Computer session on Autonomous Systems: Autonomous navigation and multi-robot coordination 3A ASI - Master MARS

Lara BRIÑÓN ARRANZ

2023-2024

Objectives

The objective of this computer lab is to generate a Matlab code to implement control strategies in order to deal with the following problems:

- autonomous navigation based on artificial potential fields
- consensus-based algorithms for multi-robot coordination
- flocking in multi-robot systems

In the first session (2h), we will design a potential field based strategy to control a robot in order to reach a goal position by navigating through obstacles. In the second session (2h), we will study the cooperative behavior of a multi-robot system which follows a simple consensus control rule. Then, in the last session (2h), both approaches, artificial potential fields and consensus, will be applied to a network of robots to achieve flocking.

Lab evaluation

This computer lab is composed of 3 sessions of 2h each. Groups of two students should be formed to work together during the three sessions.

For the first two sessions, a **written report** with the results (figures and analysis) obtained during the lab is expected for each group. These two reports will be part of the continuous evaluation of your work.

The reports of the first two lab sessions must be **sent at the end of the day of each lab session**, in PDF format, with the names:
BE_Navigation_NAME1_NAME2.pdf
BE_Consensus_NAME1_NAME2.pdf

The last lab session, will be a final test in which each group will produce a report during the lab.

The report of the final session must be **sent at the end of the session** with the name:
BE_Flocking_NAME1_NAME2.pdf

The three reports should be sent to:
lara.brinon-arranz@grenoble-inp.fr
ghadeer.shaaban@grenoble-inp.fr

1 Autonomous navigation with artificial potential fields

In this first exercise, the objective is to design a navigation strategy for a mobile robot to reach a goal position while avoiding obstacles in a two-dimensional space. The strategy to be designed and implemented for the autonomous navigation of the robot will be based on artificial potential fields. Firstly, we will implement a controller for a robot that moves following simple integrator dynamics. Secondly, we will design a new controller based on the same artificial potential field strategy to be applied to a non-holonomic robot.

Consider the following scenario: in a 2-dimensional space with an obstacle, the aim is to drive a robot to a goal position. The robot's position is denoted by $\mathbf{p} = [x, y]^T \in \mathbb{R}^2$, the goal position to be reached by the robot is defined by $\mathbf{p}_{goal} \in \mathbb{R}^2$ and the position of the obstacle is denoted by $\mathbf{p}_{obs} \in \mathbb{R}^2$. To guide the robot toward the goal position while avoiding the obstacle we define an artificial potential field composed of an attractive term and a repulsive one :

$$U(\mathbf{p}) = \alpha U_{attr}(\mathbf{p}, \mathbf{p}_{goal}) + \beta U_{rep}(\mathbf{p}, \mathbf{p}_{obs})$$

where the influence of each term can be set by tuning the parameters α and β.

Question 1: Artificial potential fields strategy (to be prepared BEFORE the first computer lab session)

1.1 Explain briefly the navigation strategy based on artificial potential fields and the role of the attractive and the repulsive potentials. Propose a simple potential field for both the attractive and the repulsive terms. Explain your choice.

1.2 Compute the gradient of your proposed artificial potential field $U(\mathbf{p})$. Explain how this gradient could be used to guide the robot to the goal.

1.3 If we consider that the obstacle is a point without dimensions, the distance vector from the robot's position to the obstacle can be expressed by $\mathbf{p} - \mathbf{p}_{obs}$. However, in real world applications, the obstacles have dimensions and are more than just a point in the space. Assuming that the obstacle has a 2-dimensional circular shape centered at $\mathbf{p}_{obs}$ and with radius R_{obs}, compute the distance vector from the robot position to the nearest point of the obstacle perimeter, denoted by $\mathbf{d}_{obs} \in \mathbb{R}^2$.

1.4 Express the gradient of the proposed artificial potential field $U(\mathbf{p})$ taking into account that the obstacle has a circular shape and using $\mathbf{d}_{obs}$ as described in the previous question.

Question 2: Autonomous navigation for a single integrator robot

First, we consider a robot with single integrator dynamics, i.e., the robot can move in any direction and we can control directly its velocity. The dynamic model of the robot is expressed as:

$$\begin{aligned} \dot{x} &= u_1 \\ \dot{y} &= u_2 \end{aligned}$$

where $\mathbf{u} = [u_1, u_2]^T$ is the control input.

For this section, use the Matlab code in the file `STUDENTS_Potential_simple_integrator.m` to implement your navigation strategy and controller for the single integrator robot.

2.1 Complete the code of the function `robot_dynamics.m` to implement the single integrator dynamics of the robot.

2.2 To study and analyze the artificial potential field approach in Matlab we have to implement the proposed navigation strategy in discrete time. To do so, the position of the robot at each step must be updated according to its dynamic model. If the sampling time is dt, write the equation to update the position of the robot at each step k using the function `robot_dynamics.m`
Hint: discretize the robot dynamics using $\dot{x}(t) \approx \frac{x(k+1)-x(k)}{dt}$.

2.3 To help you visualize the generated potential field the function `draw_field.m` is provided. Explain how this function works and the relation with the artificial potential field you proposed in Question 1.

2.4 In order to guide the robot to the goal while avoiding an obstacle, design a control law for the single integrator robot by defining a velocity vector reference to be followed, based on the artificial potential field that you proposed in Question 1.

Simulate your system with the following values:

- position of the static obstacle `p_obs=[4;4]`
- position of the goal `p_goal=[6;6]`
- robot's state initial conditions `x=[2;1]`
- potential function parameters `alpha=1` and `beta=1`

Analyze the robot behaviour when the position of the obstacle changes. Illustrate your results with a figure showing the initial and final state of the robot, the trajectory of the robot and the positions of both the goal and the obstacle.

Question 3: Autonomous navigation for a non-holonomic robot

Now, we consider a more complex model for the robot called unicycle. The non-holonomic robot dynamics are expressed as:

$$\begin{aligned}\dot{x} &= v\cos\theta\\ \dot{y} &= v\sin\theta\\ \dot{v} &= u_1\\ \dot{\theta} &= u_2\end{aligned}$$

where v is the robot's speed, θ its heading angle and u_1, u_2 the control inputs.

For this part of the computer lab, in order to implement your navigation strategy for the unicycle robot complete the Matlab code in the file `STUDENTS_Potential_unicycle.m`. Use the function `draw_robot.m` to draw your unicycle robot.

3.1 The state space of the robot is defined as $\mathbf{x} = [x, y, v, \theta]^T$. Modify the code of the function `robot_dynamics.m` to implement the unicycle dynamics of the robot.

3.2 To study and analyze the artificial potential field approach in Matlab we have to implement the proposed navigation strategy in discrete time. As in the previous case, write the equation to update the position of the robot at each step k using the function `robot_dynamics.m` with sampling time dt.
Hint: discretize the robot dynamics using $\dot{x}(t) \approx \frac{x(k+1)-x(k)}{dt}$.

In order to control the non-holonomic robot the following control strategy is proposed. With the help of the artificial potential field we design a reference velocity vector to be followed by the robot. We will design a proportional controller to make the robot's velocity track this reference velocity. There are two inputs to control the robot, linear acceleration and orientation rate, consequently, we will design two control laws as follows:

$$\begin{aligned}\dot{v} &= u_1 = -K_1(v - v_{ref})\\ \dot{\theta} &= u_2 = -K_2 \arctan\left(\tan\left(\tfrac{\theta-\theta_{ref}}{2}\right)\right)\end{aligned} \tag{1}$$

where v_{ref} is the magnitude of the reference velocity vector and θ_{ref} its orientation. For all the simulations, the controller parameters can be set as `K1=1` and `K2=10`.

3.3 In order to guide the robot to the goal while avoiding an obstacle, determine the desired velocity reference to be followed, based on the artificial potential field that you designed in Question 1 and define v_{ref} and θ_{ref} accordingly. Then, code the proportional controller proposed in Eq.(1).

Simulate your system with the following values:

- position of the obstacle `p_obs=[4;4]`
- position of the goal `p_goal=[6;6]`
- robot's state initial conditions `x=[2;1;1;`π`/4]`
- potential functions parameters `alpha=1` and `beta=1`

Analyze the robot behaviour for different values of the initial condition for the orientation θ (simulate for instance `x=[2;1;1;`π`]`). Illustrate your results with a figure showing the initial and final state of the robot, the trajectory of the robot and the positions of both the goal and the obstacle.

3.4 Analyze what happens when the parameter β related to the repulsive potential is 20 times greater than the parameter α related to the attractive potential.

Simulate your system with the following values:

- position of the static obstacle `p_obs=[5;5]`
- position of the goal `p_goal=[6;6]`
- robot's state initial conditions `x=[2;1;1;`π`/4]`

Illustrate your results with a figure showing the initial and final state of the robot, the trajectory of the robot and the positions of both the goal and the obstacle.

Question 4: More realistic scenario with circular shaped obstacles

4.1 Now, we consider a more realistic scenario where the obstacle is a circular shaped object. Following the same reasoning of Question 1.3, complete the code of function `distance_obs.m` in order to compute the distance vector from the robot's position to the nearest point of the obstacle.

4.2 Use the previously defined function `distance_obs.m` to modify the gradient of the potential function with a view to avoid obstacles with a circular shape.

Simulate your system with the following values:

- position of the center of the obstacle `p_obs=[5;5]`
- radius of the circular shaped obstacle `R_obs=0.2`
- position of the goal `p_goal=[6;6]`
- robot's state initial conditions `x=[2;1;1;`π`/4]`

Analyze the robot behaviour for different values of the radius of the circular obstacle. Illustrate your results with a figure showing the initial and final state of the robot, the trajectory of the robot and the positions of both the goal and the circular obstacle.

4.3 Consider now a more complex scenario with two obstacles. The center of the first obstacle is located at $\mathbf{p}_{obs}$ with radius R_{obs} and the center of the second one is located at $\mathbf{p}_{obs2}$ with radius R_{obs2}. Modify the gradient of the potential function to take into account both obstacles.

Simulate your system with the following values:

- position of the centers of both obstacles `p_obs=[4;4]` and `p_obs2=[6;4]`
- radius of the circular shaped obstacles `R_obs=0.2` and `R_obs2=0.2`
- position of the goal `p_goal=[6;6]`
- robot's state initial conditions `x=[2;1;1;`π`/4]`

Analyze the robot behaviour when the radius of the circular obstacles change. Illustrate your results with a figure showing the initial and final state of the robot, the trajectory of the robot and the positions of the goal and the two obstacles.