Task 1.2 – Problem Statement

Implementing Path Planning to Avoid Obstacles

From the tutorials, you have learned

- Translating from the WhyCon frame of reference to the V-REP frame of reference.
- Path planning using the OMPL plugin.

Scene Description:

Load the given scene *task1_2_hb.ttt* in V-REP simulator. The scene looks as shown in Figure 1:

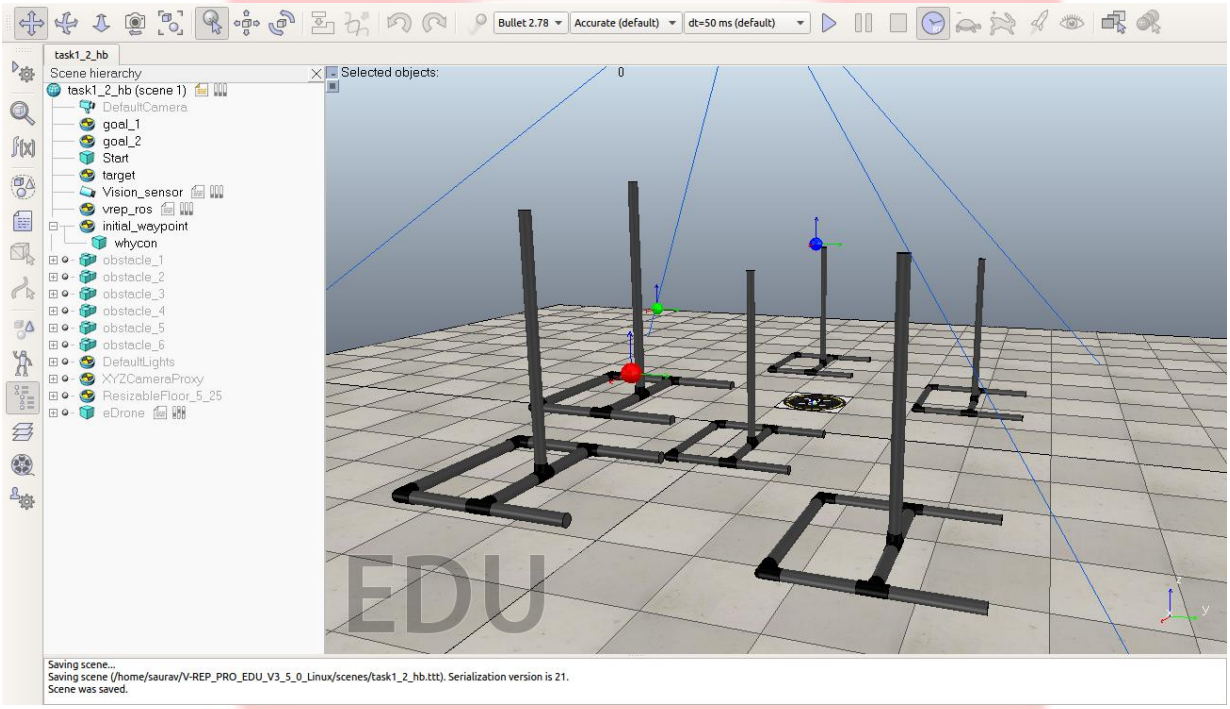

Figure 1 : task1_2_hb.ttt

Following are the various objects in the scene:

eDrone : This the the Drone model provided to you. It subscribes to topic /drone_command

Obstacles : There are six Obstacles, named obstacle_1 to obstacle_6.

initial waypoint : There is one dummy named initial_waypoint, it has a WhyCon marker associated with it.

Goals : There are two dummies indicating points to be reached namely goal_1 and goal_2.

Vision_sensor : This gives an image within the blue region with a resolution 1280 x 1024.

vrep_ros : This is the dummy which contains the path planning script and is responsible for sending waypoints to the e-Drone via ROS.

Start : This is the pad from which the e-Drone takes off.

Problem Statement:

The e-Drone should go the the intial_waypoint, then plan the path to goal_1 and traverse the planned path, followed by planning the path to goal_2, traversing this path, and return to the initial_waypoint and then disarm.

Procedure:

1. Go through, understand and perform the tutorials provided to you.
2. Make a duplicate of your "position_hold.py" Python script that you created in Task 1.1, and rename it as "**path_planning.py**".
3. Copy "**path_plan.launch**" to launch folder in the hungry_bird folder which you have cloned in the src folder of catkin_ws in the previous task i.e. to:
   `~/catkin_ws/src/hungry_bird/launch`
4. Initially, both the WhyCon markers i.e. of eDrone and initial_waypoint are not Renderable. Make the WhyCon marker at the initial_waypoint Renderable by changing its Scene Object Properties.
5. Run the following command and note down the coordinates of the WhyCon marker.

```
>> roslaunch hungry_bird path_plan.launch
```

6. Enter these coordinates as the initial destination in "path_planning.py".
7. Undo the changes you made in Step 4 i.e. make the initial_waypoint whycon marker unrenderable.
8. Make eDrone's WhyCon marker as Renderable by changing its Scene Object Properties.
9. Verify that only the eDrone's WhyCon marker is being detected by running the launch file again.

```
>> roslaunch hungry_bird path_plan.launch
```

10. In this Task you will write code in two different locations.
    a) The first will be a Lua script in the dummy **vrep_ros** in the scene **task1_2_hb.ttt**, it should do the following tasks:
       i. Should compute and publish path using the OMPL Plugin.
       ii. Subscribe to a topic which is **published by path_planning.py** which will command it to start planning a path.
       iii. Visualize the path.
       iv. Send the path computed **back to path_planning.py** in the message type geometry_msgs/PoseArray
    b) The second location is the Python script named "path_planning.py" in the package hungry_bird, it should do the following tasks:
       i. Apply a control algorithm to the drone, similar to Task 1.1
       ii. Navigate to a waypoint which has been set as the destination.
       NOTE: Drone should consider the waypoint as reached only when the error in <u>all four dimensions</u>, x, y, z and yaw, is less than 0.5.
       iii. Subscribe to the path (i.e. list of waypoints) sent to it by V-REP.
       iv. Request a new set of path points once the last waypoint of a path is reached.
11. Launch your *path_plan.launch* file after running simulation in V-REP:

```
>> roslaunch hungry_bird path_plan.launch
```

12. Run your python script, *path_planning.py* by running the following command in another terminal:

```
>> rosrun hungry_bird path_planning.py
```

13. Ideally, the eDrone should arm, plan and traverse the required path in the mentioned order and finally disarm.

Points to remember:

- Make sure you remap the topics properly to detect markers.
- Run "*rosmsg show topic_type*" to see what is the message type of the corresponding topic.

Rules:

- The following simulation settings are default and **should not be changed**.
  - Dynamics engine : Bullet 2.78
  - Dynamics settings : Accurate (default)
  - Simulation time step : dt = 50 ms (default)
  - Real-time mode: Enabled
- You are not permitted to make changes in the scene to be submitted, i.e. task_1_2_hb.ttt. The only exception to this is adding Dummy objects, of which any number can be added.
- A complete run consists of the drone navigating in the following order: Start postion, initial_waypoint, goal_1, goal_2 -> initial_waypoint -> disarm
- Arming the drone will be considered the Start of Run.
- It is optional to plan the path from the Start position to the initial_waypoint at the beginning of the run.
- It is mandatory to plan the path from the initial_waypoint to goal_1, goal_1 to goal_2 and goal_2 back to initial_waypoint.
- Disarming the drone will be considered the End of Run.
- Please follow this link to a video to understand how your final output should look like.

Submission Instructions:

Follow the instructions below to submit your Task.

1. Bag File:

a) Launch your *path_plan.launch* file after running simulation in V-REP:

```
>> roslaunch hungry_bird path_plan.launch
```

b) Run your python script, *path_planning.py* by running the following command in another terminal:

```
>> rosrun hungry_bird path_planning.py
```

c) Run the rosbag command to record the /whycon/poses. The following command records the topics */whycon/poses* for a duration of 20s. **This command should be executed only after you start the simulation and the launch file but before running the python file.**

```
>> rosbag record whycon/poses --duration=20s --chunksize=10
```

NOTE: The duration parameter is set to 20s in the above command. This has to be set as per the estimated time your code takes to complete. For example, if it takes approximately 60 seconds for your code to complete the task, you must change the duration to 60 seconds. Maximum allowed duration is **240s ONLY.**

d) Compress the bag file

```
>> rosbag compress -j bag_file_name.bag
```

e) This will compress the *bag file* and now the size will be below 5mb approximately. Look the size of the bag file in its properties to distinguish between the original and the compressed bag file.

f) Rename the **compressed bag file** as **<team_id>_task_1_2.bag** . For eg: if your Team ID is HB#105, rename it as 105_task_1_2.bag .

NOTE: Don't follow the same procedure as in the demo video(Instructions below) while recording bag file. Just run the simulator, launch the file and run the python script. Record the rosbag till it become stable at the destined position.

2. Python Code:

a) After completing your python script, path_planning.py, rename it as **<team_id>_task_1_2.py**. For eg: if your Team ID is HB#105, rename it as 105_task_1_2.py .

3. V-REP Scene:

a) After completing your Task, rename the scene as **<team_id>_task_1_2.ttt**. For eg: if your Team ID is HB#105, rename it as 105_task_1_2.ttt .

Compress these files into a .zip file before uploading. Name the .zip file as your <team_id>_task_1_2.zip. For eg: if your Team ID is HB#105, then rename it as 105_task_1_2.zip

4. Video:

a) Upon verifying that your task is complete, record a maximum 5 minute video using a screen recorder like simplescreen recorder or kazam.

b) The video must be as follows:

i. Team Slide –All members' details in a slide.

ii. Any One member of the team, running the scripts and launch files in terminal.Output of V-REP and whycon image_out window on the PC screen captured clearly.

c) Please record the video as shown in this demo.

The video should not be edited in any manner. Teams uploading an edited video will be disqualified from the competition. e-Yantra reserves the rights to disqualify any team if any foul play is suspected.

Uploading video/s on YouTube:

● Upload a one-shot continuous video with the title eYRC#HB#Task1_2#<TeamID>
(For example: If your team ID is 1234 then, save it as eYRC#HB#Task1_2#1234)

● Please note that while uploading the video on YouTube select the privacy setting option as Unlisted as shown in Figure 2. You need to upload the video as instructed on the portal.

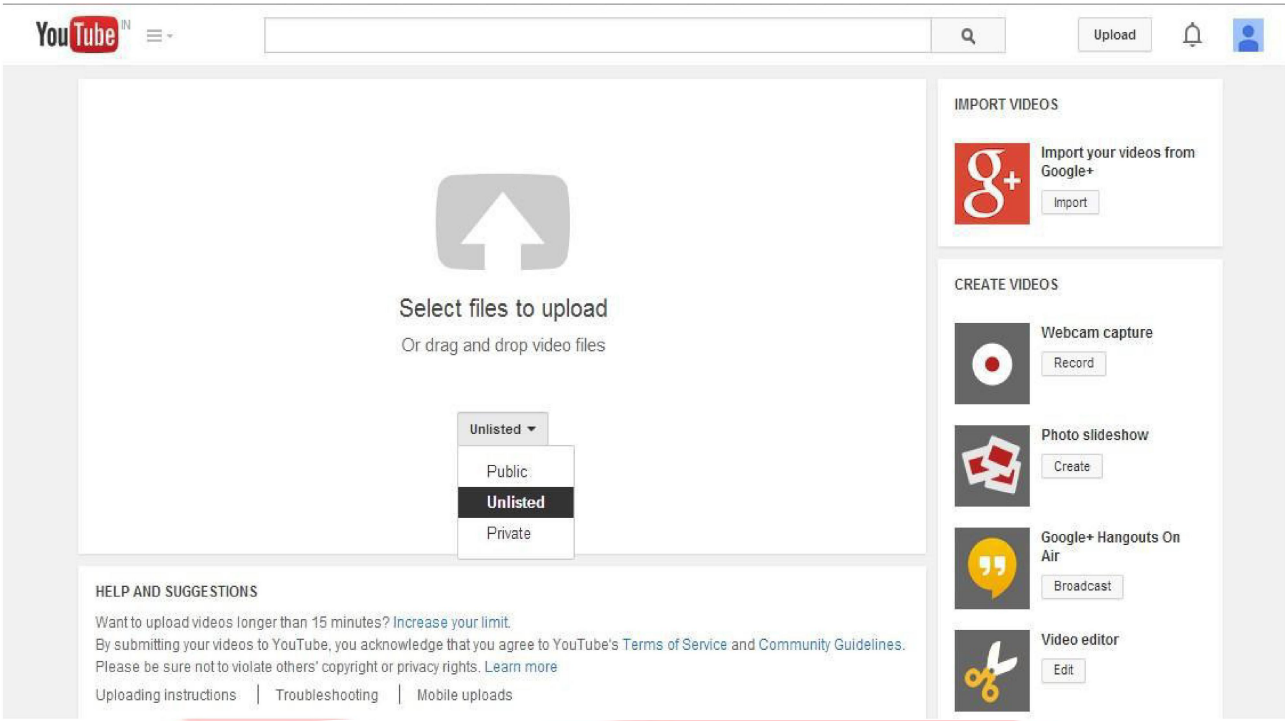

Figure 2: Uploading unlisted video on YouTube

NOTE: You must upload all of the following: (i) rosbag file, (ii) Python code (iii) V-REP scene (iv) Video in order to be evaluated. Please place files (i), (ii) and (iii) inside a .zip file before uploading. Name the .zip file as your <team_id>_task_1_2.zip. Please follow the naming convention strictly as specified in each step. For eg: if your Team ID is HB#105, then rename it as 105_task_1_2.zip

Instructions for uploading the folder will be provided on the portal.

NOTE: Deadline for the Task 1.2 submission is on 28th November 11:59 pm. You will incur penalty if submission is delayed. Details regarding penalty are provided on the portal.