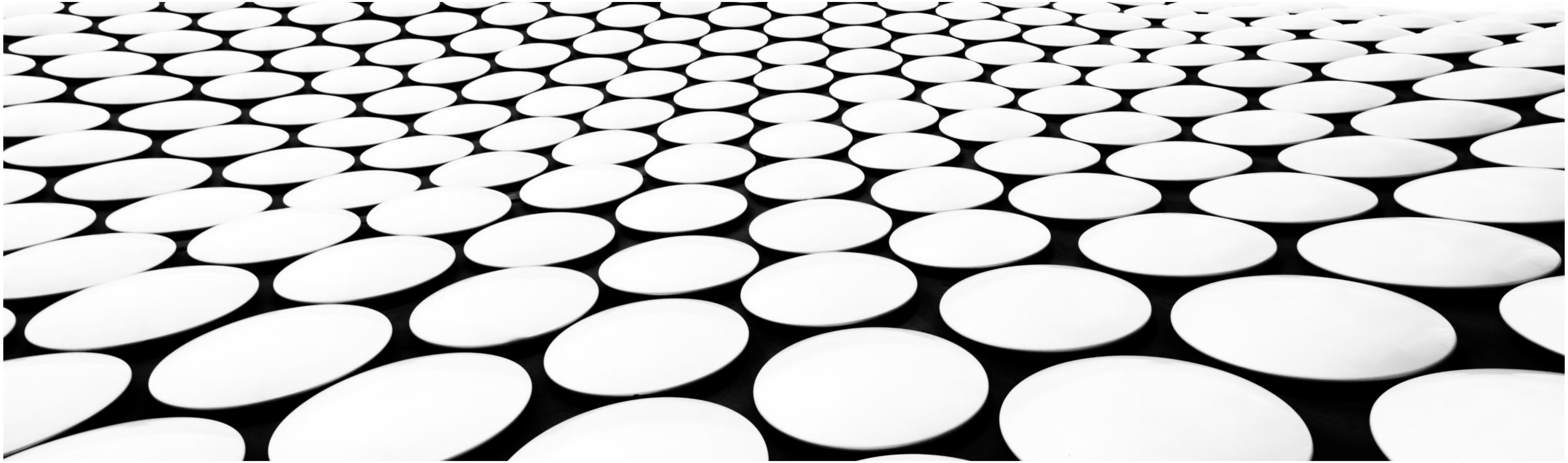
ROBOT OBSTACLE AVOIDANCE WITH REINFORCEMENT LEARNING

INTELLIGENT SYSTEMS AND REINFORCEMENT LEARNING

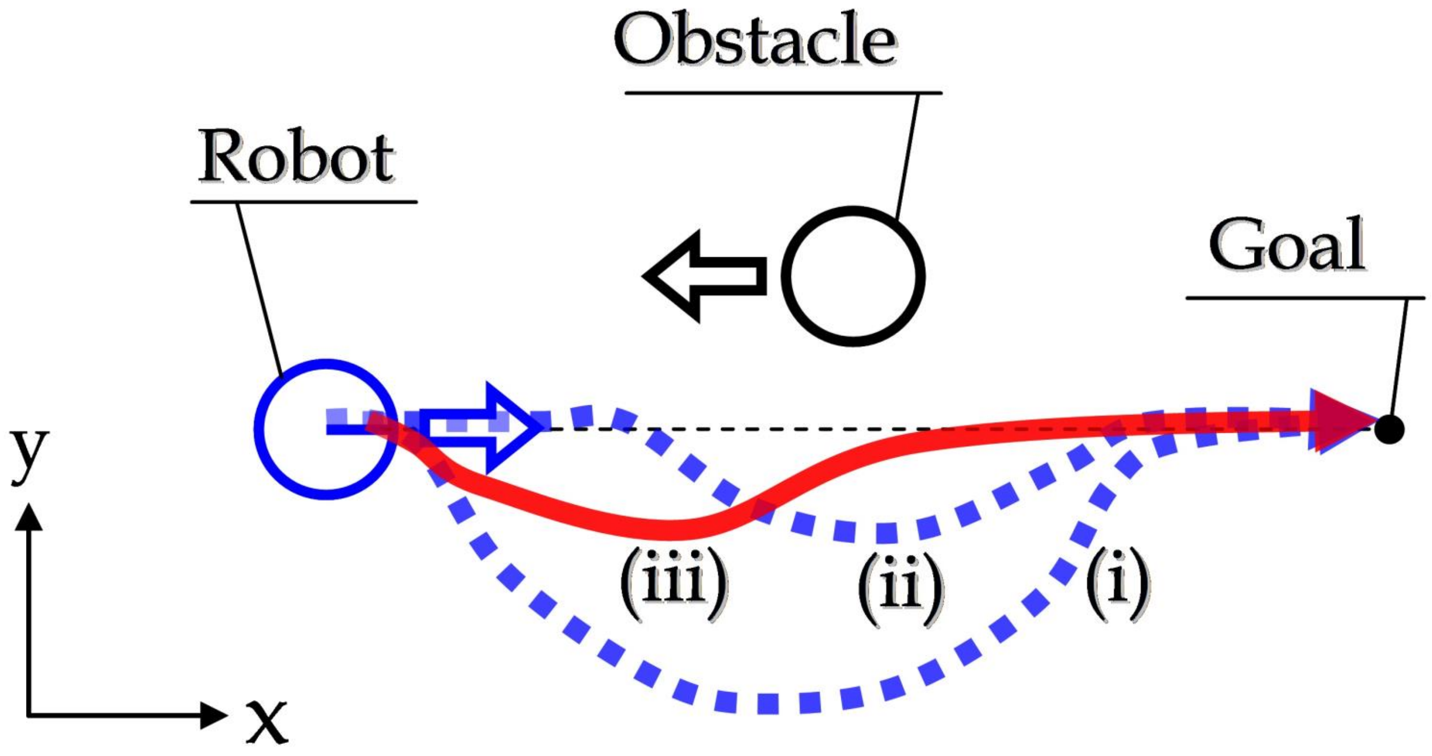
Group Members:

1. Alexandre Dietrich
2. Ankur Tyagi
3. Haitham Alamri
4. Rodolfo Vasconcelos



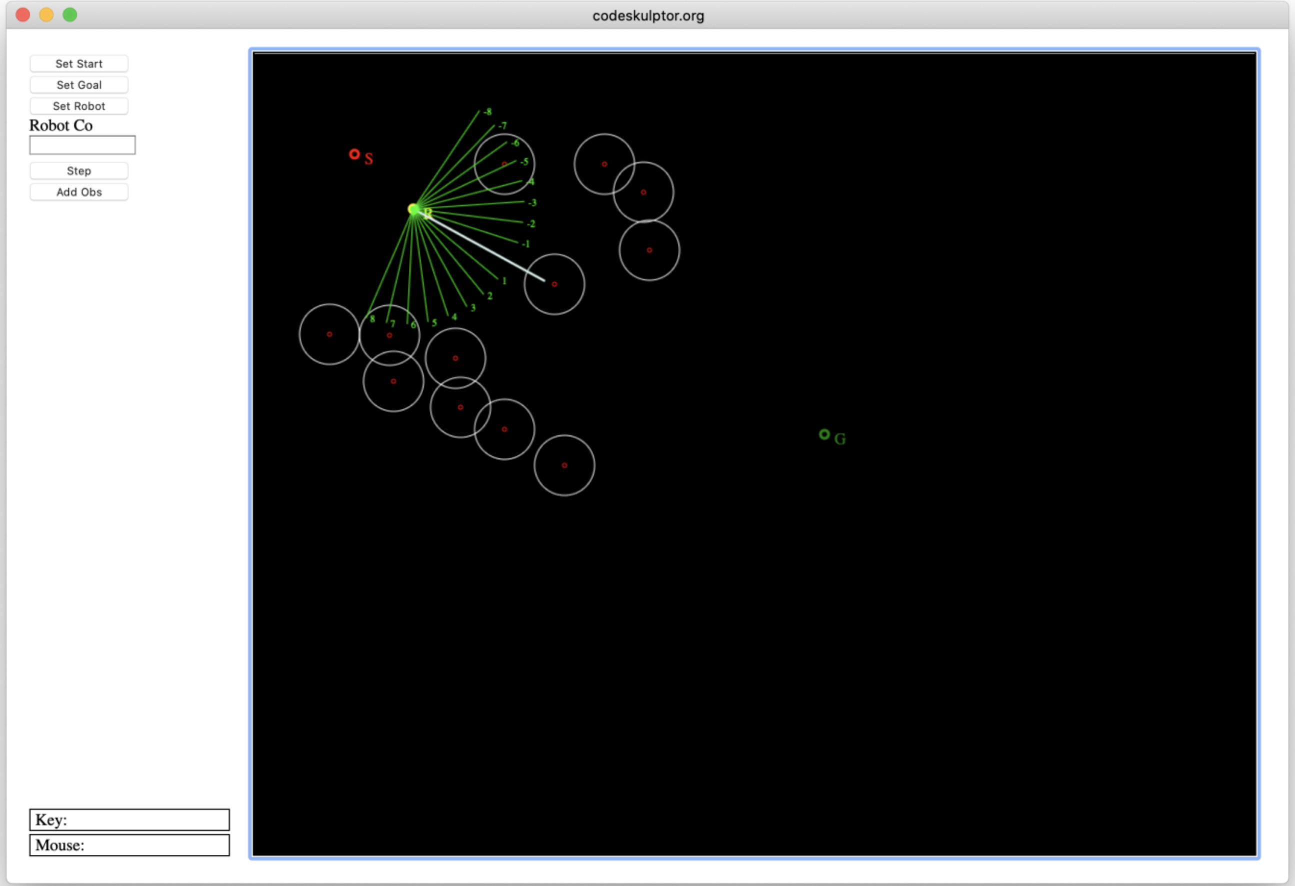
PROBLEM STATEMENT

Find the path for robot to reach an end point (goal state) while avoiding randomly generated obstacles in a 2D space



ENVIRONMENT

Environment simulation using python 3 for the project



*sample screen shot

Simulation Steps:

1. Choose the starting coordinates of the robot
2. Choose the target coordinates
3. Define static obstacles – generated randomly for each episode
4. Define number of sensors on the robot
5. Navigate the path using the step and avoid any incoming obstacles while navigating towards the target state.

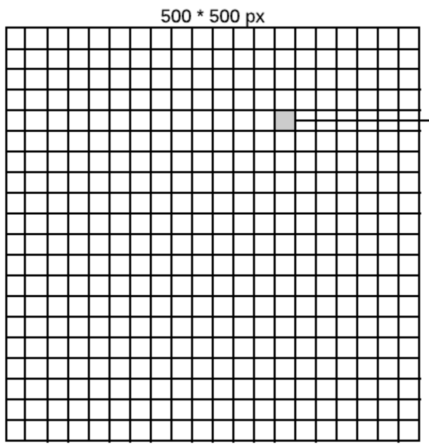
Objectives:

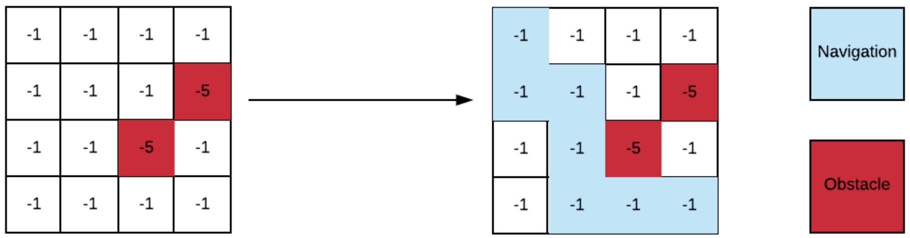
1. Identify and optimize the algorithm to avoid the obstacle and reach the destination
2. Compare the efficiency of algorithms against each other.

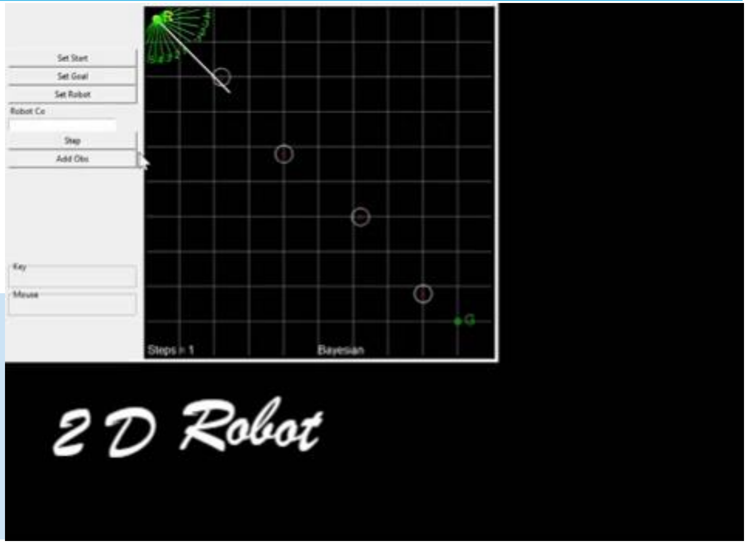
SOLUTION APPROACH

- Set the Canvas size to be 500*500 pixels
- Grid of 50*50px with each grid of 12.5 pixels
- Actions(s): Up, Down, Right and left
- Dynamically generated obstacles

- States: all possible states within a 4 x 4 grid with 0 or more obstacles
- Reward function:
  - -1 cost of step
  - -5 cost of obstacle
  - 0 goal end state







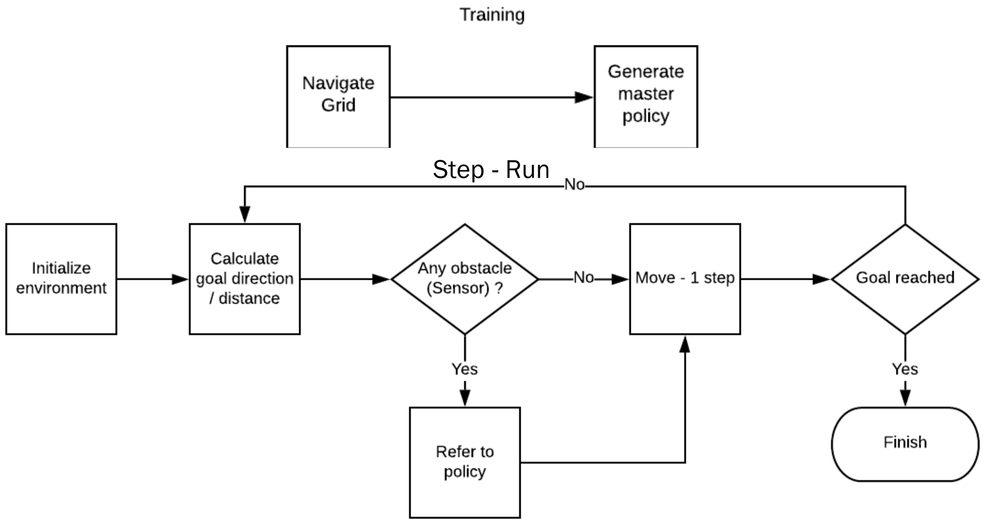
One grid - matrix with 2 possible obstacle

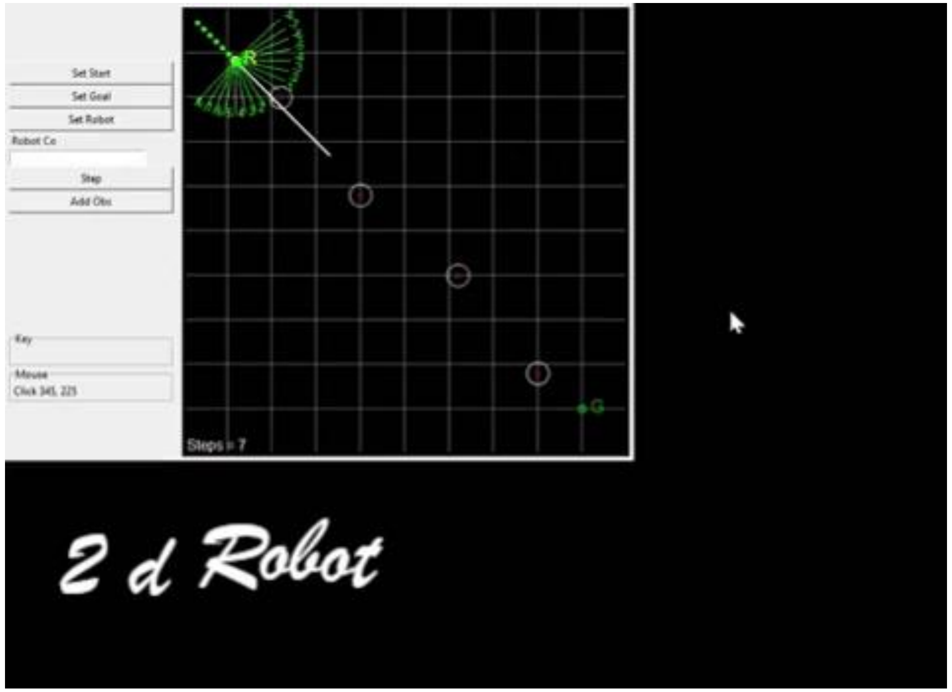
ALGORITHM – 1 (BAYESIAN + STATIC)

Execution Steps:

- Environment:
  - Starting point : top corner(0,0)
  - End point - bottom right corner (500, 500)

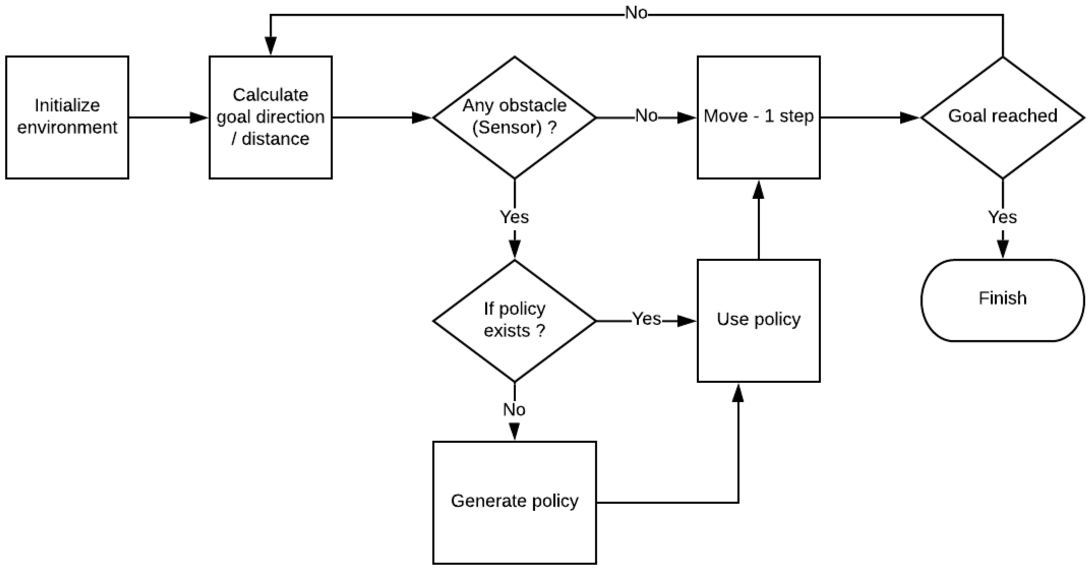
- Monte-carlo method to simulate 1000's of episodes and value iteration
- Find optimal q-value for every state in 4*4 grid and determine action based on epsilon soft policy algorithm
- The generated master policy is a dictionary of states and actions for all 50*50 grids
- Use the optimal policy to playing stage and then run through the agent from the starting point to the goal
- T(s'|s; a): probability of reaching s' if action a is taken in state s = 1 (No uncertainty)

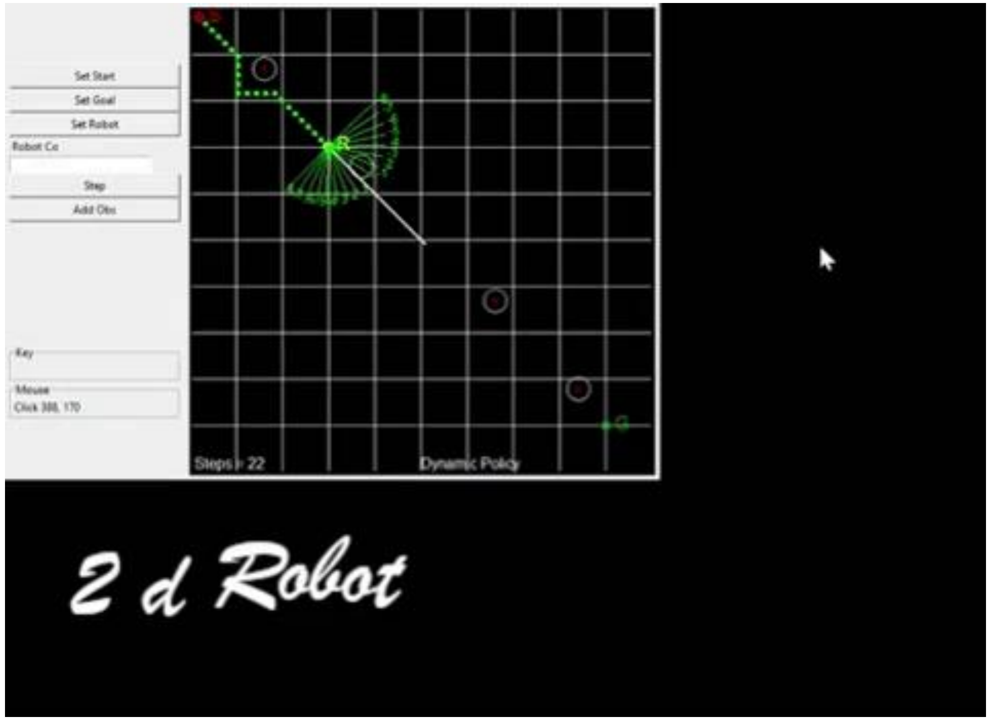




ALGORITHM – 2 (DYNAMIC BAYESIAN POLICY)

- Environment
  - Dynamic start point for the robot
  - Dynamic goal (end-point) for the robot
- In start of every episode robot begins without a policy
- If a policy is not available for that state, a new policy is created using the same Monte Carlo method
- T(s'|s; a): probability of reaching s' if action a is taken in state s < 1 (some uncertainty)





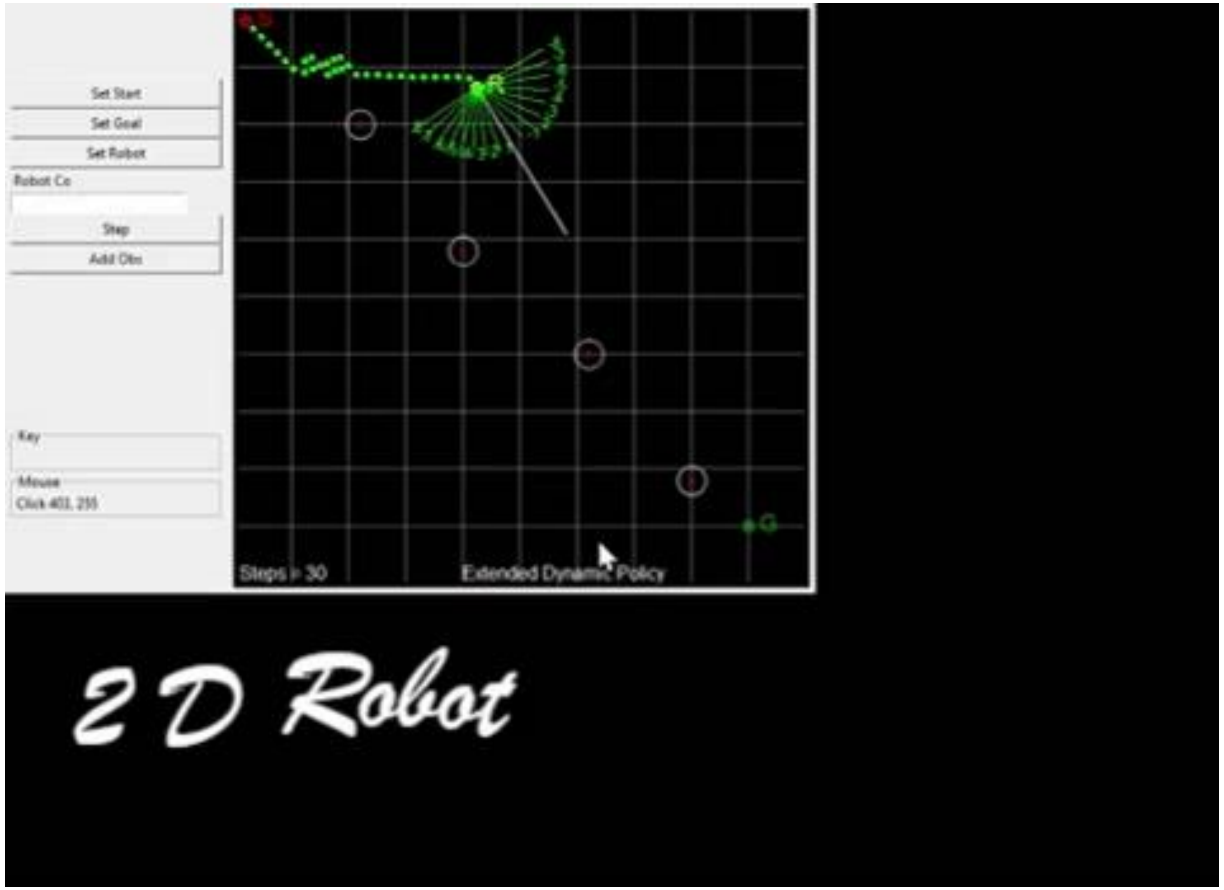
ALGORITHM – 3 (EXTENDED DYNAMIC POLICY)

- Environment
  - Dynamic start point for the robot
  - Dynamic goal (end-point) for the robot

- Algorithm extension :
  - In this algorithm, the policy matrix calculation is changed where the policy looks into nearby square matrix as well to define the path of the robot

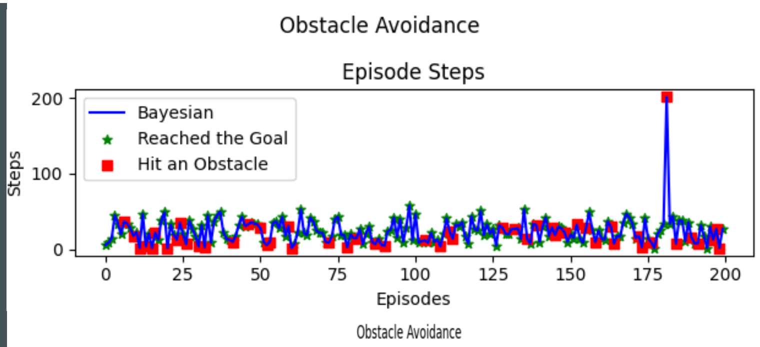
- Grid calculation changed from 50*50 to 250 * 250px

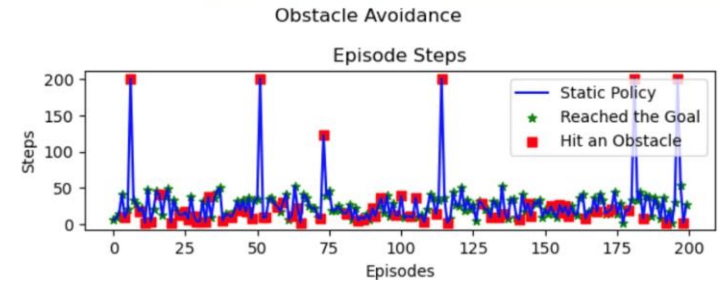
- Instead of Monte-carlo in a grid of 4*4 , we are calculating a grid of 5*5.

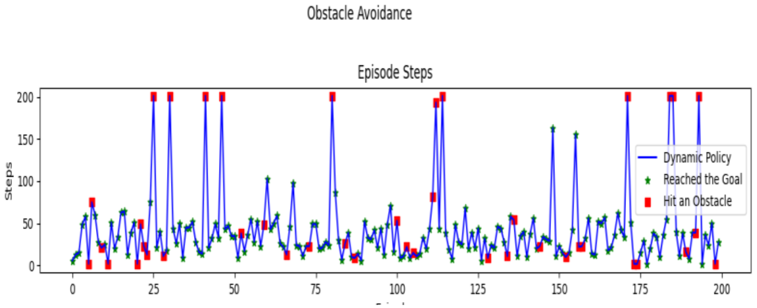


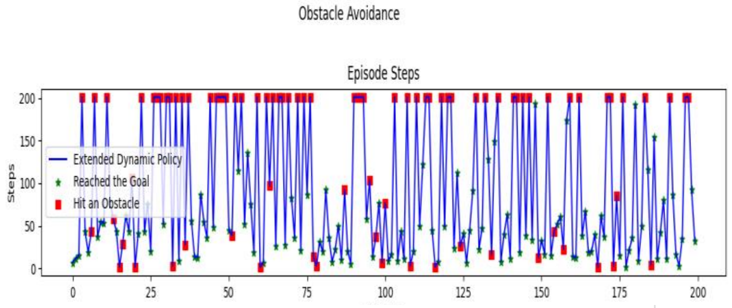
SUMMARY - ALGORITHM COMPARISONS

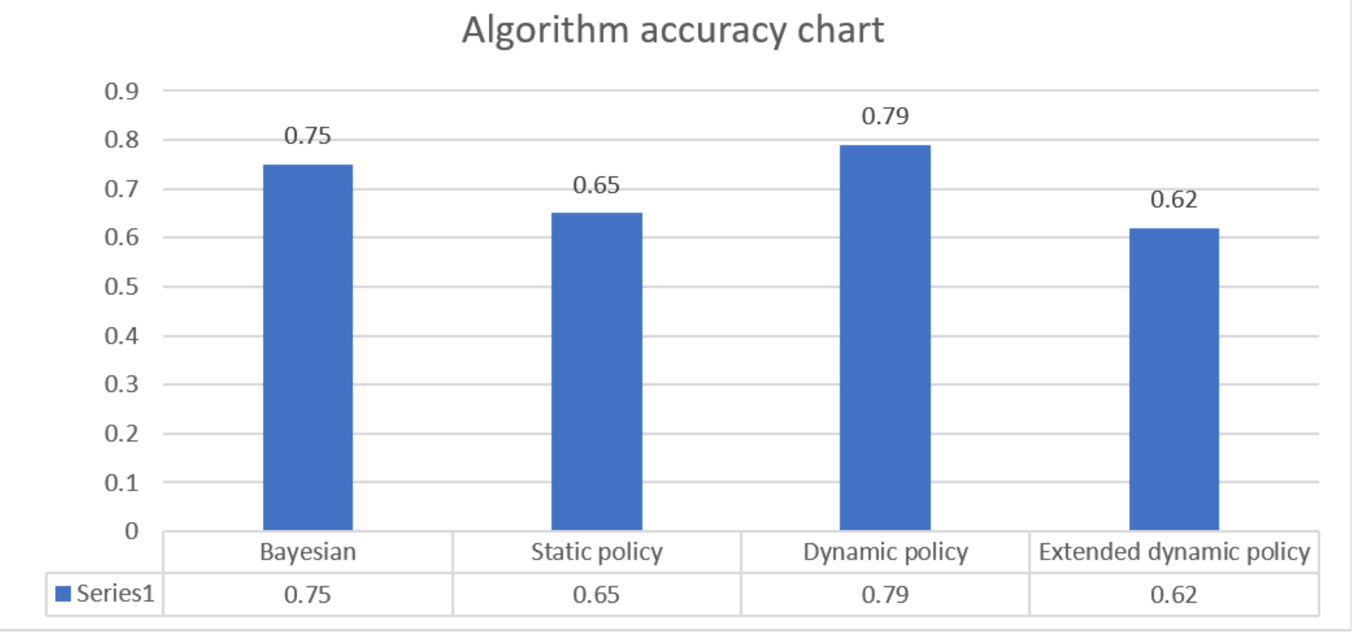
- ***Dynamic policy algorithm*** has the highest consistency among all the algorithm to navigate through obstacles while ***extended dynamic policy*** couldn't scale up as expected.
- Github: https://github.com/ravasconcelos/rl_obstacle_avoidance/
- Report: https://github.com/ravasconcelos/rl_obstacle_avoidance/blob/master/notebook/Term%20Project%20V2.ipynb











CONCLUSION – LESSON LEARNT AND FUTURE WORK

Lesson Learnt

1. It's difficult to solve continuous problems using discrete solutions
2. Enhancing object detection function has improved object avoidance algorithm

Possible extensions of the work include :

1. Use deep learning – q network to further increases the accuracy for the path navigation as done in the snake game (reference : https://github.com/maurock/snake-ga)
2. Use deep deterministic policy gradient for solving the problem in continuous domain (reference : https://www.youtube.com/watch?v=PngA5YLFuvU&t=187s)