RPN - Assignment #3

Team #3 - Makkhi-2

Tanmay Pathak - 2018102023
Amogh Tiwari - 2018111003

Dynamic Velocity Obstacle

Introduction and Theory

A Velocity obstacle (VO) refers to the set of all velocities of a robot which will result in collision with another robot moving in the same space. This formulation is used in obstacle avoidance such that we make sure that the robot(s) choose(s) a velocity which does not lie in VO thereby, avoiding collision.
There are two methods to solve this problem -
(a) By framing it as a sampling problem,
(b) By framing it as an optimization problem.

At each time step, we look at all the possible velocities which the robot may take, subject to the constraint that the chosen velocity lies between a user-specified range of minimum and maximum velocities (v_min and v_max respectively). Then, from among these velocities we look at all the velocities which do not lead to a collision. And then from among the set of velocities which lie between v_min and v_max and also don't lead to a collision, we choose the velocity which minimizes the distance between robots present position and its goal.

Note: For the purpose of this assignment, we have framed the problem as a **sampling** problem, and have solved it for the case of a **holonomic** robot.

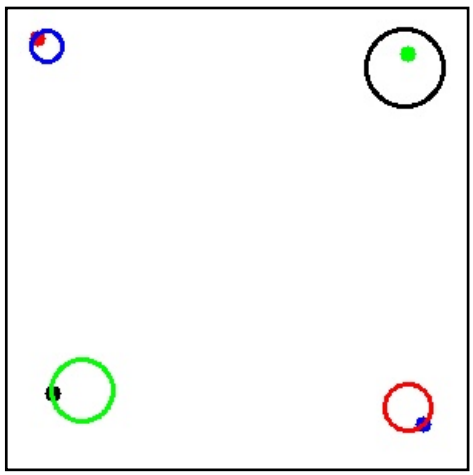
Code Outline

1. Create an empty image. This empty image represents our world.
2. Create a set of robots. (**Note:** In order to ease the coding process, the dynamic obstacles too have been modelled as robots and our "main" robot tries to avoid collision with these moving robots)
3. Define the parameters to those robots. This is done by calling the **addRobot()** function
4. Once the robot parameters have been added, they are added to our world by calling the **createWorld()** function

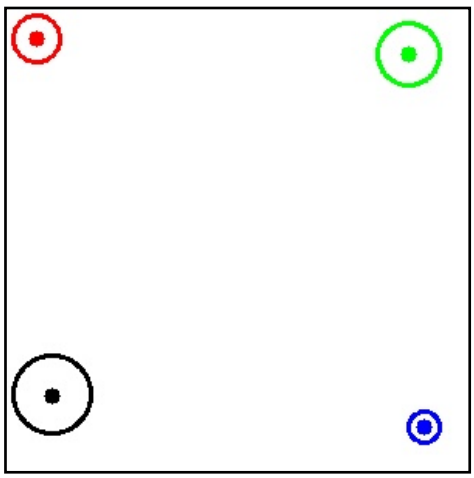
5. Once all the parameters have been created, we go to the **solver()** function. The **solver()** function is the main function which solves the problem
6. Inside the solver function, we do the following:
   - For all robots
     - Get a set of velocities which lie in range v_min and v_max
     - Now, from among these set of velocities, further filter to get the set of velocities which do **not** result in a collision
     - Once the set of velocities which do not result in a collision have been identified, from these velocities, choose the velocity which minimizes the distance between robots present location and its goal
   - Keep doing the above process for each robot, till the robot is within a threshold distance of its goal
   - In case the number of iterations go beyond a threshold maximum number of iterations, then exit even if the goal is not reached. This is done to prevent the program from going into an infinite loop for the cases where a solution is not possible.

Results

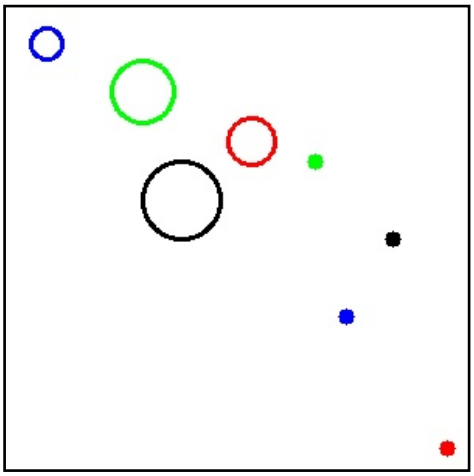
For 4 agents (1 Main agent + 3 obstacles) - LINK TO VIDEO (DIAGONAL) and (RANDOM)



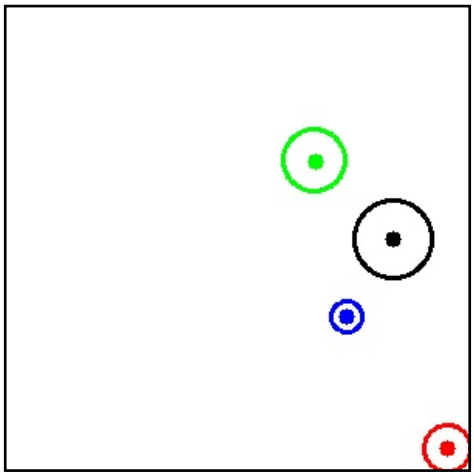
Initial Positions



Final Positions - 27 iterations

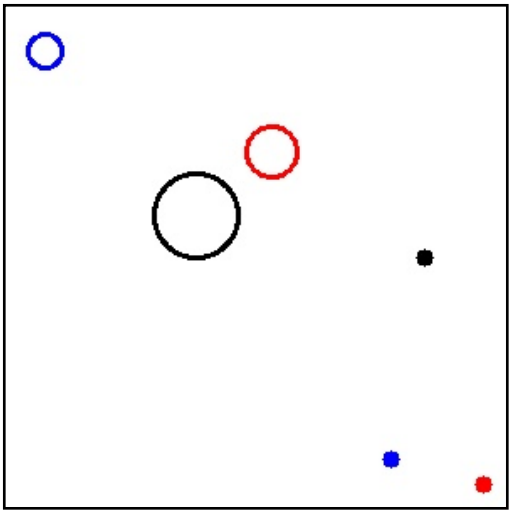


Initial Positions

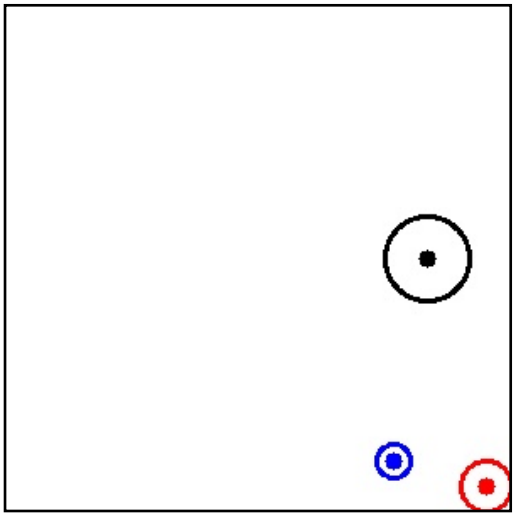


Final Positions - 19 iterations

For 3 agents (1 Main agent + 2 obstacles) - LINK TO VIDEO (RANDOM)

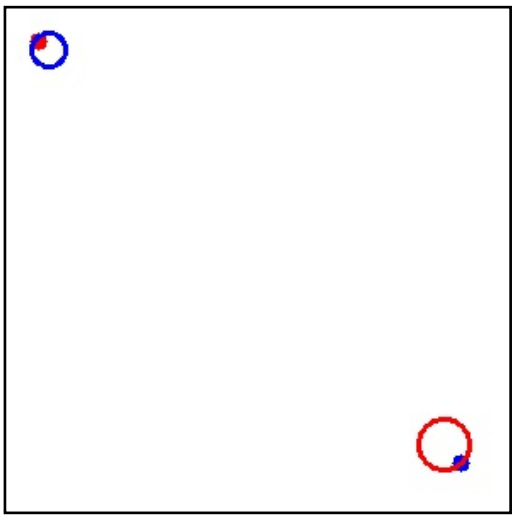


Initial Positions

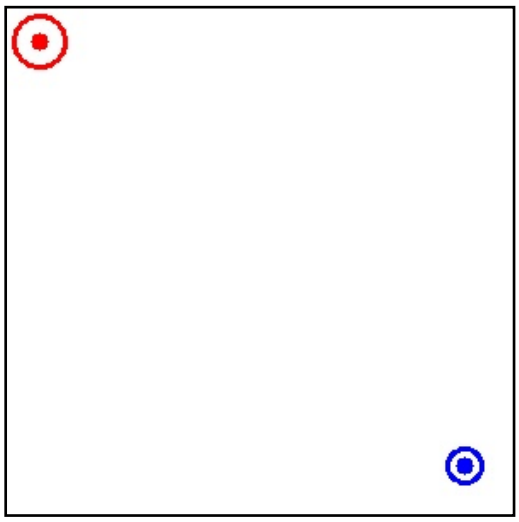


Final Positions - 23 iterations

For 2 agents (1 Main agent + 1 obstacle) - LINK TO VIDEO (DIAGONAL)



Initial Positions



Final Positions - 25 iterations