Exercise 4

Swarm control

Due date: April 1, 2018
Submit by Brightspace before midnight.

Course: SC42090 Robot Motion Planning and Control, TU Delft, DCSC
Contact: Linda van der Spaa, email: L.F.vanderspaa@tudelft.nl
Office hours: Wiener hall C-3-190, Thursday 22/3 13:00 – 14:00 (!), Thursday 29/3 10:00 – 11:00

1 Introduction

The goal of this exercise is to control a swarm of robots to follow patterns. In order to achieve this task, Lloyd's algorithm will be applied. The algorithm is described in the lecture slides.

The exercise consists of tasks that you have to do, but do not require a written answer, and questions that do require a written answer. Note: This time, no validation method is provided for the tasks. The questions to be answered are indicated by the number of points a correct answer will yield between parentheses. A maximum number of 20 points can be earned by completing this exercise, 10 points extra for the bonus part. To complete this exercise, hand in your answers to the questions using at most 2 pages (A4, 11pt). For the bonus exercise you are allowed two pages extra.

2 Matlab Implementation

In order to complete the implementation, you will have to fill out blanks (indicated by the TODO symbol) in the provided Matlab code.

Lloyd's algorithm consists of three parts:

1. Computation of the Voronoi diagram from the agents' positions
2. Computation of centers of mass of the Voronoi segments, which can be weighed by a density distribution to impose formations
3. Moving the robots towards the centers of mass of their Voronoi segments

The structure of Lloyd's algorithm is implemented in the function `Lloyd.m`. The Voronoi regions for each robot are computed in `Voronoi.m`. Of these regions, the centers of mass and moments around (0,0) are numerically estimated in `centroidNumerical.m`. Finally, moving the robots towards the centers of mass of their segments is split into two parts: a control law (`controlLaw.m`) and updating the robot positions as result of the applied control (`positionUpdate.m`).

Next to the initial positions of the robots in the swarm, the timespan of the control iterations and the necessary system parameters, `Lloyd.m` takes a mass distribution function `massFcn`. This function handle is passed on to `centroidNumerical.m` where it is used to weigh the areas of the Voronoi segments.

In the main Matlab script to this exercise, `main.m`, different mass distribution functions have been created to represent different swarm formations. Initially each 'pattern' has been given as an objective for the swarm for 2 seconds, after which the objective switches to the next formation.

Exercise 4.1: Centers of mass of the Voronoi partitions

Per robot, the Voronoi region computed by `Voronoi.m` is returned in `V`, which is a list of the corners of the Voronoi segment polygon. Numerical estimation of the area of the polygon can be achieved by dividing the square around the polygon into small squares and testing for the center of each square if it lies within the polygon.

Task: Define how the mass `Mv` and moment `Lv` of the Voronoi segment accumulate in lines `36` and `37` of `centroidNumerical.m`.

Exercise 4.2: Proportional control

Currently no control law is implemented in `controlLaw.m`. For now, we will assume the robots are holonomic and without inertia. A very simple control law is then to give them a proportional velocity in the direction in which they need to go.

$$\mathbf{v} = k(\mathbf{x}_{\text{desired}} - \mathbf{x}) \tag{1}$$

In this case $\mathbf{v}$ and $\mathbf{x}$ are vectors in 2D Cartesian coordinates.

Task: Compute $\mathbf{x}_{\text{desired}}$ from `Mv` and `Lv` obtained from `centroidNumerical.m` and implement control law (1) in `controlLaw.m` Choose $k = 5$. Make sure you also have a valid solution in case `Mv` and `Lv` are 0.

Exercise 4.3: Robot position update

Currently the control law assumes velocity can be imposed directly on the robots. In this simplest model, no dynamics of the robots need to be taken into account.

Task: Implement the position update law in `positionUpdate.m`, given the control input from the control law of the previous exercise.

3 Different ways of defining patterns

Now all blanks are filled, the `main.m` script can be run. After computing the robot trajectories for the five patterns specified, a window should appear showing an animation of the robot swarm. This animation can be toggled on and off by setting `options.animation` to 1 resp. 0. When the option `options.plot` is 1 (or any other nonzero number), the next three plots show respectively the Voronoi diagram to the initial swarm configuration, the Voronoi diagram to the final swarm configuration and the mass distribution to the final configuration.

Exercise 4.4: Patterns from pictures

So far the mass density functions were obtained from geometric functions defined in `simpleMassDensity.m`. An alternative is to extract mass density from an image. Three different mass density functions encode the stars of Fig. 1.

Task: Redefine `pattern5` using the function `imageMassDensity.m` with `'stars.mat'` as first argument.

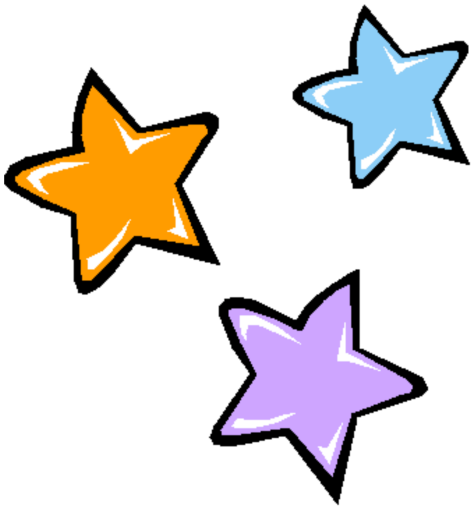

Figure 1: Stars

Exercise 4.5: Evaluating different weight functions

The function `imageMassDensity.m` takes a second argument which encodes for different types of mass density function, all describing the star pattern.

(6pt) Try the different options (1 to 3) of the second argument of `imageMassDensity.m`. What would happen in the different cases if you would let the simulation run for infinite time? Explain and relate your explanation to the shapes of the three mass density distributions. To illustrate, show a plot of the final configuration of each of the three cases.

The red squares mark robots that are in a Voronoi region with zero mass. This phenomenon occurs when providing the star pattern with the third option for type of mass density distribution.

(4pt) What do the robots do in that case? Explain the behavior of the robots marked with the red squares.

Exercise 4.6: Add robot dynamics

(3pt) Change the position update law to that of a differential drive vehicle. Show your equations. Note it will need a different kind of control input.

(4pt) Define an appropriate control law to control the robots with their changed dynamics. Show your equations and motivate your choice.

(3pt) Evaluate the behavior of the robot swarm (with a plot showing the paths of the robots) and discuss your observations.

Bonus Exercise 4.7: Obstacles

(4pt) Add obstacles to the environment and modify the mass distribution to be zero in the area occupied by the obstacles. Describe what you observe and include at least an image of the final configuration of the robots and the path followed.

(6pt) Implement and describe a method to avoid collisions with the obstacles. Describe what you observe, whether the solution is optimal and ways to improve it (you do not need to implement those improvements).