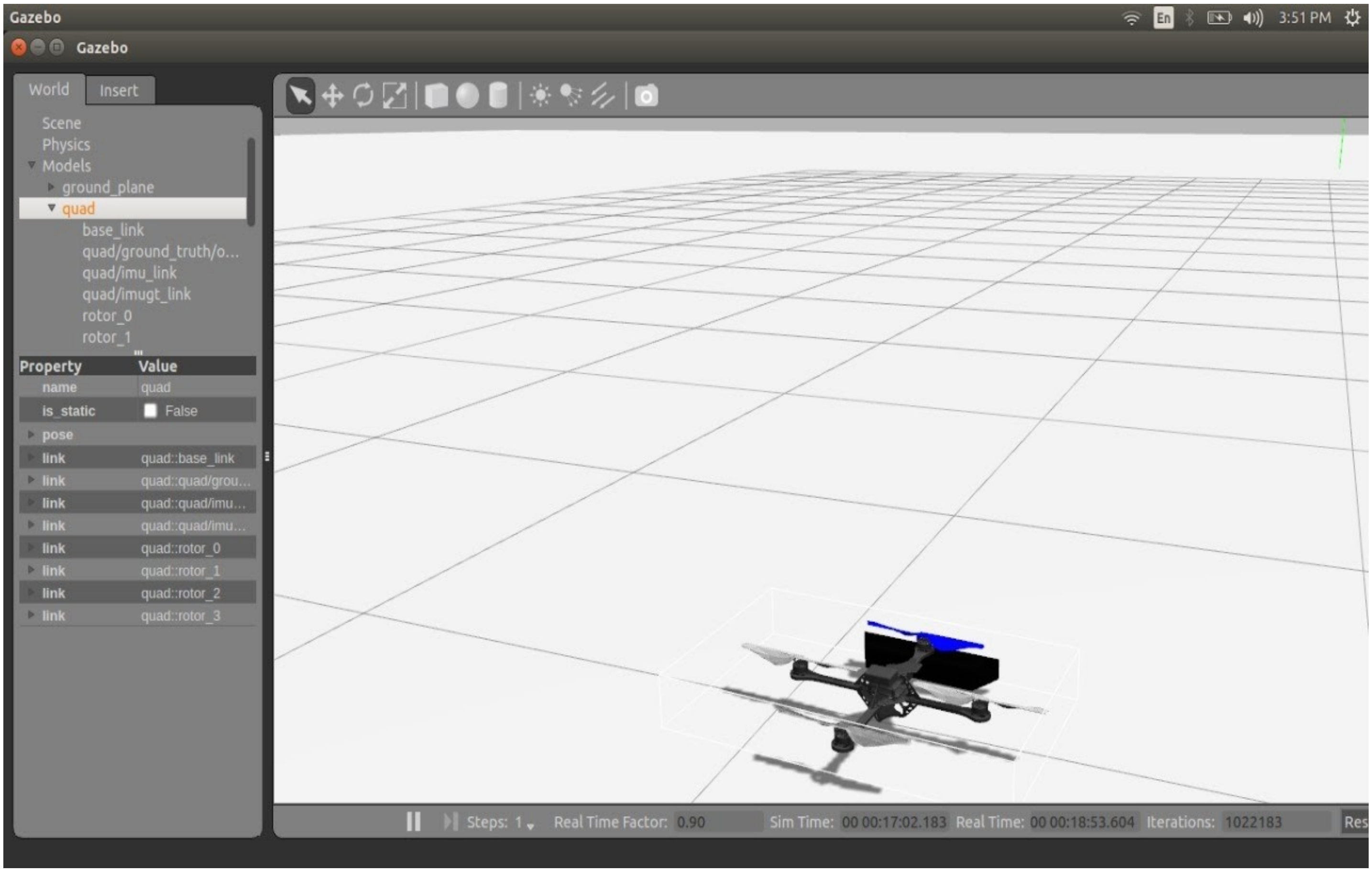



moveit.rviz* - RViz
Interact
Move Camera
Select
Displays
Global Options
Global Status: Error
Fixed Frame
Grid
MotionPlanning
Fixed Frame [world...
Add
Duplicate
Remove
Rename
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Move Z. Shift: More options.
30 fps


quad.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
world
Background Color
48; 48; 48
Frame Rate
30
Global Status: Error
Grid
RobotModel
Add
Duplicate
Remove
Rename
Time
ROS Time: 1620552991.64
ROS Elapsed: 39.92
Wall Time: 1620552991.67
Wall Elapsed: 39.85
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Move Z. Shift: More options.
30 fps


moveit.rviz* - RViz
3:34 PM
moveit.rviz* - RViz
Interact
Move Camera
Select
Displays
Global Options
Global Status: Error
Fixed Frame
Fixed Frame [world...
Grid
MotionPlanning
Add
Duplicate
Remove
Rename
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Move Z. Shift: More options.
30 fps


Gazebo
3:50 PM
Gazebo
World
Insert
Scene
Physics
Models
ground_plane
quad
base_link
quad/ground_truth/o...
quad/imu_link
quad/imugt_link
rotor_0
rotor_1
Property
Value
Steps: 1
Real Time Factor: 0.95
Sim Time: 00 00:15:49.131
Real Time: 00 00:17:31.474
Iterations: 949131


Gazebo
3:51 PM
Gazebo
World
Insert
Scene
Physics
Models
ground_plane
quad
base_link
quad/ground_truth/o...
quad/imu_link
quad/imugt_link
rotor_0
rotor_1
Property
Value
name
quad
is_static
False
pose
link
quad::base_link
link
quad::quad/grou...
link
quad::quad/imu...
link
quad::quad/imu...
link
quad::rotor_0
link
quad::rotor_1
link
quad::rotor_2
link
quad::rotor_3
Steps: 1
Real Time Factor: 0.90
Sim Time: 00 00:17:02.183
Real Time: 00 00:18:53.604
Iterations: 1022183