CSE 4360 / 5364 - *Autonomous Robots*

Project 1- Fall 2024

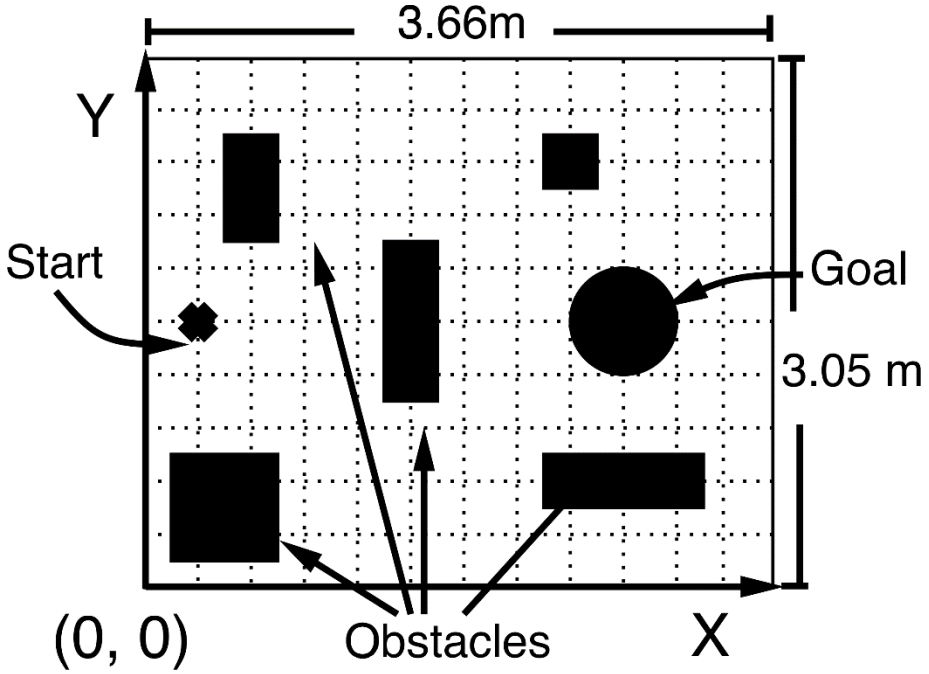
Due Date: October 29, 2024

Navigating a known Obstacle Course with the Lego Robot

The objective of this assignment is to navigate a mobile robot through an obstacle course to a goal location. The start position of the robot and the locations of all obstacles and of the goal are given before the robot begins.

The workspace is a rectangular area measuring 16’ x 10’ (4.88m x 3.05m). The lab space is tiled with 1’x1’ squares, creating a grid structure that simplifies navigation and planning. Obstacles are constructed from 1’x1’ cardboard squares and will be placed within the workspace. The goal is marked by a black circle with a radius of 1’ (0.305m).

The center of the goal area, the obstacles, and the starting position will align with the intersection points of four floor tiles. All elements will be oriented to match the grid layout of the tiles. An example of such an obstacle course is illustrated in the figure below.



Obstacles, goal, and start location of the robot will be chosen by the instructors and TAs, and the will not be known to the students before the testing date. The location of obstacles, start, and goal will be provided at compile/run time. Obstacles will be chosen such that there will always be a path that is at least 2 tiles (0.61 m) wide.

Given the obstacle locations, you will need to encode them in a way for your program to process. For example, in C you could encode them in a header file or preamble to your c-file by providing the coordinates of their centers. For the obstacle course shown above this could look as follows:

```
#define MAX_OBSTACLES   25                  /* maximum number of obstacles */
int num_obstacles = 13;                     /* number of obstacles         */
double obstacle[MAX_OBSTACLES][2] =         /* obstacle locations          */
{{0.61, 2.743},{0.915, 2.743},{1.219, 2.743},{1.829, 1.219},
 {1.829, 1.524},{ 1.829, 1.829}, {1.829, 2.134},{2.743, 0.305},
 {2.743, 0.61},{2.743, 0.915},{2.743, 2.743},{3.048, 2.743},
 {3.353, 2.743},
 {-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1},
 {-1,-1},{-1,-1},{-1,-1}};

double start[2] = {0.305, 1.219};           /* start location              */

double goal[2] = {3.658, 1.829};            /* goal location               */
```

You will be given a short period of time at the beginning to encode your specific obstacles.

At the end of the project, each group must hand in a report and give a short demonstration of their robot. During this demonstration, you should provide a short description of the robot and navigation system and be prepared to answer some basic questions.

The Project

1. Build a mobile robot for this task.

   Using the parts in your robot kit, build a mobile robot for the navigation task. Since the robot must be able to navigate through the course, an important design criterion might be that the robot is able to keep track of its position through tracking its motor speed or wheel encoder turns. The ability to detect the goal once it is reached or detect crossing an obstacle is not needed but could help.

   Your project report should include a short description of your robot design (including the critical design choices made).

2. Implement a navigation strategy to address the task.

   Implement a navigation strategy that will permit your robot to accomplish the navigation task with arbitrary obstacle, goal, and start configurations. We stress you will not know the obstacle locations before run time, so a path-planning algorithm is likely the best approach. We leave the specific implementation up to you.

   The speed of your robot and the length of the path generated by your robot are not of major importance here. The main objective is reaching the goal while not hitting any obstacles (i.e. without crossing over one of the obstacle tiles). In addition, the robot must stay within the assigned workspace.

   Your report should contain a description of the important components of your navigation strategy and the actual code for the robot.

Words of Caution

Dead reckoning (i.e. keeping track of the position of the robot using only internal sensors) for the Lego Robots can be relatively imprecise, and navigation strategies might therefore sometimes fail. Do not get discouraged if your robot does not succeed every time. Also, moving at slower speeds can improve the overall precision.