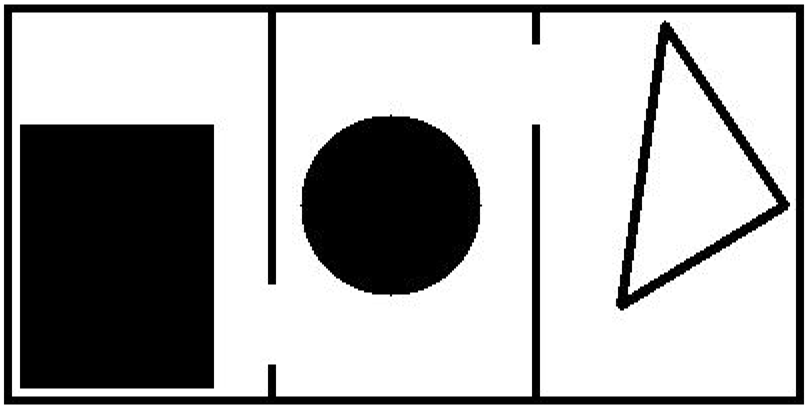
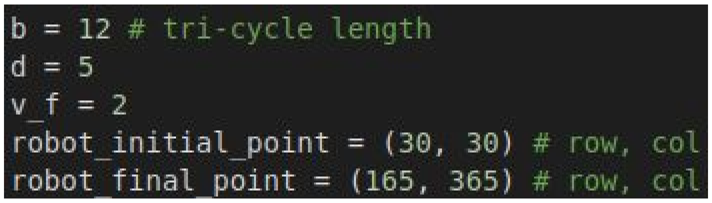
RRT Implementation (Holonomic and Non-holonomic)

Sachin Chandani (20161201, sachin.chandani@students.iiit.ac.in)
Pratikkumar Bulani (2019201074, pratikkumar.bulani@students.iiit.ac.in)

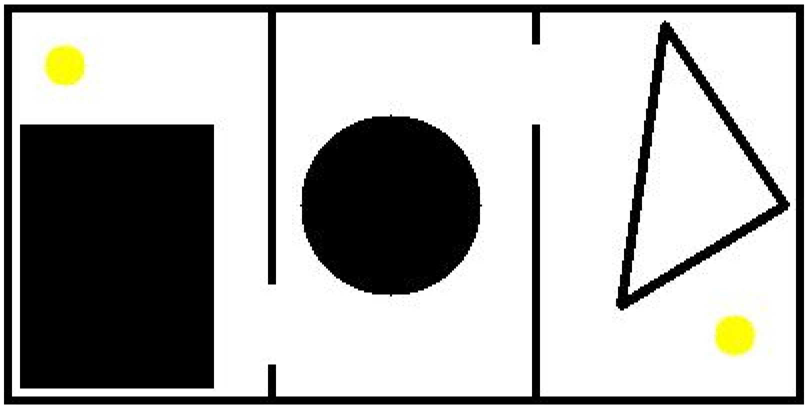
The **maze obstacle image** used for the algorithm implementation is:



The **robot's geometry** is initialized as (same for both, holonomic and non-holonomic):



```
b = 12 # tri-cycle length
d = 5
v_f = 2
robot_initial_point = (30, 30) # row, col
robot_final_point = (165, 365) # row, col
```



Initially, the RRT contains the robot's initial point as a node in the tree. The node structure of RRT for a holonomic robot is: (node's y location, node's x location, forward angle from the parent to reach this node, pointer to the parent node). The node structure of RRT for a non-holonomic robot is: (node's y location, node's x location, node's orientation theta, steer angle from the parent to reach this node, pointer to the parent node).

RRT Implementation for Holonomic robot:

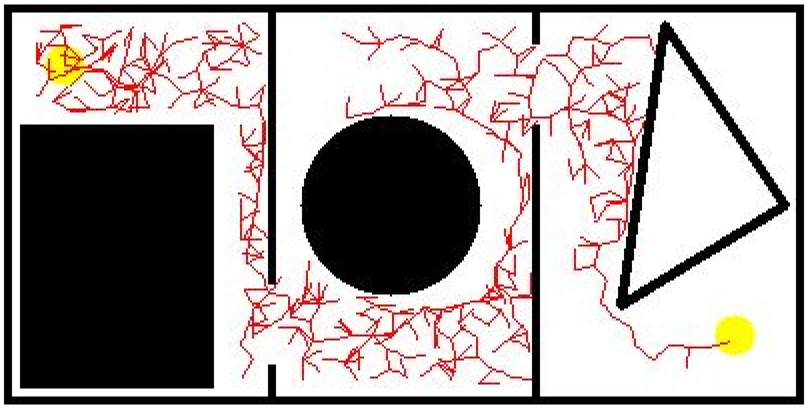
1. Start with the robot's initial point as the node present in the RRT.
2. Do till all the available free spaces are added in the RRT:
   a. Find a random location in the configuration space. Here the configuration space is the (x, y) point on the image. This is given by *np_q_rand* variable.
   b. Find the nearest node in the RRT based on Euclidean distance. This is given by *np_q_near* variable.
   c. Let's move along the direction given by (*np_q_rand* - *np_q_near*) by 10 steps starting from *np_q_near*. The function *compute_motion_primitive* does this for us. And this function returns the path moved by the mid-platform, left wheel and right wheel.
   d. Check the validity of this path i.e. check whether this path has collided or not.
   e. If a collision has occurred, go back to step 2. Else proceed to the next step.

f. Add the final location of this path to the RRT.
g. Show the frame where this path is added to the image.
h. If this added node is in the locality of the robot's final point, then break out of this loop.

Path Tracing: Path is traced from destination back to source using the parent pointer of the node.

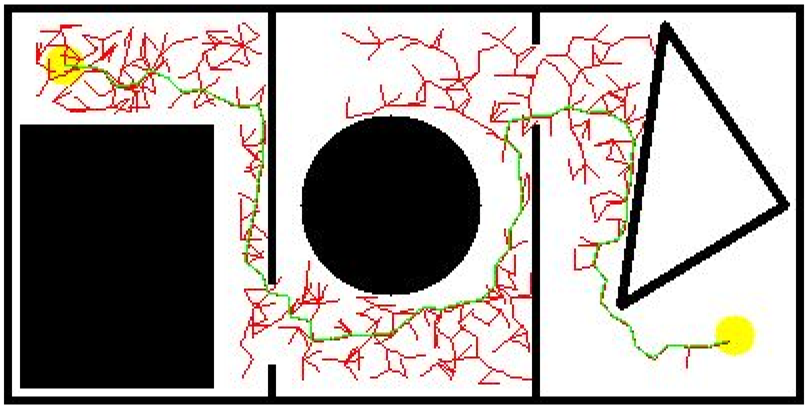
Results: All the videos are added inside the .zip file. And are also shared. But the shared videos might not open.

1. Video titled "Holonomic 1 - RRT.mp4" shows how the tree grows every iteration



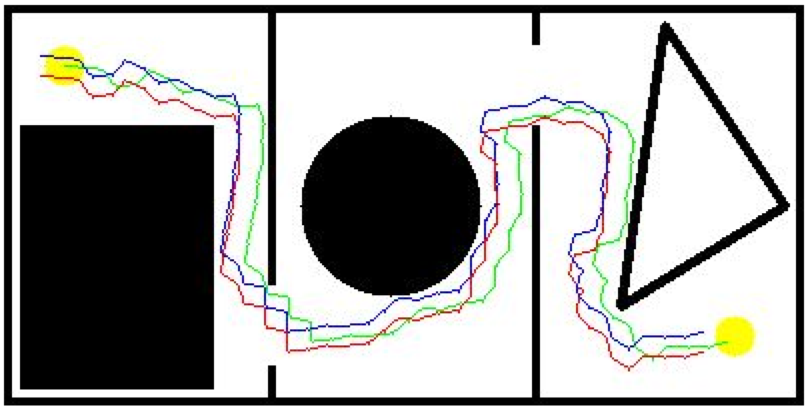
https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pratikkumar_bulani_students_iiit_ac_in/EeFPjiIs91IAjYY4PwT1qGEBkQ3MyxZb44HWDayCfsU-1Q?e=brckR1

2. Video titled "Holonomic 2 - Mid platform on RRT.mp4" shows the mid-platform path traced over the RRT



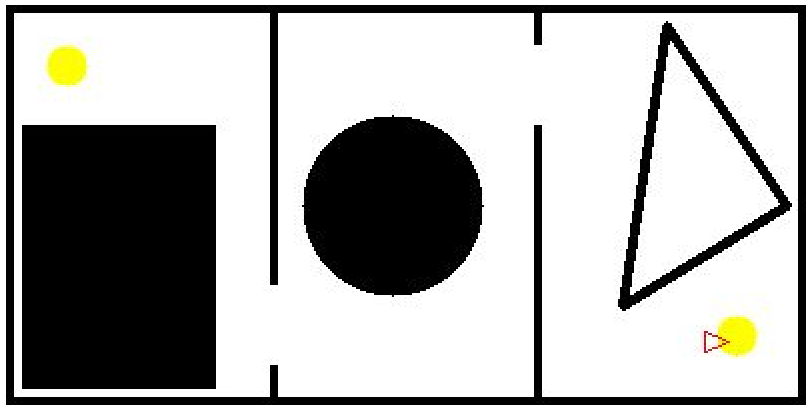
https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pratikkumar_bulani_students_iiit_ac_in/EaOoAN52g-RIowp-S3363A8BfzcnzmZBbIAsCDDjv7kcLA?e=2T1KJ1

3. Video titled "Holonomic 3 - Path traced all wheels.mp4" shows individual trajectories of all the wheels



https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pratikkumar_bulani_students_iiit_ac_in/EfV1pWZdCxpHv4ojI1ZhCMwBbTgv6Ae9jLHdXu0SzaZYeQ?e=WezOxd

4. Video titled "Holonomic 4 - Path traced with robot.mp4" shows the final robot movement in the maze

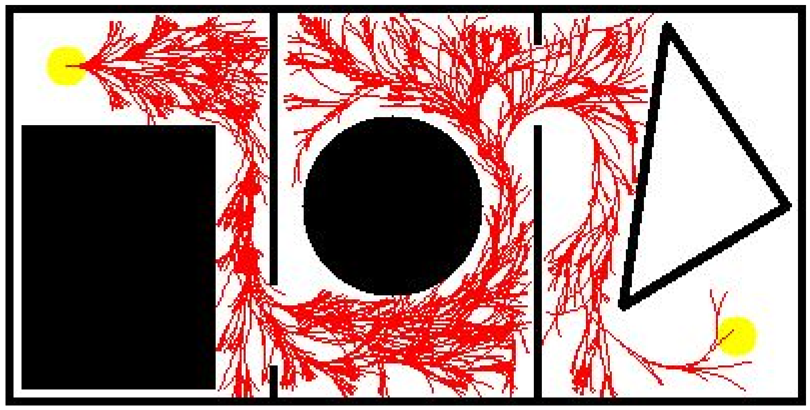


https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pratikkumar_bulani_students_iiit_ac_in/EYPIFqW0n6xBtrg22LzD8jsBVqwdTMaJpXJQ3cDDOdeg8A?e=KMZDoV

RRT Implementation for Non-Holonomic robot:
1. Start with the robot's initial point as the node present in the RRT.
2. Do till all the available free spaces are added in the RRT:
    a. Find a random location in the configuration space. Here the configuration space is the (x, y, theta) point on the image. This is given by *np_q_rand* variable.
    b. Find the nearest node in the RRT based on Euclidean distance and the achievable orientation (maximum steering angle is 30^o). This is given by *np_q_near* variable.
    c. Let's move along the direction given by (*np_q_rand* - *np_q_near*) by 10 steps starting from *np_q_near*. The function *compute_motion_primitive* does this for us. And this function returns the path moved by the mid-platform, left wheel and right wheel. While doing so, it ensures that the kinematic equations are followed.
    d. Check the validity of this path i.e. check whether this path has collided or not.
    e. If a collision has occurred, go back to step 2. Else proceed to the next step.
    f. Add the final location of this path to the RRT.
    g. Show the frame where this path is added to the image.
    h. If this added node is in the locality of the robot's final point, then break out of this loop.

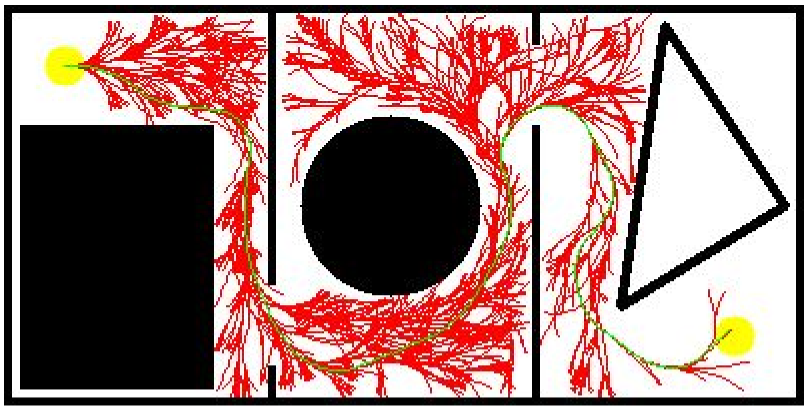
Path Tracing: Path is traced from destination back to source using the parent pointer of the node.

Results: All the videos are added inside the .zip file. And are also shared. But the shared videos might not open.
1. Video titled "NonHolonomic 1 - RRT.mp4" shows how the tree grows every iteration



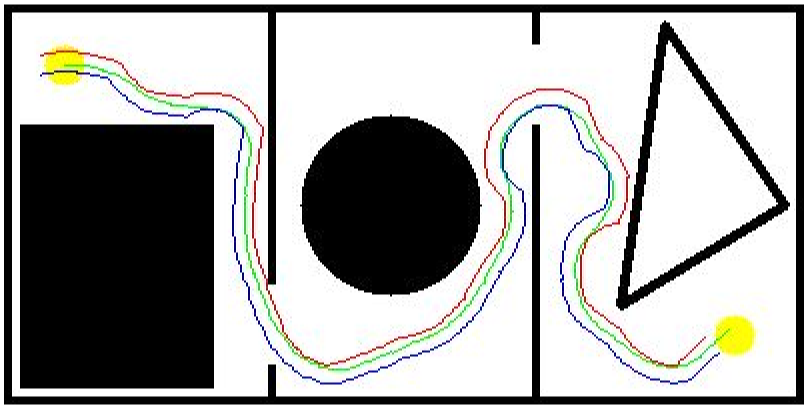
https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pratikkumar_bulani_students_iiit_ac_in/EV_xazfyP59Hr0mcJg5mMVgB5uTQ-PWXGU9cWFjM2zjESA?e=Lg1POu

2. Video titled "NonHolonomic 2 - Mid platform on RRT.mp4" shows the mid-platform path traced over the RRT



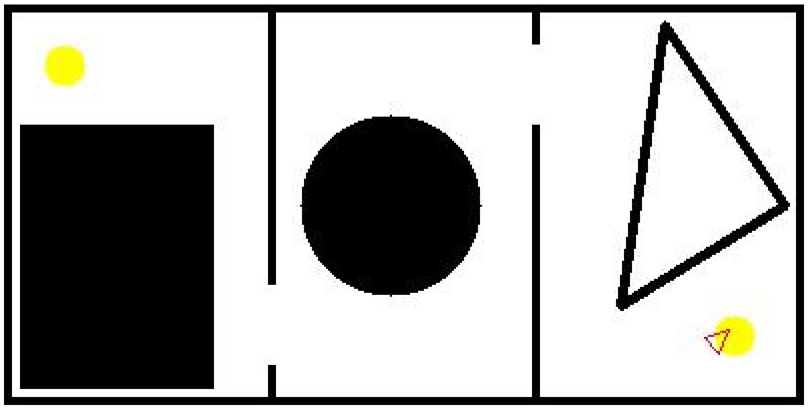
https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pratikkumar_bulani_students_iiit_ac_in/EZ7NYKrTjK9Ckpc_00h1W2kBpaKRGhb3cmBv5J4XH1JdJw?e=WfpUwl

3. Video titled “NonHolonomic 3 - Path traced all wheels.mp4” shows individual trajectories of all the wheels



https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pratikkumar_bulani_students_iiit_ac_in/EezQdKC82uNPo5Ad_9ax6dYBqTOUctn-yEQjxdJpXckkuw?e=to1YfT

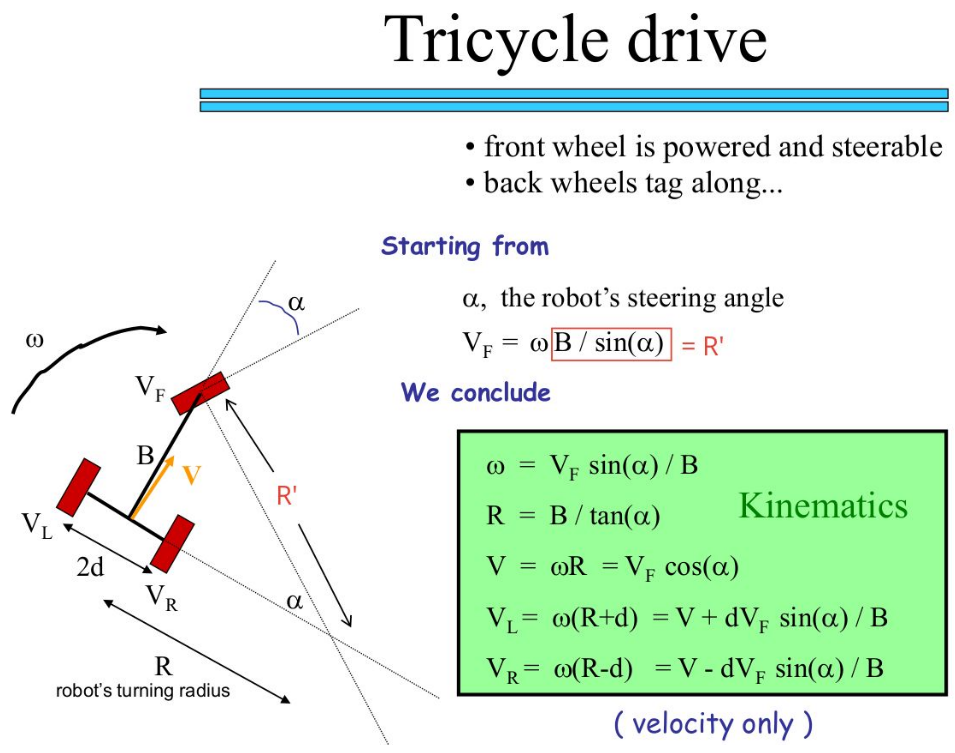
4. Video titled “NonHolonomic 4 - Path traced with robot.mp4” shows the final robot movement in the maze



https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pratikkumar_bulani_students_iiit_ac_in/EYxHq1DV6FBImgMYIJj_TYIBxog7TMuNQVUvlkeh_rQFTQ?e=tffRrK

Kinematics model used for holonomic: We have used a triangular robot where all the wheels steer along the same direction. So the orientation of the robot will not change. The robot is constantly looking forward.

Kinematics model used for non-holonomic: We have used the tricycle model constraints for the non-holonomic robot.



Tricycle drive

• front wheel is powered and steerable
• back wheels tag along...

Starting from

α, the robot's steering angle

$V_F = \omega \boxed{B / \sin(\alpha)} = R'$

We conclude

Kinematics

$$\omega = V_F \sin(\alpha) / B$$
$$R = B / \tan(\alpha)$$
$$V = \omega R = V_F \cos(\alpha)$$
$$V_L = \omega(R+d) = V + dV_F \sin(\alpha) / B$$
$$V_R = \omega(R-d) = V - dV_F \sin(\alpha) / B$$

(velocity only)

We can derive these equations if you want

Various experimentations done by us:

1. We tried using the cost grid map for the holonomic case to trace the shortest path between the source and the destination. The results were good but were not satisfactory. We shifted our approach to store the parent pointer for every next discovered node. We got this idea from the article: https://www.linkedin.com/pulse/rrt-implementation-non-holonomic-robot-part-2-car-like-ankit-saini/
2. We divided the path into multiple short segments as if we don't then the robot was able to enter the hollow obstacles. We have kept a hollow triangle obstacle on the right hand side.
3. We experimented for various values of forward velocity *v_f* and step size. Step size chosen for the holonomic robot is 5 and that for non-holonomic robot is 10. Each step = 1 time unit. We came to this set of values after extreme experimentation. We were just looking at the traced path for deciding these values. When *v_f* or time step is increased, then the path length each iteration is high.
4. For the non-holonomic robot, the maximum steering angle is kept to be 30°. The algorithm works for 20° as well but consumes a lot of RAM (> 4GB). We were trying to restrict the RAM requirement within 4GB. And 10° was way too less for our complex maze. Dubin's car uses 40°. So we thought that keeping maximum steering angle as 30° is a good option for us.
5. Earlier we thought to trace each wheel separately. In this case, all the wheels were moving parallel to each other. Because of this, the robot was extending and compressing. This was an erroneous code. Later we thought to find the relative position of the wheels wrt the midpoint. The later one was the correct one.
6. So as to add more nodes per iteration, we thought of adding all the possible locations that we found during the path tracing. This code is commented in our notebook but works well.

And many more such experimentations.

End of the report

Thanks to the TAs (YVS Harish and Abhinav Gupta) who helped us complete this assignment.