**Computer Project**

**Introduction to mobile robotics**

**Cédric Lécuyer**

**Lucile Gaurand**

**17.12.06**

**Summary:**

[Introduction 3](#_Toc499368300)

[I. Documentation of the algorithm 4](#_Toc499368301)

[1) Differential Drive Model 4](#_Toc499368302)

[2) The initialized code 4](#_Toc499368303)

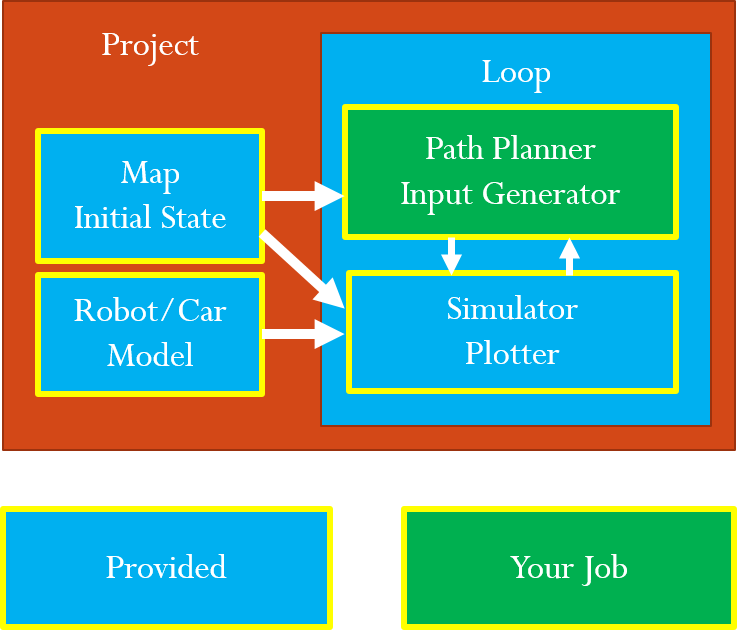
[II. Results of the program 6](#_Toc499368304)

[Conclusion 7](#_Toc499368305)

# Introduction

For the Mobile Robotics course, our term project consist in programming the navigation of a robot. Using MATLAB, we are going to simulate a mobile robot.

To create our mobile robot, we are going to use the differential drive with a simplified car model (round shape). The initial state will be a random position and a random orientation.

In order to have the complete simulator, we started with the programs given and we added our part. The map initial state, the car model, the loop and the simulator, plotter were provided. Hence we created the path planner.

Therefore, the aim is to control the car to park in a certain area while avoiding the obstacles.

# Documentation of the algorithm

## Differential Drive Model

* 5\*1 State vector
  + , = Robot position at reference frame X, Y axis
  + = Robot heading represented in reference frame
  + , = Robot wheel velocities
* 2\*1 Input vector
  + = Robot wheel accelerations
* System model()
  + Discrete time update:
  + ,

Wheelbase

## The initialized code

First and foremost, we have to create the obstacle map. This should be done in the userInit file. We inflate the obstacles to be sure that the robot will not touch obstacles as we are considering the center of the robot. Therefore, all the obstacles have outlines.

Then we have to complete the PlotAll file. We add the obstacle map and we can see if the path planning avoid obstacles showing that it is properly constructed. We add the path planning.

At that point, we can create the **path planning**. First we generate every nodes with the same distance between them. Next, we delete the nodes which are in obstacles. Then, we generate the path between each node. After this graph search, we delete the paths which cross obstacles. Finally we apply a path planning method.

We choose the **Dijkstra algorithm** to path plan.

To do the **path planning** following the Dijkstra method, we add the cost of each node in the path. The cost will tell how far the node is from the goal. Then we can swap the beginning and ending point, we begin at the goal and end at the start point. An end function when the goal is reached has to be added. To continue, we choose only the paths that goes from the beginning to the end. Finally, we choose the shortest path (should be done in the userInit file NOT TRUE).

We can come back to the PlotAll file and add different colour for chosen path.

According to path planning and robot position we can now implement the **robot control**.

Finally we can do the process of **code refactoring** by improving the code, correcting some mistakes and make the code more readable.

Our variables are:

# Results of the program

Here are the results of our program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Used Computer = Intel Atom D525 / 2GB RAM | | | | |
| **MAP#** | Pass/Fail | Path Planning Time  (real world time) | Running time  (simulation time) | Remark |
| **Easy** | P | seconds | seconds | - |
| **Moderate** | P | seconds | seconds | - |
| **Hard** | P | seconds | seconds | - |

# Conclusion

P