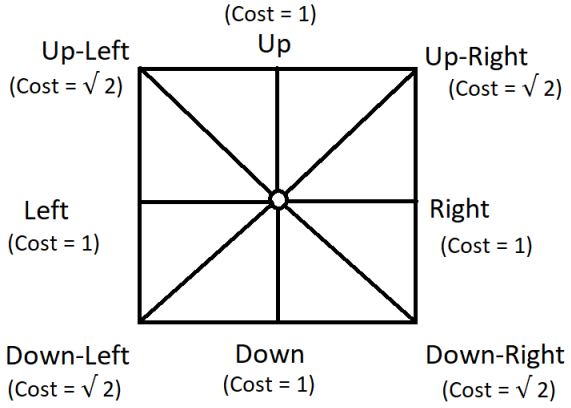
Project 2: Implementation of Dijkstra and A* algorithm on point and rigid robots

Due Date – 03/31/2019

Dijkstra and A* on a point robot

- Implement Dijkstra and A* algorithm to find a path between start and end point on a given map for a point robot (radius = 0; clearance = 0).
- Consider workspace as a 8 connected space, that means now you can move the robot in up, down, left, right & diagonally between up-left, up-right, down-left and down-right directions.



Dijkstra and A* on a point robot (Continued..)

- Use Half planes and semi-algebraic models to represent the obstacle space. (Read Chapter 3: Geometric Representations and Transformations from Planning Algorithms by Steven M. LaValle)

- Show optimal path generation animation between start and goal point using a simple graphical interface. You need to show both the node exploration as well as the optimal path generated.

Dijkstra and A* on a point robot (Continued..)

- For this part of the project your code must take following inputs from the user

1. Start point
2. Resolution / Grid Size for the map
3. Goal point

Remember to check the feasibility of all inputs

- Your code must output an animation of optimal path generation between start and goal point on the map. You need to show both the node exploration as well as the optimal path generated. (For Python and C++, students can use openCV to make the GUI)

Dijkstra and A* on a rigid robot

- Implement Dijkstra and A* algorithm to find a path between start and end point on a given map for a rigid robot (radius ≠ 0, clearance ≠ 0).
- Consider workspace as a 8 connected space.
- Use Half planes and semi-algebraic models to represent the obstacle space.
- Use Minkowski addition to enlarge the obstacles.
- Show optimal path generation animation between start and goal point using a simple graphical interface.

Dijkstra and A* on a rigid robot (Continued..)

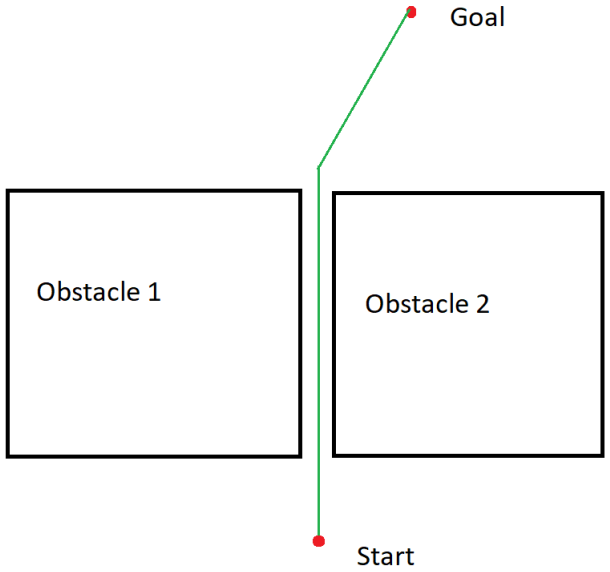
- For this part of the project your code must take following inputs from the user

1. Start point
2. Robot radius
3. Clearance
4. Resolution / Grid Size for the map
5. Goal point

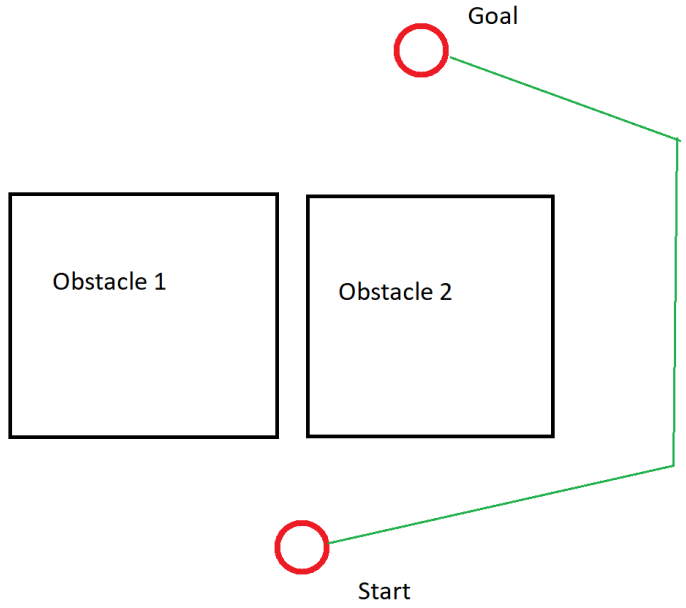
Remember to check the feasibility of all inputs

- Your code must output an animation of optimal path generation between start and goal point on the map. You need to show both the node exploration as well as the optimal path generated. (For Python and C++, students can use openCV to make the GUI)

Difference between point and rigid robot



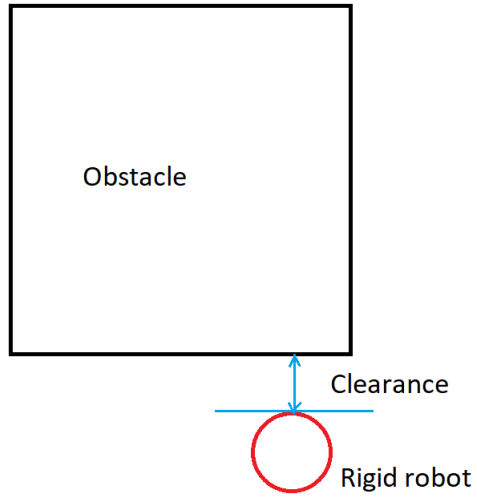
Navigation scenario for point robot



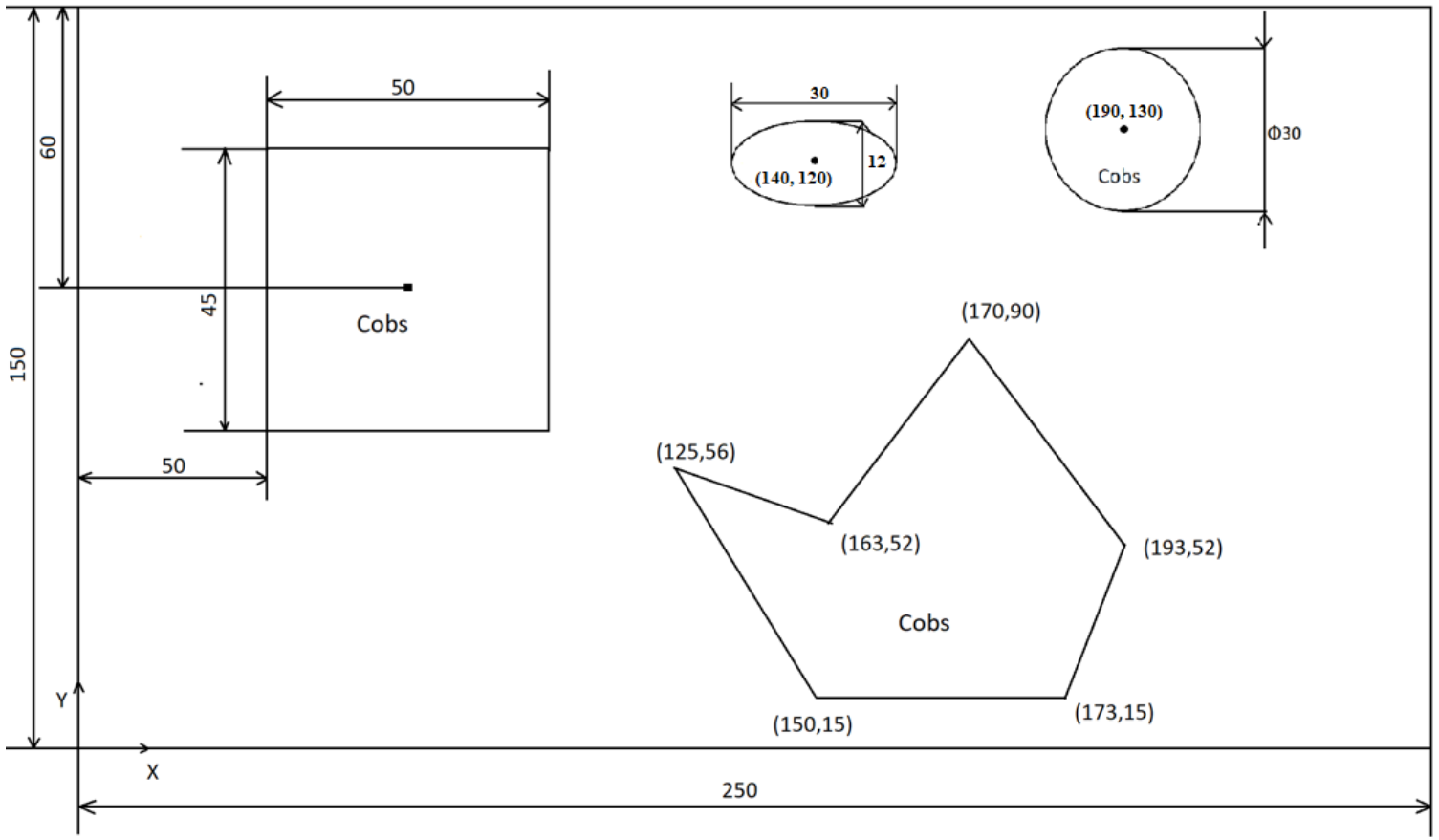
Navigation scenario for rigid robot

Clearance

- Clearance is a maximum distance between the obstacle and the extreme point of the rigid robot.



Map to be Used



Note –

1) The map is not scaled to the true dimensions

Due Date and Deliverables

- Due date: 03/31/2019
- Deliverables:

1. ReadMe file (Describing how to run the code)
2. Source files for
    I. Dijkstra_point
    II. Dijkstra_rigid
    III. A*_point
    IV. A*_rigid

Submission Details

- You are required to submit a zip file with the file structure as shown

proj2_firstname_lastname_codingLanguage
— codes
— readme.txt