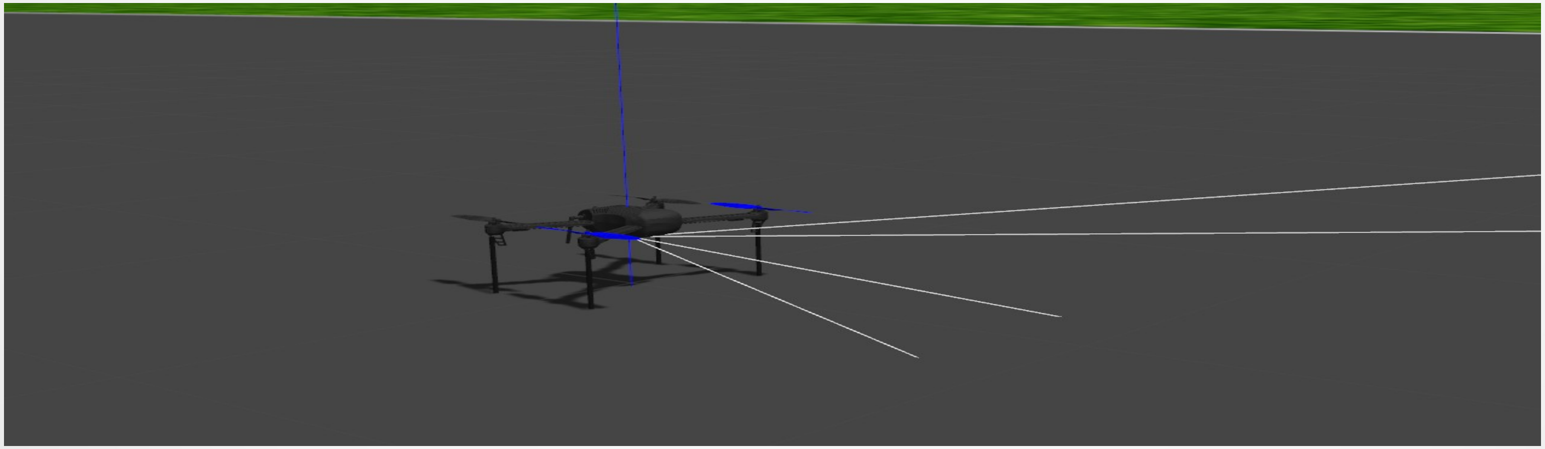
DroneRush

- DRONE SIMULATION USING ROS AND GAZEBO



- DGMD E-17
- Aboubacar Bah

Content

- Goal
- Hardware and Software
- Source Code
- Simple Program
- Obstacle Avoidance
- Challenges
- Future Work

Goal

- Drone Control
- Fly drone to waypoints
- Adding Sensors to Drone
- Obstacle Avoidance using Lidar for Drones

Hardware and Software

- Asus Q524UQ
- Ubuntu 20.04.2 LTS
- Python 3.8.5
- Catkin Workspace
- Matplotlib, opencv, scipy, …
- Ardupilot
- Mavros and Mavlink
- Gazebo11
- ROS Noetic

Source Code

https://github.com/Boubisto/DGMD-E-17

http://wiki.ros.org/ROS/Tutorials/
http://gazebosim.org/tutorials
https://ardupilot.org/ardupilot/index.html
http://users.isr.ist.utl.pt/~mir/pub/ObstacleAvoidance.pdf

Simple Program

- Program to fly a drone
- C++

- //initialize ros
- ros::init(argc, argv, "gnc_node");
- ros::NodeHandle gnc_node;
-
- //initialize control publisher/subscribers
- init_publisher_subscriber(gnc_node);
-
- // wait for FCU connection
- wait4connect();
-
- //wait for used to switch to mode GUIDED
- wait4start();
-
- //create local reference frame
- initialize_local_frame();
-
- //request takeoff
- takeoff(3);

Simple Program

Demo

Simple Program

- Program to fly a drone to waypoints
- C++

```
std::vector<gnc_api_waypoint> waypointList;
gnc_api_waypoint nextWayPoint;
nextWayPoint.x = 0;
nextWayPoint.y = 0;
nextWayPoint.z = 3;
nextWayPoint.psi = 0;
waypointList.push_back(nextWayPoint);
nextWayPoint.x = 5;
nextWayPoint.y = 0;
nextWayPoint.z = 3;
nextWayPoint.psi = -90;
```

Simple Program

Demo

Obstacle Avoidance

- Obstacle Avoidance Program with Ardupilot
- C++

- Take off and control Loop

```
//initialize control publisher/subscribers
init_publisher_subscriber(n);

// wait for FCU connection
wait4connect();

//wait for user to switch to mode GUIDED
wait4start();

//create local reference frame
initialize_local_frame();

//request takeoff
takeoff(2);

set_destination(0,0,2,0);

ros::Rate rate(2.0);
int counter = 0;
while(ros::ok())
{

ros::spinOnce();
rate.sleep();

}
```

- Parse the Lidar Data

```
sensor_msgs::LaserScan current_2D_scan;
current_2D_scan = *msg;
float avoidance_vector_x = 0;
float avoidance_vector_y = 0;
bool avoid = false;

for(int i=1; i<current_2D_scan.ranges.size(); i++)
{
float d0 = 3;
float k = 0.5;

if(current_2D_scan.ranges[i] < d0 && current_2D_scan.ranges[i] > .35)
{
avoid = true;
float x = cos(current_2D_scan.angle_increment*i);
float y = sin(current_2D_scan.angle_increment*i);
float U = -.5*k*pow(((1/current_2D_scan.ranges[i]) - (1/d0)), 2);

avoidance_vector_x = avoidance_vector_x + x*U;
avoidance_vector_y = avoidance_vector_y + y*U;

}
}
```

Obstacle Avoidance

Demo

Challenges and Future Work

- Software Compatibility
- Parameters tuning

- Autonomous Drone Simulation
- Controlling Multiple Drones
- Utilize Mission Planner

End

- Thank you for your time!