I. Results

A. *Implementation of RRT and RRT* in 2D environment*

We first implemented RRT and RRT* in 2D using pygame. The following are the plots obtained:

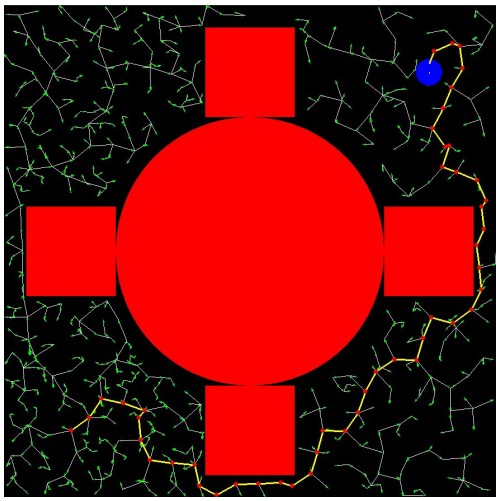

Figure 1. RRT implementation in 2D

The simulation time for RRT is about 2 seconds. We can clearly see that the path obtained is not optimal but the we get to the solution faster.

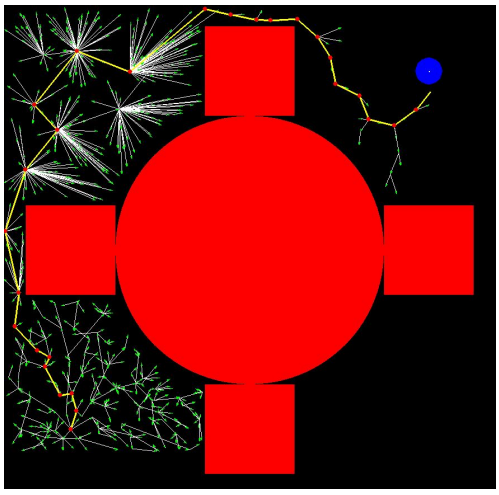

Figure 2. RRT* implementation in 2D

The simulation time for RRT* is about 113 seconds. We can clearly see that the path obtained is more optimized than RRT but the time taken to reach the solution is significantly increased. The nodes exploration is directed towards the goal position.

B. *Implementation of RRT in 3D environment*

As earlier mentioned, for simulation in 3D matplotlib package was used. The following are the results of RRT and RRT* for a static 3D environment.

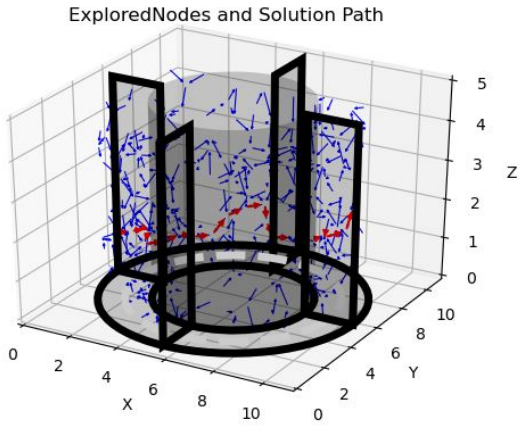

Figure 3. RRT implementation in 3D

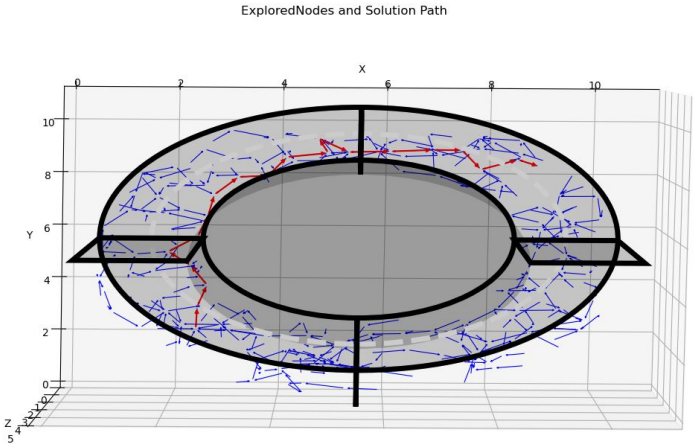

Figure 4. RRT implementation in 3D top view

In addition to this, we have created an animation of drone using matplotlib which follows the solution trajectory.
Currently, the path generated by RRT is not smoothened. By implementing the gradient descent algorithm, the solution path is smoothened. The following is the drone trajectory after the path of the drone is smoothened. As RRT generates new path each time, the path shown in the below figures may be little different from the above graphs but difference in the smoothness of the path is clearly seen.

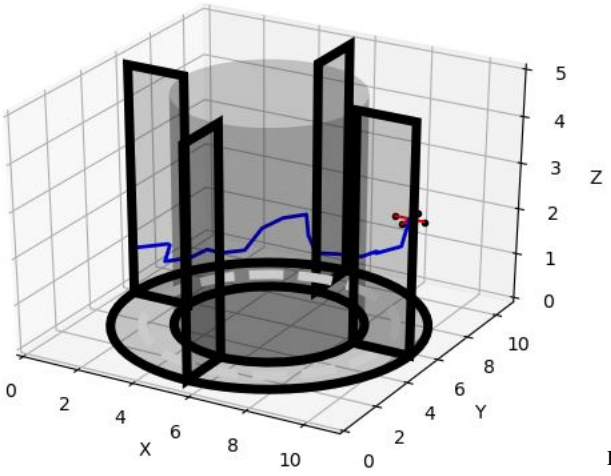

Figure 5. RRT implementation drone animation

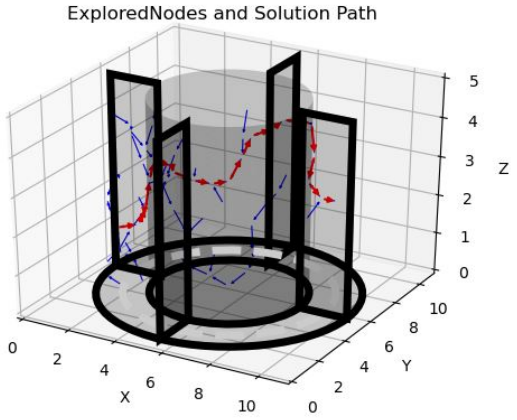

Figure 6. RRT implementation in 3D with smoothing

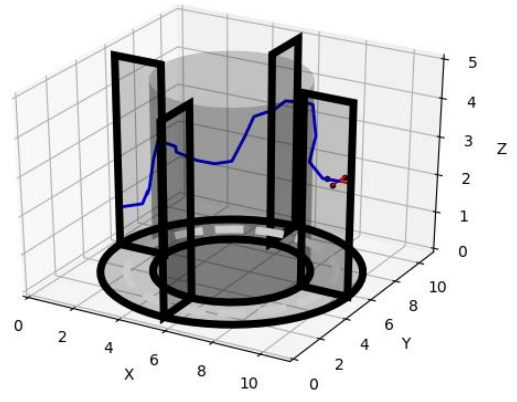

Figure 7. RRT implementation in 3D smooth drone path

C. *Implementation of RRT* in 3D environment*

Similar to RRT, we also implemented RRT* algorithm for a static 3D environment. Here are the results:

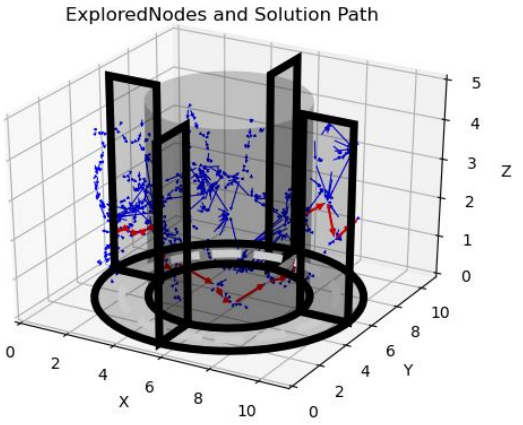

Figure 8. RRT* implementation in 3D

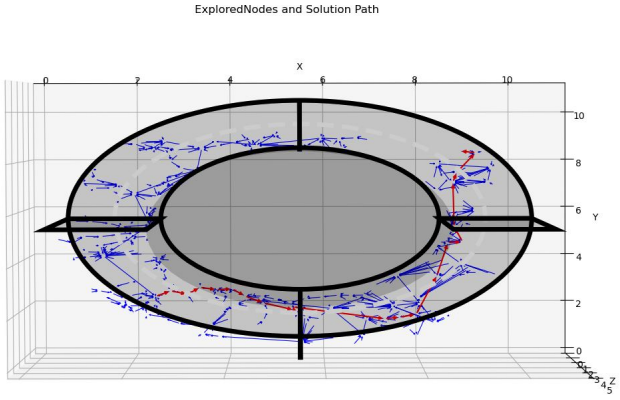

Figure 9. RRT* implementation in 3D top view

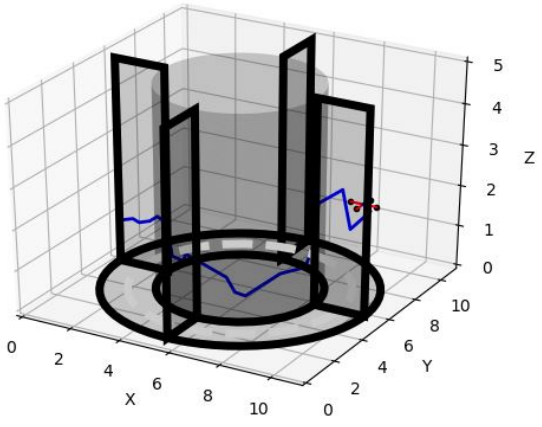

Figure 10. RRT* implementation drone animation

After performing the smoothing, the following are the results:

D. *Dynamic Environment path planning results:*

We have developed RRT and RRT* algorithms to adapt to dynamic changes in the environment. Below sequence

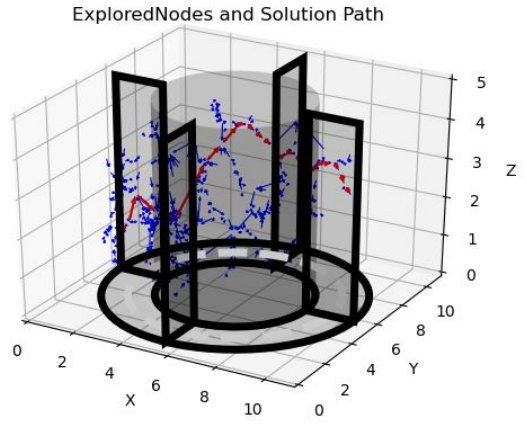

Figure 11. RRT* implementation in 3D with smoothing

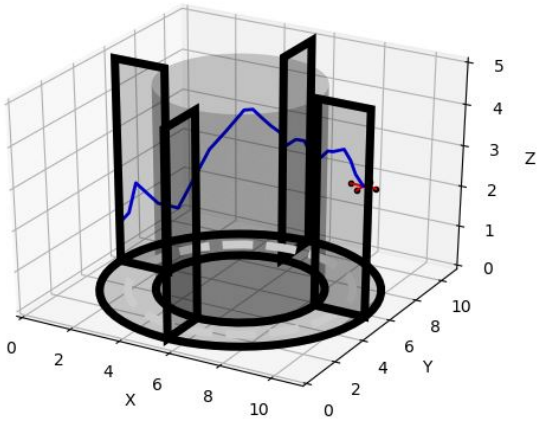

Figure 12. RRT* implementation in 3D smooth drone path

of images explains the pipeline with the output results. The following are the results obtained:

1) First RRT/RRT* generates the initial solution path considering the initial position of the obstacles(drones).

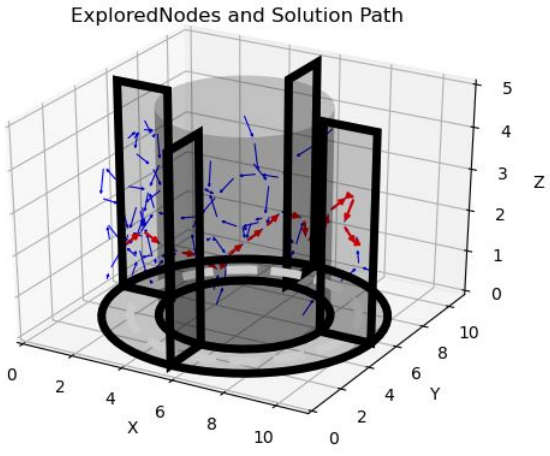

Figure 13. Generating initial solution path

2) The drone animation below shows the drone starting to the follow the initial generated solution path.

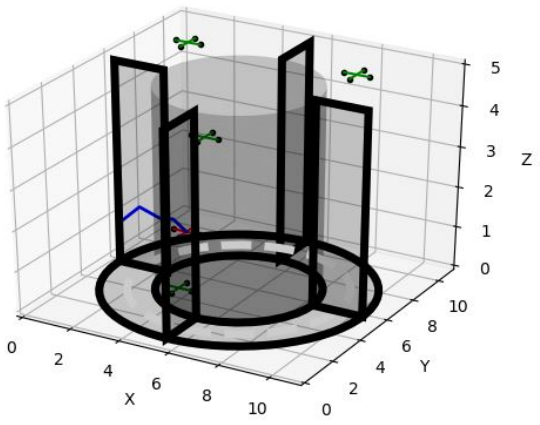

Figure 14. Drone following the initial solution path

3) In the above picture, one of the obstacle drone was obstructing as well as close to the drone. Hence, replanning of the path occurs.

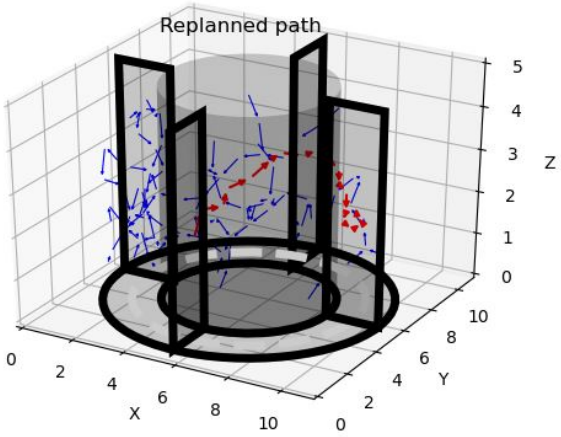

Figure 15. Replanning of drone path from current position

4) Now the figure below shows the animation of the drone following the replanned path

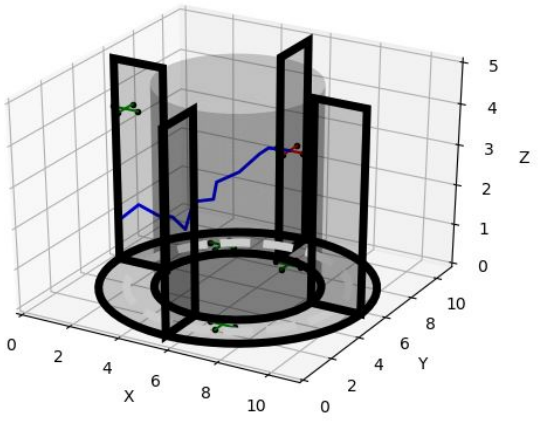

Figure 16. Drone following replanned path