Assignment-1: RRT

Team Name: Ananth and Sudarshan Team
Team Members
Ananth Rachakonda (2021702025)
Sudarshan S Harithas (2021701008)

1 Question 1: Non-holonomic RRT for differential drive

1.1 Kinematics of a Differential Drive

Motion primitives:

$$\theta_c(t_n) = \theta_c(t_{n-1}) + \int_{t_{n-1}}^{t_n} \omega_c(t)dt \tag{1a}$$

$$x_c(t_n) = x_c(t_{n-1}) + \int_{t_{n-1}}^{t_n} V_c(t)\cos(\theta_c(t))dt \tag{1b}$$

$$y_c(t_n) = y_c(t_{n-1}) + \int_{t_{n-1}}^{t_n} V_c(t)\sin(\theta_c(t))dt \tag{1c}$$

Left and right wheel kinematics:

$$R_c = V_c/\omega_c \tag{2a}$$

$$R_L = R_c - b \tag{2b}$$

$$R_R = R_c + b \tag{2c}$$

$$V_R = (R_c + b)\omega_c \tag{2d}$$

1.2 Explanation of the RRT algorithm for the non-holonomic differential drive

The RRT Algorithm for the non-holonomic differential drive is iterative and typically runs until a predefined number of iterations or until the goal condition is reached. In this work, it has been implemented to run until the goal condition is satisfied.

- **Random Sampling:** In each iteration, the following are sampled:
  - **Random Local Goal:** A random point in the robot's obstacle free configuration space - with an optional final goal bias.
  - **Random Controls:** A vector of random platform center linear and angular velocities from the domain of allowable linear and angular velocities.
- **Trajectory Computation:** Using the motion primitives provided in Eq. 1, we compute the robot's platform center trajectories for a finite time for each of the random sampled linear and angular velocities while making sure that the computed trajectories are collision free.
- **Tree Propagation:** Propagate the tree in the direction of nearest (end point to the random point) goal biased collision free trajectory

1.3 Figures comparing trees in goal biased and unbiased RRT

The convergence of the tree towards the goal in one sense is ensured by the goal bias. However, with enough number of samples, the algorithm may converge to the final goal even without goal bias - without guarantees.

It can be observed in Fig. 1a vs Fig. 1b, and Fig. 2a vs Fig. 2b that the goal bias ensures the occurrence of random point closer to the final goal as the tree approaches the final goal. This results in the trajectories closer to the goal being less scattered (and more directed) than the trajectories away from the goal. This is achieved by dynamically changing the variance of the distribution. Variance is chosen to be directly proportional to the distance of the last node in the tree.

It can be observed that the distance between the trajectory of wheels becomes observably large sometimes. This is due to deviation from true value due to accumulated error during numerical integration.

1.4 Implementation Details

The holonomic UAV has been developed on Ubuntu 20.04 in a virtualenv and the non-holonomic differential drive is built on Windows in a conda virtual environment.

2 Question 2: Holonomic RRT for quadrotor UAV

2.1 Kinematics of UAV

$$a_x = (-\cos\phi \sin\theta \cos psi - \sin\phi \sin\psi)T/m \tag{3a}$$
$$a_y = (-\cos\phi \sin\theta \sin\psi + \sin\phi \cos\psi)T/m \tag{3b}$$
$$a_z = g - (\cos\phi \cos\theta)T/m \tag{3c}$$

2.2 Explanation for RRT for Holonomic UAV

- **Sampling:** Sample multiple values of input ϕ. θ and Thrust T and using equation 3 we obtain the corresponding values of acceleration.
- **State Propagation:** using the acceleration along each axis we propagate the state. Two pruning methods are included in state propagation, firstly we check if the state is above the ground (z¿0) and secondly we check if the state is collision free.
- **Local Goal** a random feasible point (point that's not in/on an obstacle) is chosen as the local goal. We perform a goal biased RRT.
- **Tree expansion** The point in the tree that is closest to the local goal is chosen for further expansion of the tree.
- **Convergence** once the tree expands sufficiently such that it is in a certain region of radius near the goal we terminate the search and recover the path and other control parameters.

2.3 Results

2.3.1 Environment 1

The trajectory generation and execution in environment 1 is give in Fig. 3a, Fig. 3b.

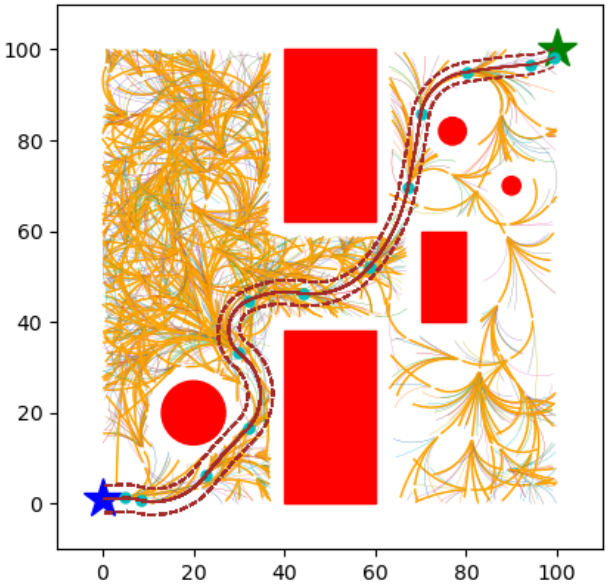
2.3.2 Environment 1: Change in start and goal locations

The trajectory generation and execution in a new start and goal location in environment 1 is give in Fig. 4a, Fig. 4b.

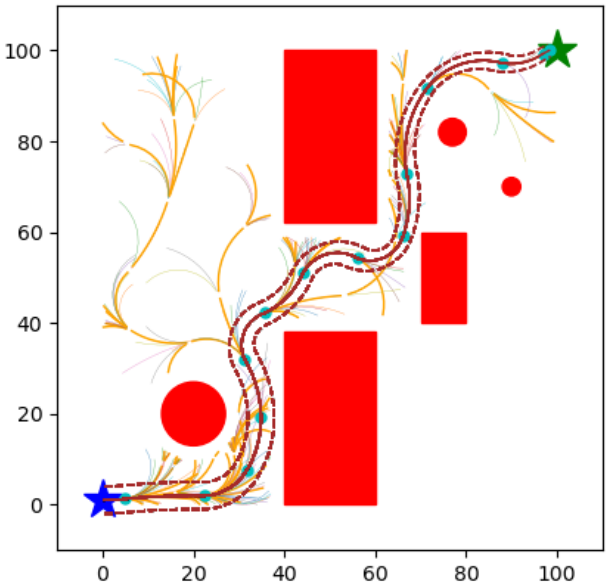
2.3.3 Environment 2

The trajectory generation and execution in custom environment 2 is given in Fig. 5b, Fig. 5b.

Please Find the Demonstration videos in this **Link**

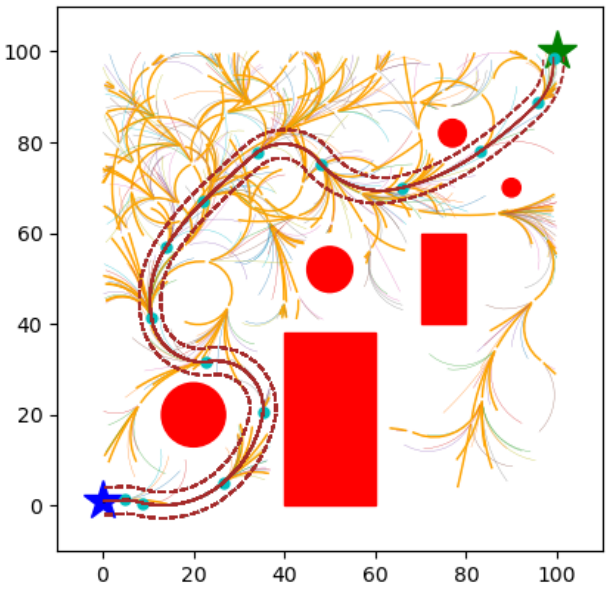


(a) An image of RRT in original environment without bias

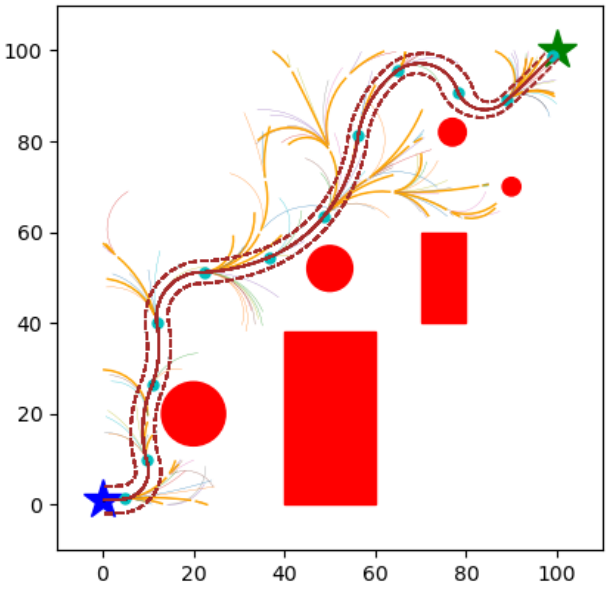


(b) An image of RRT in original environment without bias

Figure 1: Original Environment Results

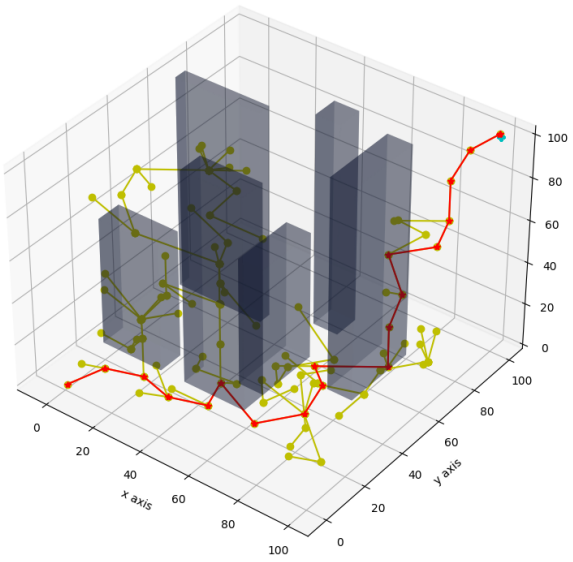


(a) An image of RRT in custom environment with bias

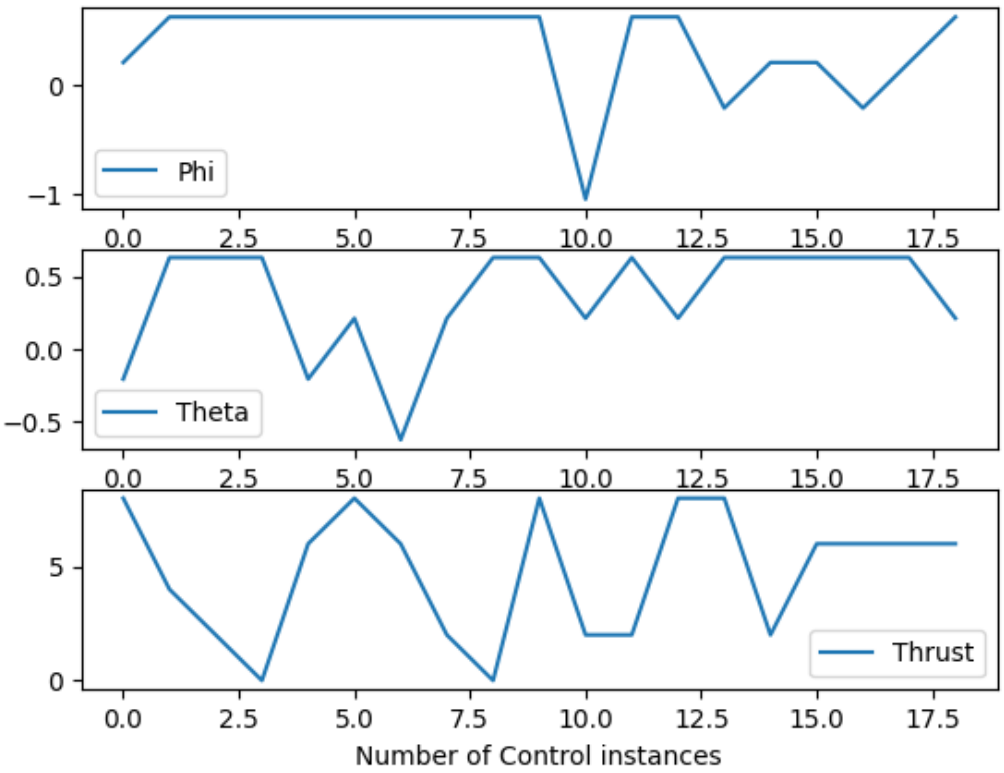


(b) An image of RRT in custom environment with bias

Figure 2: Custom Environment Results

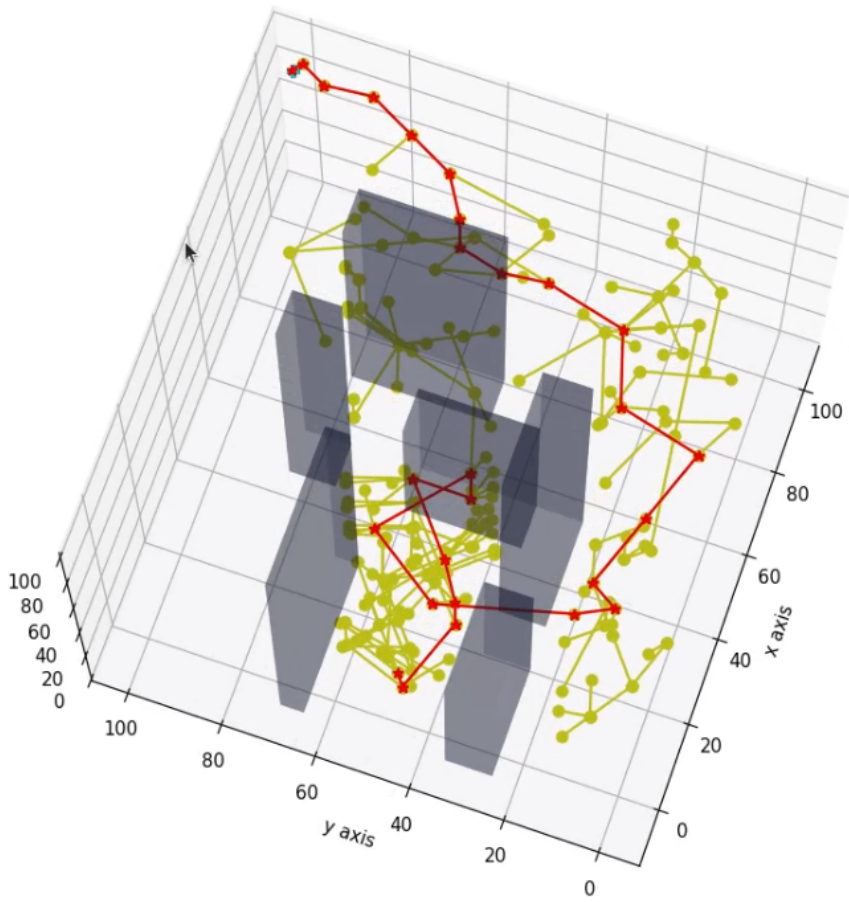


(a) Trajectory generation with start at (0 , 1 ,0) and goal at (100,100,100)

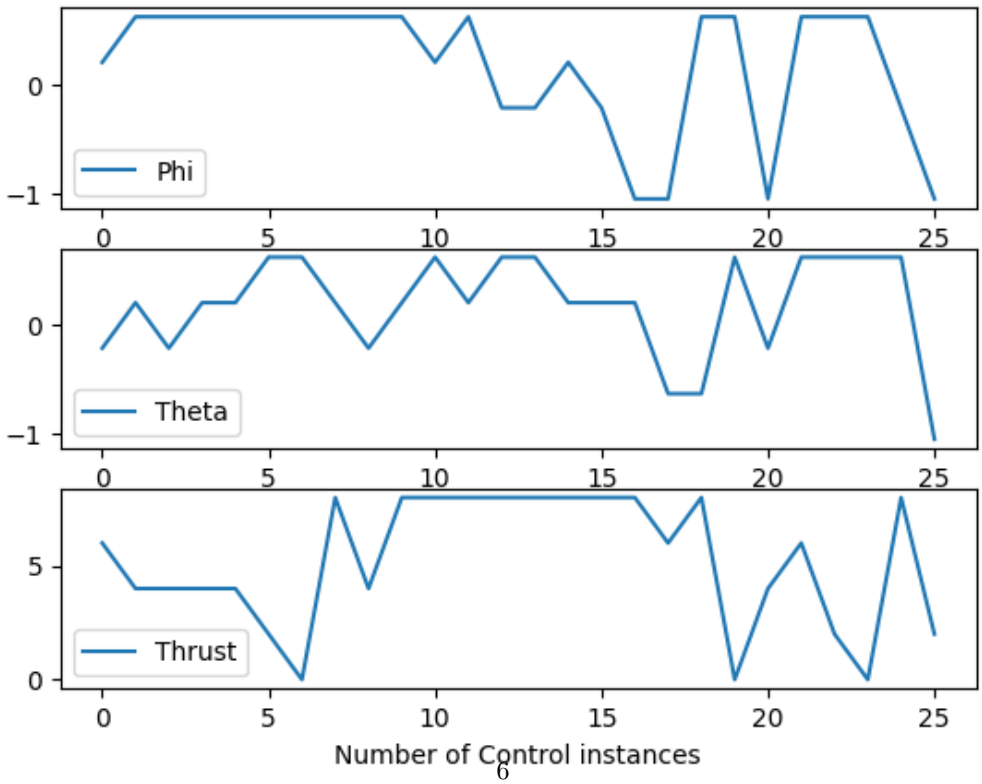


(b) ϕ, θ , T plot of control values

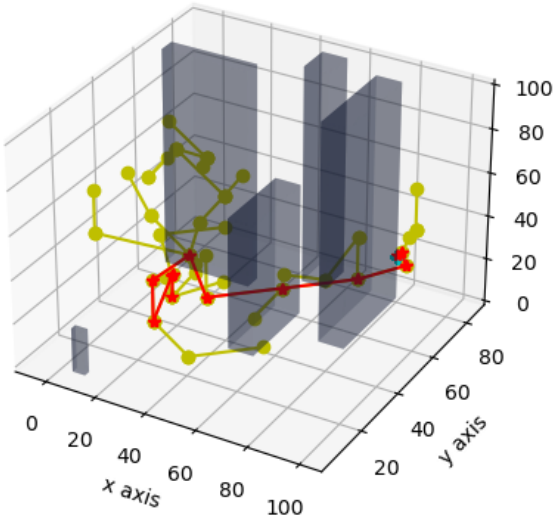
Figure 3: Original Environment Results Trail 1



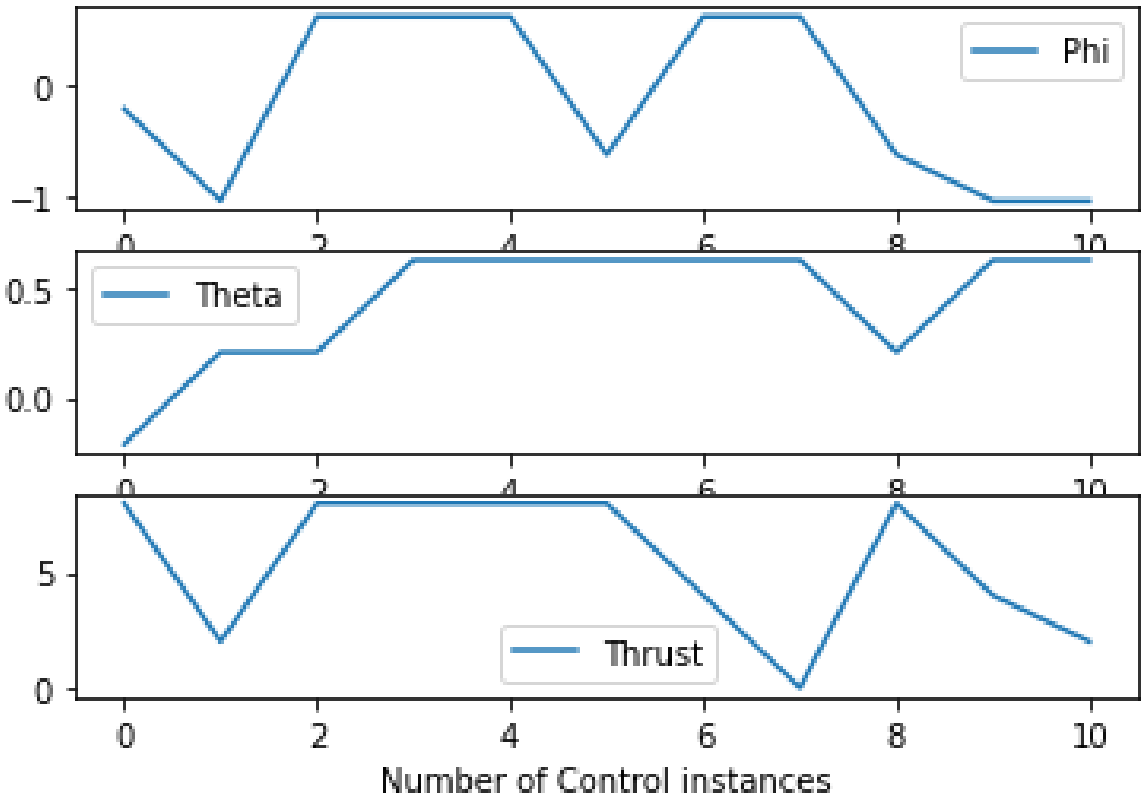
(a) Trajectory generation with start at (10 , 50 ,1) and goal at (100,100,100)



(b) ϕ, θ , T plot of control values



(a) Trajectory generation with start at (10 , 50 ,1) and goal at (100,50,50)



(b) ϕ, θ , T plot of control values

Figure 5: Custom Environment Results