Assignment 3-Report

1. VO using sampling

Approach followed:

- We assume that the obstacle trajectories and velocities are known precisely at every instant.
- Loop over a max no. of iters and while the agent's current position is greater than a certain threshold distance from goal:
    - Compute desired velocity v_{des} in the direction of the vector joining the current position and the goal position.
    - Iterate over obstacles list and calculate Velocity Obstacle constraints as follows:
        - Find the vector joining the current position and the obstacle position r_{12} and the slope of this vector. Denote this slope by θ.
        - Find the tip of the VO cone at $r_1 + v_2$ and angle θ_c of the VO cone at the tip. This gives us the angles made by the tangents of the VO cone:

          $\theta+\theta_c$ and $\theta-\theta_c$
    - Sample random velocity vectors with angle ranging from $(0,2\pi)$ at intervals of 0.1 and magnitude ranging from 0.02 to v_{des}.
    - Find out if the tip of these randomly selected vectors lie within the VO cone. To do so, check if the angle made by the randomly chosen vector is greater than $\theta-\theta_c$ and less than $\theta+\theta_c$
    - Append sampled velocities that lie outside the VO cone into a $v_{suitable}$ list. From this list choose the velocity with the minimum norm to the previous velocity.

2. VO using fmincon

Approach followed:

- We assume that the obstacle trajectories and velocities are known precisely at every instant.
- While the agent's current position is greater than a certain threshold distance from the goal:
    - **Check for obstacles within sensor range:**
      Iterate through the obstacles list and check which of the obstacles lies within the sensor range of the agent. This is defined in the code `inSensorRange.m`. Current sensor range is defined as 5 units.

    - **Compute VO constraints for these obstacles:**
      As defined in `getConstraints.m` for each agent within the sensor range, we check if the agent is indeed in collision with the obstacles by checking if d^2 - R^2 < 0. We calculate the Velocity Obstacle collision constraint as follows:

$$d^2 = |\vec{r}|^2 - \frac{\vec{r}.\vec{V}_{1/2}}{|\vec{V}_{1/2}|^2} \geq R^2$$

      Where r denotes the relative position vector between the agent and the obstacle.
      I.e . $r = r_1 - r_2$ where
      r_1 = position vector of agent
      r_2 = position vector of obstacle
      V_1 = velocity of agent
      V_2 = velocity of obstacle
      $V_{1/2}$ = relative velocity vector between agent and obstacle such that,

$V_{1/2} = V_1 - V_2$
R = (radius of agent + radius of obstacle)

This constraint is convex wrt v_x and v_y for a holonomic robot so we can directly use it as a convex constraint during our minimization process.

- **Define cost function**

  We define the following goal reaching cost function:

  $$J = (x_n - x_g)^2 + (y_n - y_g)^2$$

  $$x = x_0 + v_x \Delta t$$

  $$y = y_0 + v_y \Delta t$$

  where (x_n , y_n) denote the current position of the agent and (x_g , y_g) denote the goal position.

  This cost is defined in `cost_distance.m`.

- **Minimise cost function subject to these constraints**

  We use MATLAB's inbuilt optimizer `fmincon` to minimize the cost function defined above subject to the VO constraint to find the set of controls v_x and v_y that the robot should use to avoid the constraint.

```
controls = fmincon(cost,init,A,b,Aeq,beq,lb,ub,constraints);
```

  Here we use max and min velocity limits as lb and ub. So we have,

  $V_{min} = 0$

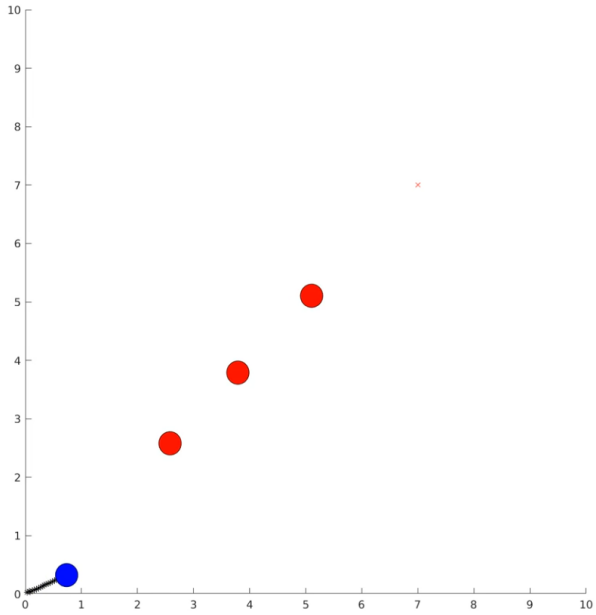
$V_{max} = 0.4$

- **Move agent with controls found**

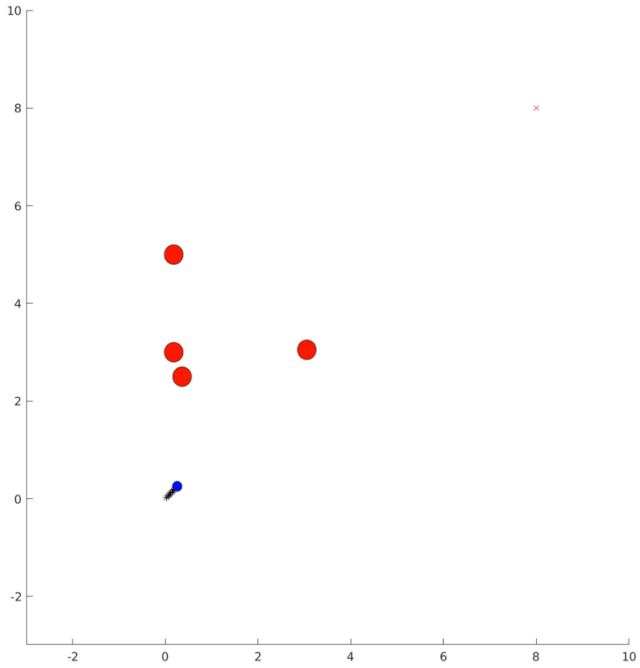
We use holonomic kinematics to make the agent move using the controls found.

$$x = x_0 + v_x \Delta t \qquad y = y_0 + v_y \Delta t$$

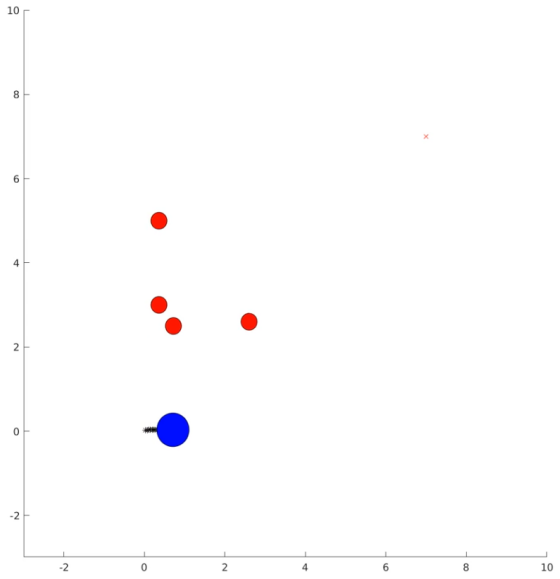
Tests performed:



- **Agent radius = obstacle radius:**
  - -0.2 m/s along x=y towards origin
  - -0.2 m/s along x=y towards origin
  - 0.05 m/s along x=y towards goal



- **Agent radius < obstacle radius:**
  - -0.2 m/s towards origin
  - 0.1 m/s along y = 2.5
  - 0.2 m/s along y = 2.2
  - 0.1 m/s along y = 5



- **Agent radius > obstacle radius:**

- -0.2 m/s towards origin
- 0.1 m/s along y = 2.5
- 0.2 m/s along y = 2.2
- 0.1 m/s along y = 5