Lab 7 – Using potential field methods for obstacle avoidance

REPLAN team

Tuesday 8th April, 2025

Contents

compiled at: 08/04/2025, 06:53

The idea

We present the "leader-followers" architecture that allows us to control the behavior of a team of agents. The trajectory of the leader is obtained by applying the notion of potential field and the pursuers (the followers) are kept in formation by actions which penalize deviations from the positions and velocities ensuring the formation.

1 Theoretical background

The planning problem for a team of agents is complex because:

- the complexity of the computation increases with the number of agents involved;
- there are specific difficulties (task assignment, collisins avoidance between agents);
- there are structural constraints (communication, observation radius).

A popular solution is to implement a so-called *leader-followers* method:

leader is the agent whose task is to generate / follow a trajectory that allows to reach the destination;

followers they have simple command laws that allow them to follow the leader and maintain a formation (usually with a predefined structure).

So far, the leader's trajectory has resulted from procedures such as those described in Lab 2 (the trajectory is generated offline and then tracked online) or in Lab 3 (the trajectory is calculated online based on current information).
We consider an "integrator" system[1], characterized by position ($p \in \mathbb{R}^n$) and velocity ($v \in \mathbb{R}^n$), these being the state, respectively the control action of the system:

$$\dot{p} = v. \tag{1}$$

Next, we define a *potential field* composed from:

repulsive components For a collection of obstacles with centers c_i we apply a law proportional to the distance from them (radial repulsion):

$$\phi\left(\|p - c_i\|\right) = \frac{a_1}{a_2 + \|p - c_i\|}, \quad \phi\left(\|p - c_i\|\right) = \begin{cases} \frac{a_1}{a_2}, & p = c_i, \\ 0, & \|p - c_i\| \to \infty, \end{cases} \tag{2}$$

attractive component In order to direct the agent to the destination, we penalize the distance from it (linear, exponential, etc.):

$$\psi\left(\|p - p_d\|\right) = \|p - p_d\|. \tag{3}$$

[1] At the next level of complexity we consider the "unicycle" or "Dubins car" models.

The total potential field consists of the sum of these components:

$$P(x) = \psi\left(\|p - p_d\|\right) + \sum_i \phi\left(\|p - c_i\|\right), \tag{4}$$

and the command applied to the system is (in the simplest case), proportional to the gradient of the potential field at the point corresponding to the current position:

$$u = -\nabla P(p) = -\left(\frac{p - p_d}{\|p - p_d\|} + \sum_i \frac{-a_1}{\left(a_2 + \|p - c_i\|\right)^2} \cdot \frac{p - c_i}{\|p - c_i\|}\right). \tag{5}$$

To ensure a formation we have at our disposal several methods (depending on the nature of the agent's dynamics), its sensors (relative or absolute positioning, coverage radius) and the communication graph between agents.

Considering N "double-integrator" agents, characterized by state (composed from position – $p \in \mathbb{R}^n$ and velocity – $v \in \mathbb{R}^n$) and control action (acceleration – $u \in \mathbb{R}$):

$$\dot{p}_i = v_i, \quad v_i = u_i. \tag{6}$$

Each agent has a list of neighbors $\mathcal{N}_i$ to which it is linked by weights $\omega_{ij} \geq 0$ (zero value appears if agents i, j are not neighbors).

Adjacency relations (the graph of interconnections of agents) are characterized by the " Laplacian matrix ":

$$L = [\ell_{ij}]_{i,j=1:N}, \text{ with } \ell_{ij} = \begin{cases} \sum\limits_{k \in \mathcal{N}_i} \omega_{ik}, & \text{if } i = j, \\ -\omega_{ij}, & \text{if } i \neq j. \end{cases} \tag{7}$$

To define a formation, relations between positions and relative speeds with respect to the neighbors are required:

$$p_j - p_i = p_j^\star - p_i^\star, \quad v_j - v_i = v_j^\star - v_i^\star. \tag{8}$$

These are enforced through suitable choices of the control actions

$$\begin{aligned} u_i = &- k_p \sum_{j \in \mathcal{N}_i} \omega_{ij}(p_i - p_j - p_i^\star + p_j^\star) \\ &- k_v \sum_{j \in \mathcal{N}_i} \omega_{ij}(v_i - v_j - v_i^\star + v_j^\star), \end{aligned} \tag{9}$$

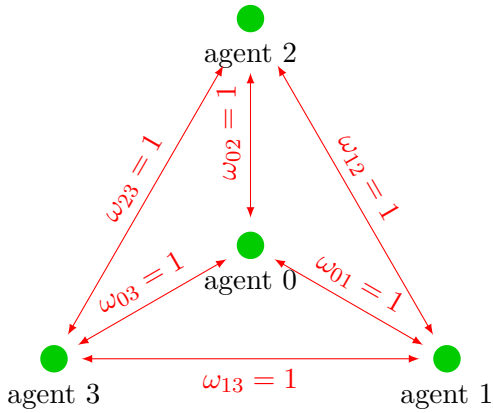
where $k_p > 0$, $k_v > 0$.
The evolution of position and speed errors, respectively $e_p = p^\star - p$, $e_v = v^\star - v$ is governed by dynamics

$$\begin{bmatrix} \dot{e}_p \\ \dot{e}_v \end{bmatrix} = \begin{bmatrix} 0 & I_{nN} \\ -k_p(L \otimes I_n) & -k_v(L \otimes I_n) \end{bmatrix} \begin{bmatrix} e_p \\ e_v \end{bmatrix}, \tag{10}$$

where $'\otimes'$ denotes the Kronecker product.

2 Implementation of a formation control algorithm

We consider 1 + 3 agents that we want to bring in a triangle configuration: in the center is positioned the leader (agent 0) and the followers (agents 1, 2 and 3) are positioned relative to it and follow his speed, as is outlined below:



The Laplacian of the above graph, as per definition (7), is

$$L = \begin{bmatrix} 4 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}, \text{cu } \mathcal{N}_1 = \{0,2,3\}, \mathcal{N}_2 = \{0,1,3\}, \mathcal{N}_3 = \{0,1,2\}. \quad (11)$$

and the positions and speeds that define the trajectory are:

$$p_0 = \Big(0,0\Big),\ p_1 = \Big(\frac{\sqrt{3}}{2}, -\frac{1}{2}\Big),\ p_2 = \Big(0,1\Big),\ p_3 = \Big(-\frac{\sqrt{3}}{2}, -\frac{1}{2}\Big), \quad (12a)$$

$$v_0 = \Big(0,0\Big),\ v_1 = v_2 = v_3 = \Big(0,0\Big). \quad (12b)$$

With these elements we now build the (9) commands that apply to the agents:

```
control=control-kp*(agent[i-1]['position'][-1,:]-agent[j-1]['position'][-1,:]-p[i,:]+p[j,:])-kv*(agent[i-1]['velocity'][-1,:]-agent[j-1]['velocity'][-1,:]-v[i,:]+v[j,:])
```

This later allows us to increment [2] the systems' state (position and velocity):

```
agent[i-1]['position']=np.vstack((agent[i-1]['position'], agent[i-1]['position'][-1,:]+T*agent[i-1]['velocity'][-1,:]))
agent[i-1]['velocity']=np.vstack((agent[i-1]['velocity'], agent[i-1]['velocity'][-1,:]+T*control))
```

[2] Although in (6) we consider continuous time dynamics, in the code we implement the discretized version.

3 Proposed exercises

Exercise 1. The code associated with this lab only implements the formation control assumption that the leader is at rest. Modify the code so that the leader follows an arbitrary trajectory.

Exercise 2. Modify the code (including the version obtained in the previous exercise) for the case of the Dubins car whose dynamics are given by the equations:

$$\begin{cases} \dot{x} &= u_V \cos\phi, \\ \dot{y} &= u_V \sin\phi, \\ \dot{\phi} &= \frac{u_V}{L} \tan u_\phi. \end{cases}$$

Exercise 3. Solve the following requirements:

i) For a given collection of obstacles (defined by their centers) and a destination, construct and illustrate the potential field defined in (4);

ii) Starting from a random starting point, apply the command (5) to system (1). Illustrate the resulted trajectory;

iii) Consider a group of agents to whom you apply the control action from the previous item but where, in addition, you add a collision avoidance term:

$$\sum_{i \neq j} \zeta\left(\|p_i - p_j\| - d_{ij}\right) = \sum_{i \neq j} \left(\|p_i - p_j\| - d_{ij}\right)^2.$$

This term maintains the distance between agents i and j at $d_{ij} > 0$. Illustrate the resulting trajectories.