American University of Sharjah
College of Engineering
Computer Science & Engineering Dept.
P.O. Box 26666
Sharjah, UAE

Instructor: Dr. Michel Pasquier
Office: ESB-2057
Phone: 971-6-515-2883
E-mail: mpasquier@aus.edu
Semester: Spring 2024

CMP 494-10 INTELLIGENT AUTONOMOUS ROBOTICS

PROJECT #1

Due date/time: Sunday 10 March 2024, at 11 pm

Instructions: Submit via the assignment box on *iLearn,* before the deadline, a single *Zip/Rar* archive that contains your *models* and *explanations* in one *Word doc* and all your *Webots project folders.* Make sure to include the *name and AUS ID* of each teammate at the start of all your files. Include them in the archive name as well, as per this format: "CMP494-PR1-SThrun74321-OKhatib64213-DKumar83524.zip". Follow the instructions below and any other complement posted later on *iLearn.* Late submissions will be penalized as per the course policy.

Note that you are to complete this assignment *as a team of three or four students.* Furthermore, each team must *work independently* and hand in their own original answers. You are *not* allowed to discuss or *share* any solution or to *copy* from others or from any sourced material. Plagiarism and cheating will be severely penalized, starting with a zero grade for the assignment. Recall you are bound by the AUS Academic Integrity Code, which you signed when joining AUS.

Assignment: In this project, you are to program a mobile robot in simulation, so that it can reactively navigate a simple environment, as elaborated hereafter.

System setup: Download and install the Webots robot simulator from Cyberbotics.com, then get familiar with the software, if you have not done so already. All necessary features have been explained in class, but feel free to go through the tutorials again. The Webots user guide and reference manual are of course essential resources.

For each project stage hereafter, create a Webots project called “Project1”, “Project2”, etc. that will contain your *world* model and robot *controller/s* (and other folders as created by Webots). Note that projects have a lot in common so you may want to create the next one by duplicating then editing the current project.

Mobile robot: Throughout this project, you will use the *E-puck* robot available in Webots, which is a powerful little wheeled robot equipped with an array of range sensors, light sensors, a camera, as well as an accelerometer.

Additionally, you will need to equip the robot with a positional sensor such as a *GPS device,* as per the provided Webots GPS demo, in order to easily retrieve the robot's location. (It is a necessary assumption, because this project is not about localization, which is otherwise a complex issue.)

Lastly, you will need to add likewise a *pen device* to your robot, as per the built-in Webots pen demo, to allow tracing the many trajectories it will follow while you define and explore various behaviours. Make sure the pen is positioned properly and that you enable it in your code. Keep things as simple as possible (e.g., no need to change the pen color…)

Robot programming: Decide which programming language you will use throughout the project i.e., either Python (strongly recommended), C, or C++. (Using Java is not recommended due to various issues.) Note that, as was emphasized in class, your code should look very much the same regardless, as the same program logic should be implemented in all cases. What matters is the design, from architecture and decision logic to sensing and control directives, not the program syntax.

While your report will explain your design, models, etc. you are strongly advised to include appropriate comments in your robot controller code as well, to help the reader/instructor better understand and appreciate your effort and implementation.

Task environment: In this project, the robot's environment is a square room that contains a number of blocks representing obstacles. The environment is assumed to be static i.e., nothing will change, other than the robot itself moving around.

In each of the numbered stages hereafter, you will use the same environment, adding or removing obstacles as specified. You need therefore to build Webots worlds that match each of the 2D maps hereafter. The size of the square room should be about 20 times the size of the robot (i.e., 40 times the radius). This is to ensure the task of moving the robot around will be neither trivial nor impossible.

Each piece of furniture acts as an obstacle and can be modelled as simple block, as there is no need for details. Make sure, however, that the size and the position of each block matches closely what the maps show (use a ruler to measure). Lastly, the floor should be flat and have a uniform color (i.e., you should not use the chessboard-like floor seen in many demos, as this would be visually confusing).

Reactive behaviours: In each of the stages hereafter, you will write a robot controller program that implements the *Artificial Potential Field* (APF) approach, first one behaviour at a time, then integrating them all. Recall that, using APF, the velocity vector of your mobile robot is determined at each simulation step as the sum of the attraction force/s toward the goal and various repulsive forces away from the obstacles, as summarized below and illustrated in Figures 1 and 2 hereafter.

The direction of the attraction vector is given by the direction of the goal, relative to the current location of the robot. Its intensity can be constant, but it must be set carefully so that the robot is mostly attracted toward the goal when far enough from obstacles, and mostly repelled by them when too close.

Repulsive forces are calculated by measuring the distance to nearby obstacles, and their intensity increases as the robot gets closer from the object, and conversely.

The intensity may be inversely proportional to the distance, or more generally defined by a dropoff function. Another parameter is the distance threshold beyond which the force is always set to zero. A repulsive force is therefore created for each of the distance sensors on the robot. You should always test your model (and your controller program) by placing the robot at various locations, observing the resulting behaviour, and fine-tuning any model parameter/s as needed.

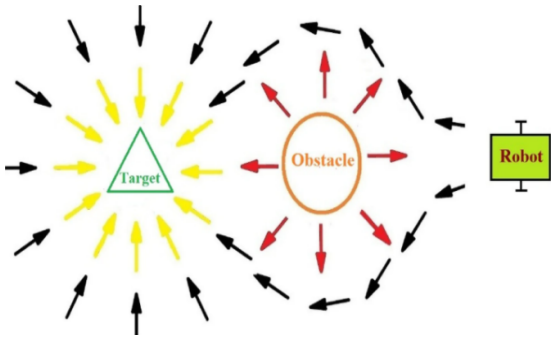

Figure 1: Attractive and repulsive forces.

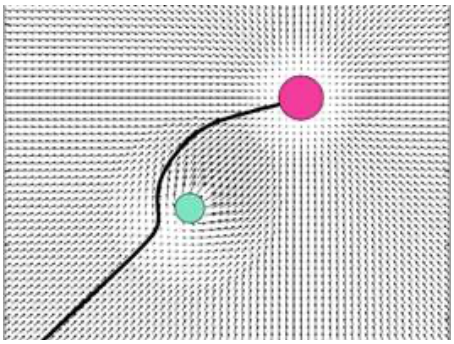

Figure 2: Example of a trajectory.

1. Goal destination: (a) Define an APF model that will *drive* the mobile robot *toward* the goal, regardless of initial position and orientation, and *stop* when (approximately) *at* that location. The goal destination should be located as per the map in Figure 1. Note that it is not a physical object and does not act as an obstacle. You can assume the goal location (X-Y cords on the map) is always known to the robot, and use the positional sensor to retrieve the robot's location. Explain your design clearly in the report, providing the necessary equation/s for the APF vector/s, dropoff, etc.

(b) Build a Webots world for the map in Figure 3 i.e., an empty arena surrounded by four walls, and implement the simplest controller program as per your design. Experiment in simulation by placing the robot at different initial locations and orientations. Include in your report screenshots (with trace) of your experiments, with appropriate comments, including how you refined your model parameters.

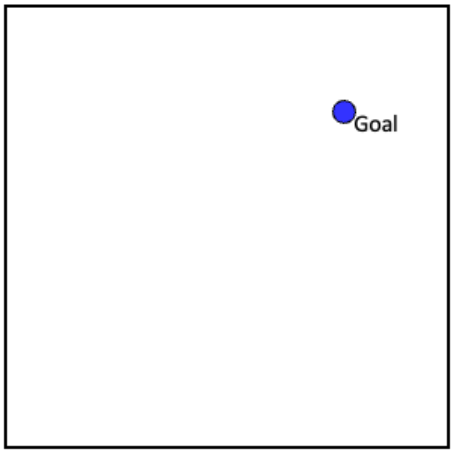

Figure 3: Location of the goal.

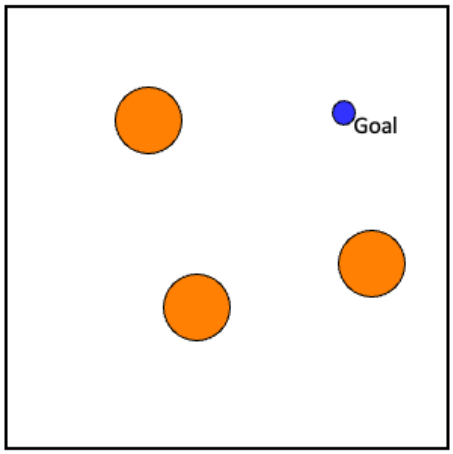

Figure 4: Adding three barrels.

2. Walls and barrels: (a) Define an APF model for each of the arena walls, so that the mobile robot will *not* collide into any of them. Explain your design clearly in the report, providing the necessary equation/s for the APF vector/s, dropoff, etc.

(b) Now, define an APF model for barrels i.e., cylindrical obstacles that are placed vertically on the floor, so that the mobile robot will move *away* from any of them, in addition to the walls. Explain your design clearly in the report, providing the necessary equation/s for the APF vector/s, dropoff, and other parameters as needed.

(c) Build a Webots world for the map in Figure 4, where three identical barrels are added to the arena. Each should be sized and positioned exactly as shown on the map above. Implement and test the simplest controller program as per your design. Experiment in simulation by placing the robot at different initial locations and orientations, including especially the three corners away from the goal. Provide in your report screenshots (with trace) of your various experiments, with appropriate comments, including how you refined your model parameters to make it work.

(d) Identify a starting location and orientation that would cause the robot to be *stuck,* either by not moving at all or by repeating the same "back and forth" motions… Include in your report a screenshot (with trace) of this scenario. Explain exactly why it is happening, how you could fix it, and what the pros and cons of your modified design are. (If you cannot identify such a situation, make sure the size of the robot, arena, and barrels are exactly as specified.)

(e) Explain the general idea of how we should modify the APFs to make the robot reach the goal destination as *fast* as possible, while avoiding collision of course. Similarly, explain the general idea of how we should modify the APFs so that the robot trajectories will be as *safe* as possible. For bonus marks, feel free to actually implement these scenarios, including in your report some screenshots (with trace).

3. Crowded warehouse: **(a)** Consider the map of Figure 5, which shows a warehouse with five barrels. Explain what new issue will appear that did not exist previously. Modify your APF model of the obstacles to address this issue as much as possible, and providing accordingly the revised equation/s for the APF vector/s, dropoff, etc. Indicate the pros and cons of the new design.

(b) Build a Webots world for the map in Figure 5, with the five identical barrels positioned exactly as shown. Implement and test a controller program as per your revised design. Experiment in simulation by placing the robot at different initial locations and orientations, including again the three corners away from the goal. Provide in your report screenshots (with trace) of your experiments, including comments about the issues faced and how you addressed them.

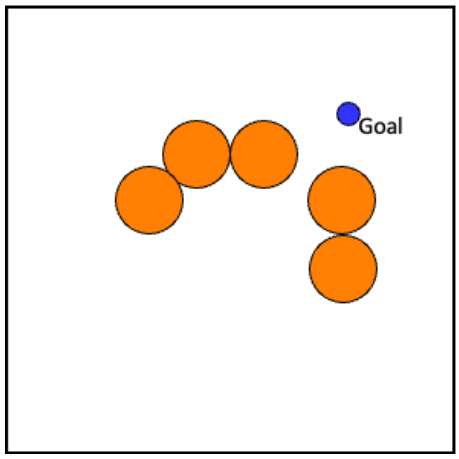

Figure 5: A simple warehouse.

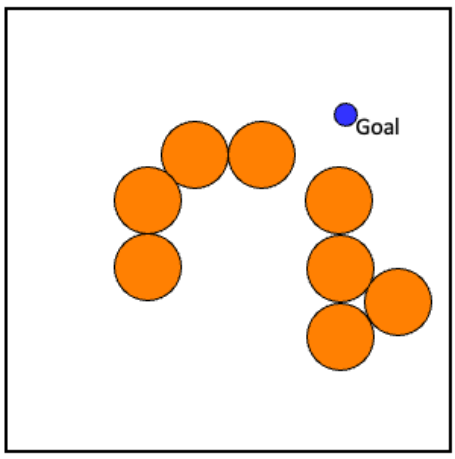

Figure 6: A crowded warehouse.

(c) Now, build a Webots world for the map in Figure 6, with the additional barrels positioned exactly as shown. Experiment in simulation with your model in part (b), by placing the robot at different initial locations and orientations, including again the three corners away from the goal. Provide in your report screenshots (with trace) of your experiments, then comment on what you see.

Explain how you could possibly modify your APFs so they still work in this more crowded environment. Indicate the pros and cons of the revised design, considering not only different starting positions but also different goal locations. For bonus marks again, feel free to actually implement these models, including in your report some screenshots (with trace).