DRDO DGRE's Vision Based Obstacle Avoidance Drone

Inter IIT Tech Meet 9.0

Team name: M1_DRD_17

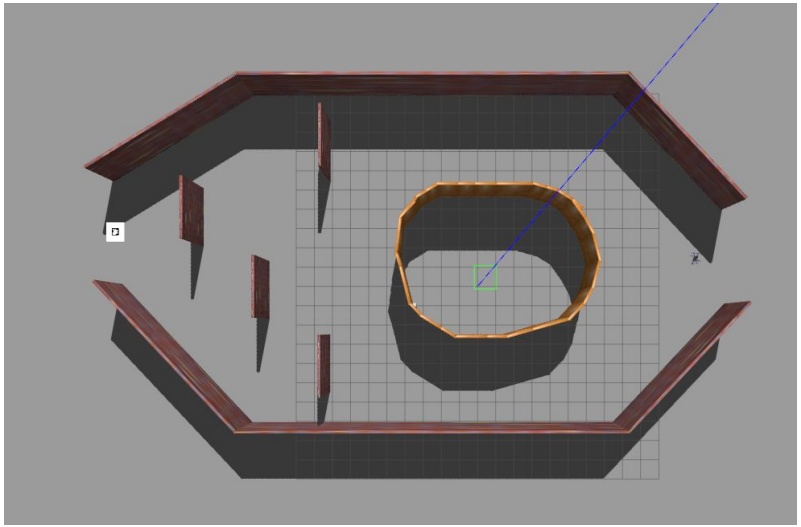
Problem Statement

The problem statement consists of a drone and a maze filled with several obstacles.

There is a marker at the other end of the maze.

The drone must fly autonomously from the starting to ending point.

It must use its sensors to avoid obstacles.



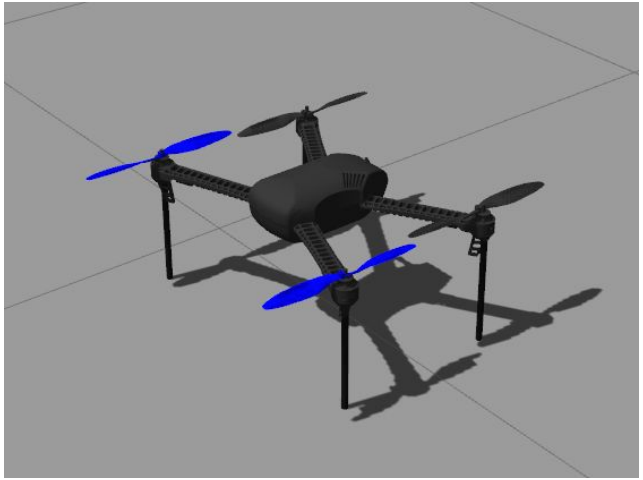
Problem Statement

Sensors:

- Forward facing depth camera
- Downward facing rgb camera

Software:

- Gazebo for simulation
- Ardupilot for flying the drone
- ROS for inter-process communication



Approach

Approach

The problem statement consists of 2 main parts:

- Fly through the maze, avoiding all obstacles
- Land on Aruco marker 0

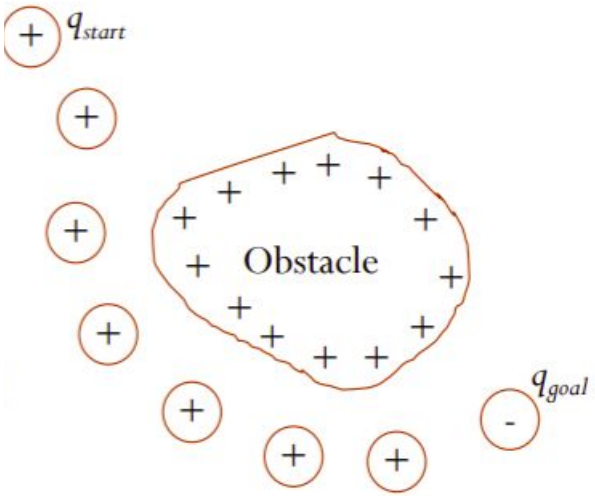
The 2 parts have been tackled separately. The drone flies through the maze, avoiding obstacles.

When the aruco marker with 0 id is detected, the drone flies towards it and lands.

Flying and avoiding obstacles

For navigation we are using potential field based approach:

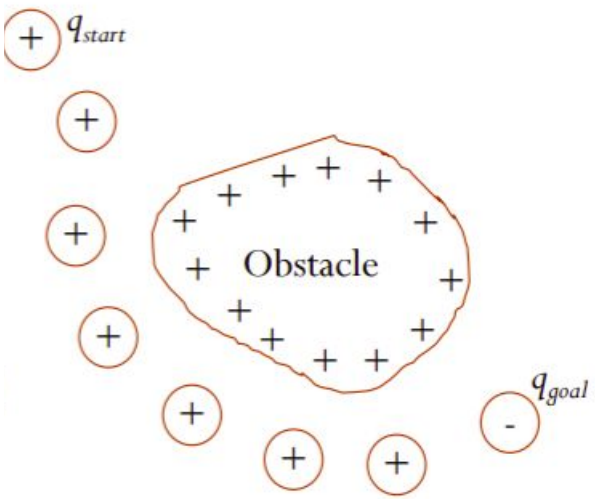
1. We are subscribing to the point cloud from the front depth camera.
2. Drone is assumed to have a positive “charge”. All the points are assigned a positive charge as well.
3. A large negative charge is put in front. This incentivizes the drone to move forward.



Flying and avoiding obstacles

4. Since we should not exceed 5m altitude, a large positive charge is put there.

5. After calculating the resultant force because of all these charges, the drone is given a velocity setpoint proportional to the forces.

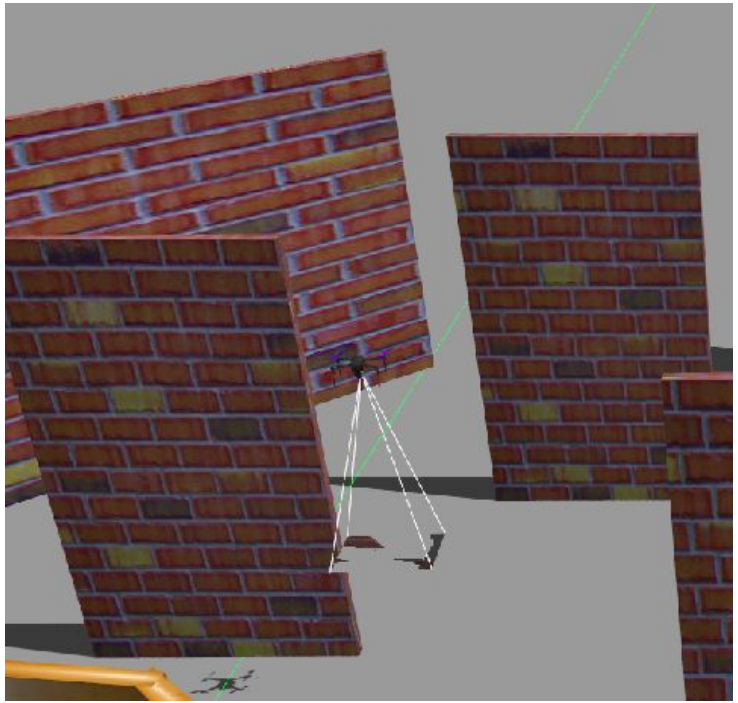


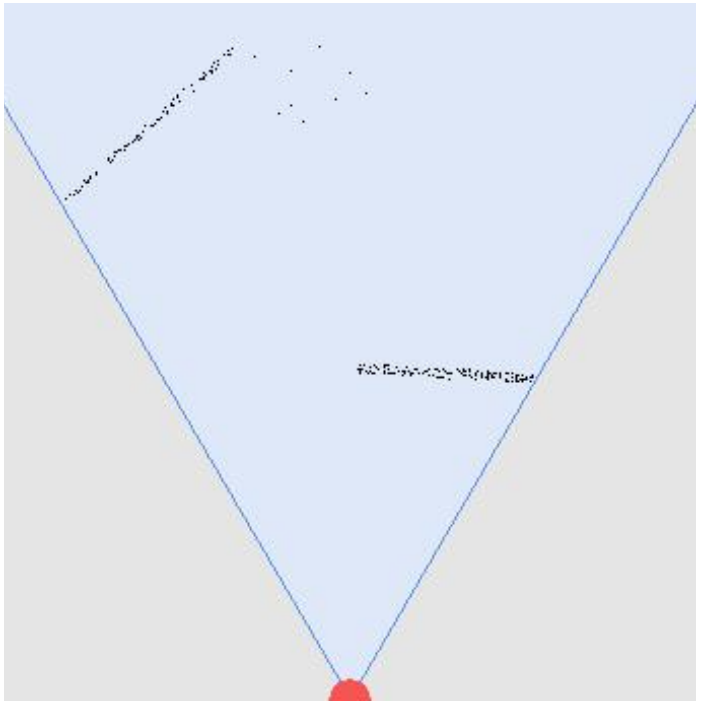
Some other points...

The entire point cloud contains 640x480 = 307200 points. We are randomly sampling 1000 points from the point cloud to increase computation speed. For most cases this is sufficient.

We are using an inverse linear law to calculate force on the drone.

$$\vec{F} = F\hat{i} - D\hat{k} - \sum_{p} \frac{k}{|r|^2}\vec{r}$$



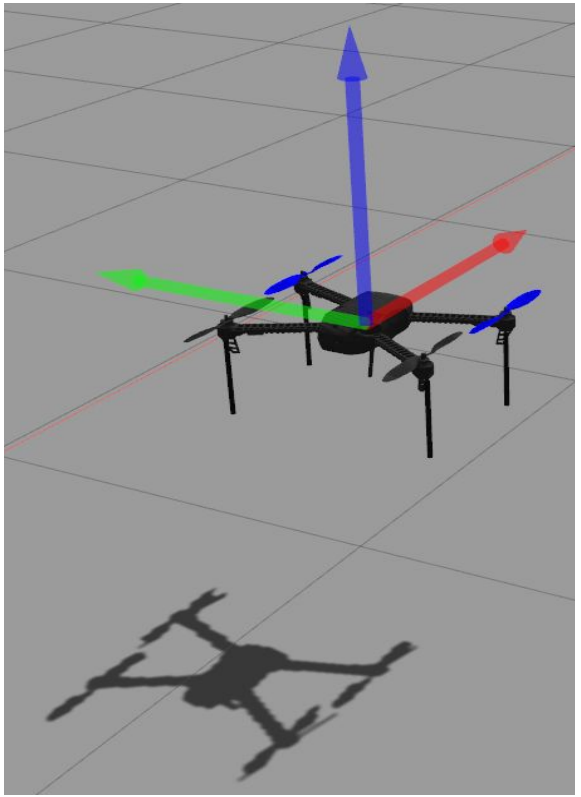


Low Level Control

Once we have the force components, we set a velocity setpoint proportional to the force.

We do this by publishing to the topic `/mavros/setpoint_velocity/cmd_vel_unstamped`

This topic accepts a message of type `geometry_msgs/Twist`



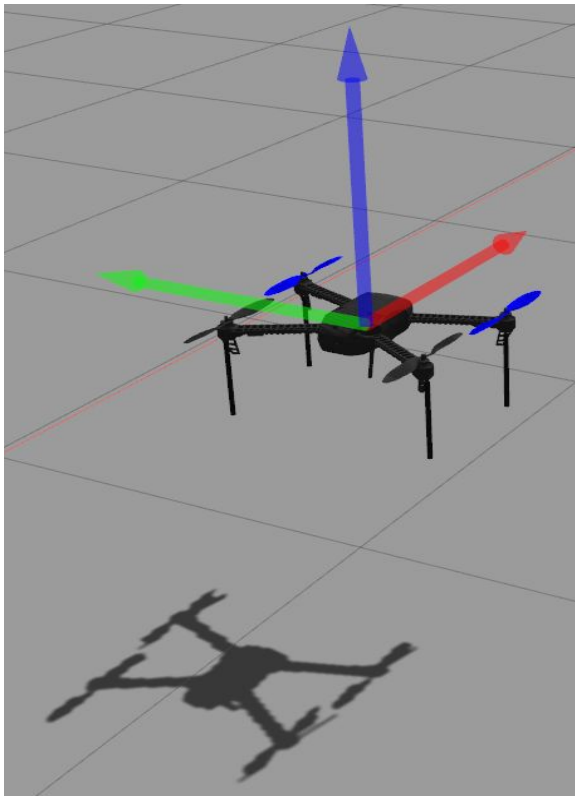
Low Level Control

msg = Twist()

msg.linear.x = $k_x F\mathbf{i}$

msg.angular.z = $k_y F\mathbf{j}$

msg.linear.z = $k_z F\mathbf{k}$



Landing on Marker

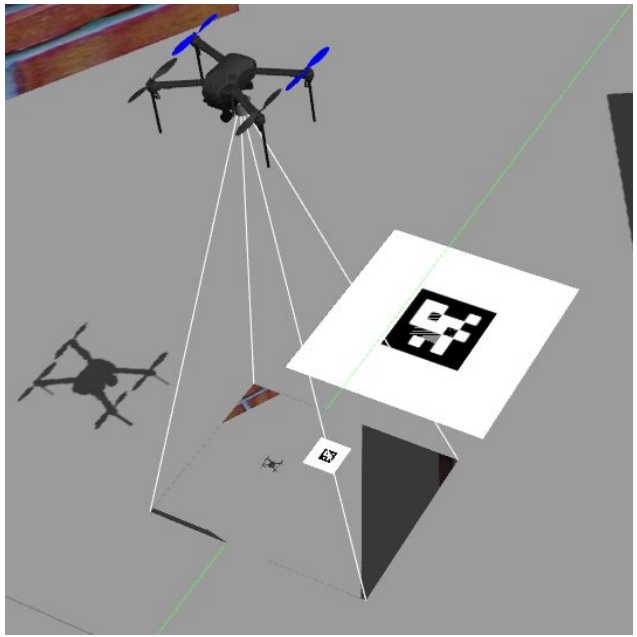
Aruco Marker Detection

We use OpenCV Aruco library to detect Aruco markers in the camera feed.

If a marker with ID = 0 is found, we switch states and navigate relative to the marker.

We find the distance and direction to the center of the marker and apply a proportional force.

Once over the marker, we land.



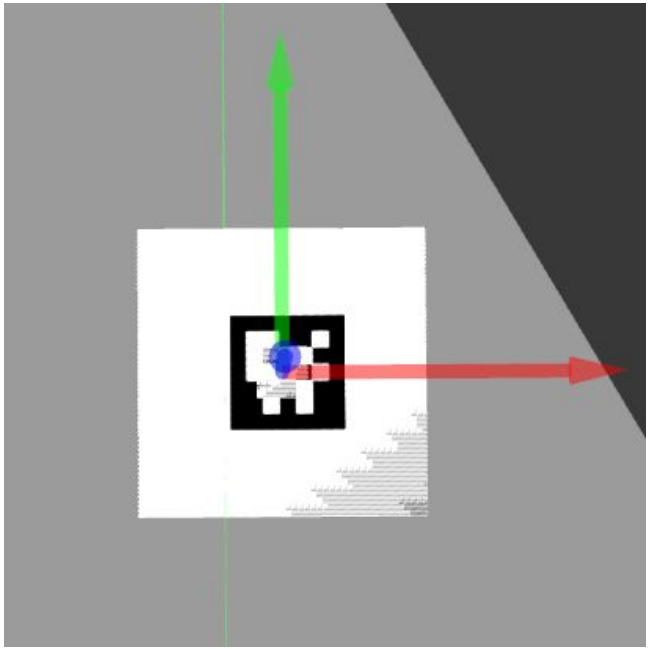
Low Level Control

After calculating the force we issue a velocity setpoint in a similar manner.

msg = Twist()

msg.linear.x = $k_x F\mathbf{i}$

msg.linear.y = $k_y F\mathbf{j}$



Pros & Cons

Pros

Very simple algorithm

Requires very little computational resources

Requires almost no libraries

Performs exceptionally well, specially when only horizontal movement is required.

Cons

There is a possibility to get stuck, if sum of forces is zero.

There is a possibility that the drone might turn around and go back to starting position.

Owing to large blind spots in its vision, the drone might crash. This is specially possible if the drone needs to traverse in z-direction.

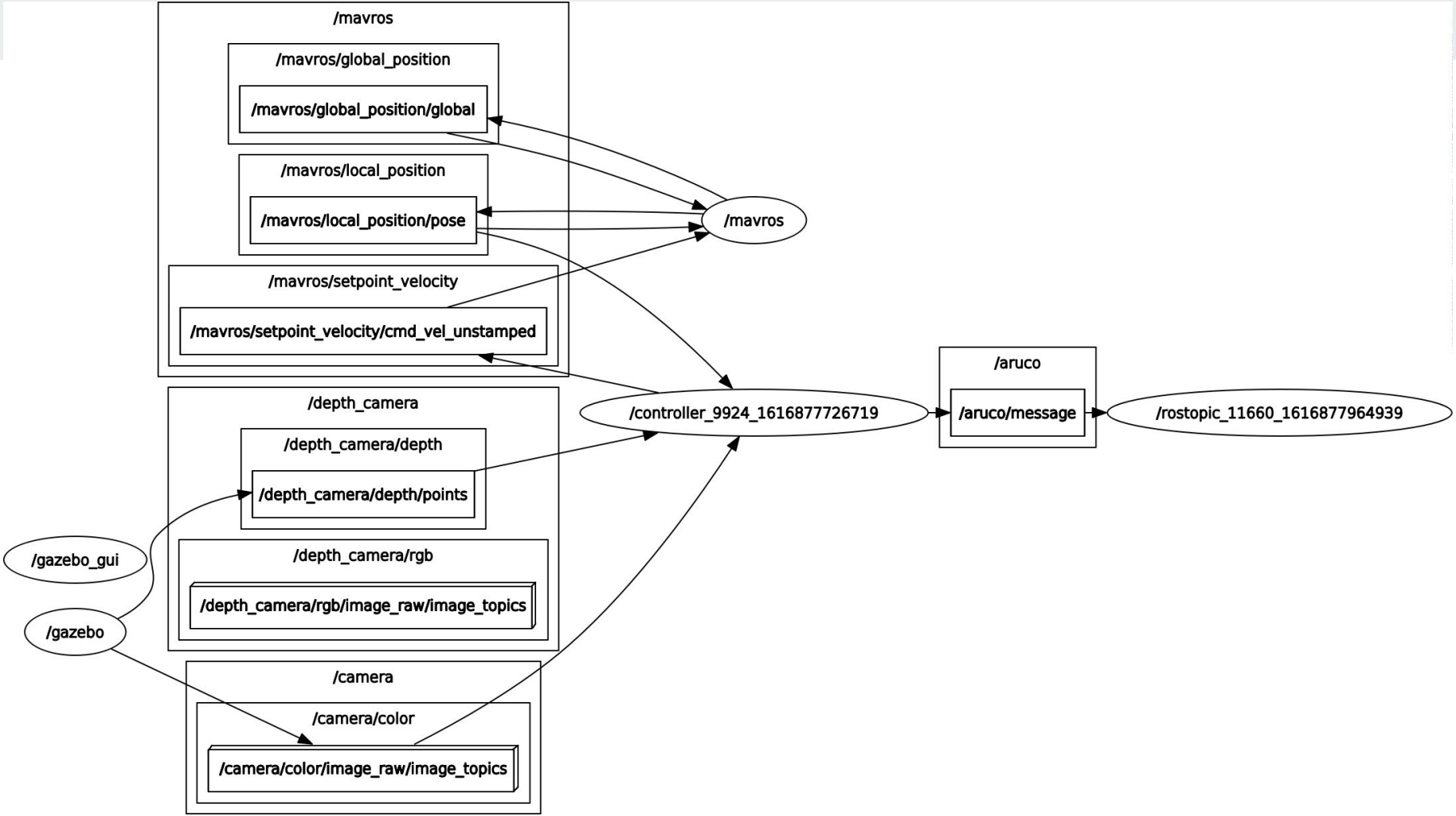
Improvements

An occupancy grid can be constructed from the vision data. This will enable the drone to “remember” what it saw previously. This will reduce the effects of blind spots.

Proper path planning techniques (like RRTConnect) can be used. This will prevent crashes and enables the drone to take safer and faster paths.

Navigation information like the heading can be used to make the drone preferably go towards the end instead of turning back to the starting point.

Appendix


/mavros
/mavros/global_position
/mavros/global_position/global
/mavros/local_position
/mavros/local_position/pose
/mavros
/mavros/setpoint_velocity
/mavros/setpoint_velocity/cmd_vel_unstamped
/aruco
/depth_camera
/controller_9924_1616877726719
/aruco/message
/rostopic_11660_1616877964939
/depth_camera/depth
/depth_camera/depth/points
/gazebo_gui
/depth_camera/rgb
/depth_camera/rgb/image_raw/image_topics
/gazebo
/camera
/camera/color
/camera/color/image_raw/image_topics

States of the drone

The drone uses a state system to ensure that correct instructions run at correct times.

The states of the drone are:

- TAKEOFF
- FLYING
- MARKER_DET
- LANDING
- LANDED

Takeoff and Landing

Takeoff:

1. Flight mode is set to GUIDED.
2. ARM command is issued.
3. TAKEOFF command is issued. Takeoff altitude is 3m.

Landing:

1. LAND command is issued.
2. DISARM command is issued.

Thank You