Unmanned Ground Vehicle control through potential field

Robotics and automation

This project aim at controlling a mobile robot trajectory toward an objective, avoiding obstacles that can appear, disappear and move in a given area. For the simulation, we will define the environment of the vehicle and use Matlab to create the space and simulate the robot behavior. Therefore, it will be made in two steps. The first step is the creation of the space using a potential field representation of the space, and the second step will be the simulation of the robot trajectory towards the target using ode23 to solve the dynamics and calculate the sum of the forces at each new position.

Contents

Environment generation

A- Introduction

As mentioned in the introduction, we need to think about the space that the vehicle will move in. Its specificities are given in the exercise and aim at a Matlab simulation. The work area is a 14 by 14 square (from (0, 0) to (14, 14)) in the (x, y) plan. We consider the target at the position (12, 12) and two obstacles at (5, 6) and (7, 6).

For the autonomous vehicle to reach the target we gave him without hitting the obstacles, we need to design an algorithm that will find a path in the area. To do so, we will represent the space as a potential field. The idea is to be able to generate a repulsive force for the obstacles and an attractive one for the target, this way each location in the space will be able to sense each force. That will make possible to calculate a sum of the forces and to create a direction for the robot to follow. We will talk about the robot movement in part II. Using this method, we can play on the accuracy (scale of each area's cell), the obstacles or target importance (need to avoid from far away if it is big or dangerous) and we can update the map between each computation of the robot's position without any issues or heavy processing.

Before going on some mathematics, trying to have a good idea of what we want to do in terms of physics is always interesting. To simulate the forces, we will give a potential value to each point in the space so that we can simulate the action of a force. To illustrate this with an example, it is like slightly making a slope with a pipe to bring the water to the position we want without using additional pressuring system. Thanks to the difference of potential (which is related to the height of each point of the pipe in this example), we are able to create movement by using the gravity of the Earth, and the liquid will go from the highest part to the lowest (the gravity point towards "-z"). However, the force is not existing in our situation, we create it by generating a variation of an arbitrary value, so that the obstacles got a really high height when there is a slope on all the plan towards the target.

The relation between a force $\vec{F}$ in the plan (x, y) and its potential V is given by:

$$\vec{F} = -\overrightarrow{grad}\, V = \begin{pmatrix} -\frac{\partial V}{\partial x} \\ -\frac{\partial V}{\partial y} \end{pmatrix}$$

We could only calculate the sum of the forces created by those sources at the robot position; however, we want to see what it looks like if we extend it to the overall space, which allow us to be sure it is working correctly (and maybe some real time display of the mapping if needed in the application). Therefore, we will compute the potential field of the area.

B- Computing the potential field

The next step is to find the expression of those attractive (target) and repulsive (obstacles) forces' potentials. Further explanations are given later in part I and II.

An attractive potential is given by:

$$V_T = k\,r$$

And a repulsive potential is given by:

$$V_O = \frac{k}{r}$$

With k a given constant and r the distance between the target and the robot. This distance r is given by:

$$r = \sqrt{(x_T - x)^2 + (y_T - y)^2}$$

So, we can define the potential associated with the target V_T and the potential related to the obstacles V_O:

$$V_T = k\sqrt{(x_T - x)^2 + (y_T - y)^2} \quad and \quad V_O = \frac{k}{\sqrt{(x_0 - x)^2 + (y_0 - y)^2}}$$

Therefore, if we want to calculate the overall potential of a point, we can sum all together in the following expression:

$$V_{tot} = \frac{k_{O,1}}{r_{O,1}} + \frac{k_{O,2}}{r_{O,2}} + k_T r_T$$

Making the sum of the forces gives us a third force, that is basically induced by the total of the potentials that we just expressed. Calculating V_{TOT}=$V_T + V_{O,1} + V_{O,2}$ gives us the value of the potential on each point.

C- MATLAB generation

We will proceed as follow:

- Set the position of the target and the obstacles;
- Set the target's and the obstacles' weight (*k*);
- Cut the area in a sufficient number of cells;
- Compute the value potential for each of those cells;
- Draw a 3D plot of the potential regarding the (x, y) position.

We obtain the following potential field:

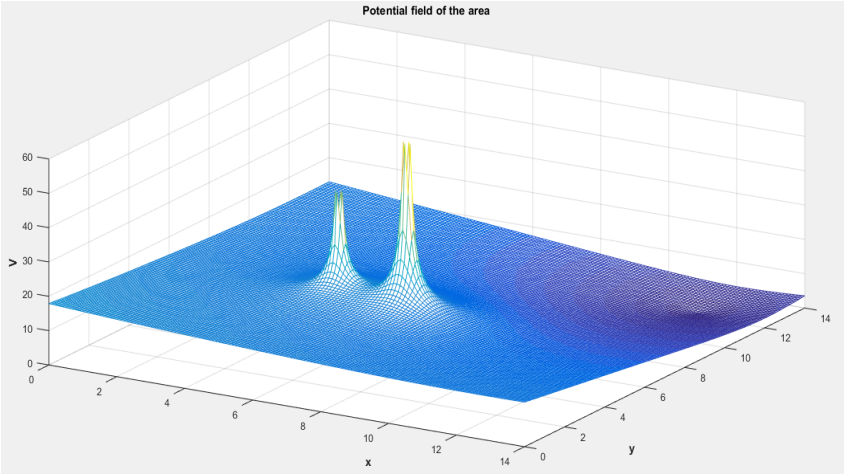

Figure 1: 3D plot representing the potential field for the area we are working on.

We can have a better idea from the top using a level's line representation:

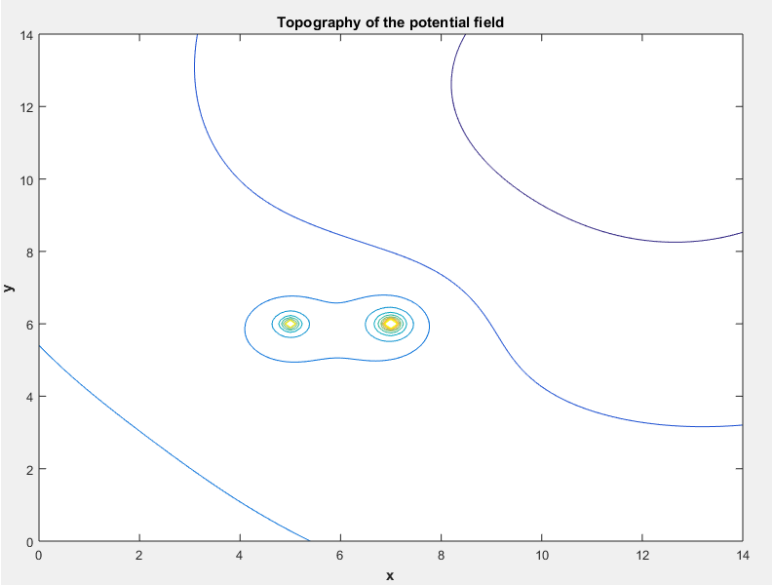

Figure 2: Topological representation of the potential field of the area.

On each representation, yellow is the heighest potential, whereas dark blue is the lowest. It's important to mention that this is not a real representation of the area. The environment is flat and the obstacles are not necessarily peaked rocks or things alike (it could even be holes). This representation is a tool we use to compute a path to the target.

At the end we can see that the expressions of the potential we used allowed us to create a smooth potential field regarding the position of each point. Everything is linked, and we can see the attractive source generate a linear force that does not depends the distance from the source to the point and the repulsive force is an inverted squared function of r which means that the force decreased according to the distance.

An interesting point is the second blue line on the topographic plot, we see around the point (8, 8) how the repulsive potential is changing the "same potential" line, compensating the decreasing potential with higher values because of the obstacle position.

Potential field navigation

Now that we are familiar with the environment we are moving in, we want to make our vehicle reach the target. To do so we need two things. First, we need to find the system dynamics, which will allow us to simulate the movement of the robot. Then, we need to find the sum of the forces that apply on the vehicle from the different obstacles and the target. The direction of the resultant of the forces is the direction our robot needs to go to.

A- The robot's dynamics

For this problem we chose a robot that have three wheels, two on the back are controlling its speed and the front one can turn to control the direction. The following scheme of the vehicle in a 2D-space describe the variables of the problem:

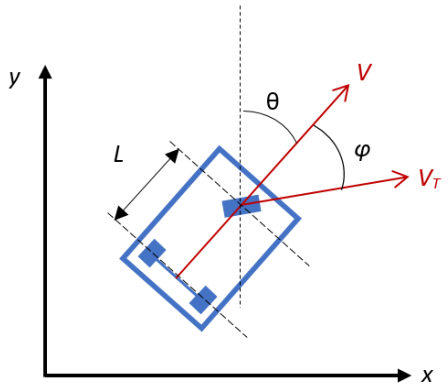

Figure 3: Schematic presentation of the vehicle's situation.

On this scheme, ***L*** is the wheel base, ***V*** is the speed of the robot, V_F is the speed of the front wheel, which is defined by the steering angle **ϕ**, which is defined by the orientation of the vehicle, given by **θ**.

We can express the speed of the vehicle using the steering angle ϕ as the following:

$$V = V_F \cos\varphi$$

$$\dot{\theta} = \frac{V_F}{L} \sin\varphi$$

Indeed, a variation of θ is induced by an angle $\varphi \neq 0$. We can then give the expression of a variation of x and a variation of y:

$$\begin{cases} \dot{x} = V \sin\theta \\ \dot{y} = V \cos\theta \end{cases}$$

$$\begin{cases}\dot{x} = V_F \cos\varphi \sin\theta \\ \dot{y} = V_F \cos\varphi \cos\theta\end{cases}$$

Actually, our front wheel is not helping to speed up the robot, only to direct it. That means, its speed's value is the same as the speed of the overall robot given by the back wheels, and we can right $V = V_F$ which gives us the following equations to describe the system:

$$\begin{cases}\dot{x} = V \cos\varphi \sin\theta \\ \dot{y} = V \cos\varphi \cos\theta \\ \dot{\theta} = \dfrac{V}{L}\sin\varphi\end{cases}$$

That's the way we find the equations given in the setup of the exercise.

For the simulation, we will fix the speed $V = 4$ and $L = 2$.

B- Resultant of the forces on the vehicle

Now we want to process the sum of the forces induced by obstacles and target on the vehicle.

We will start by calculating the forces regarding the expression of the potential given in the first part. We describe F_{O1} and F_{O2} the forces induced by the obstacles 1 and 2 respectively, and F_T the force induced by the target.

$$F = \begin{pmatrix}F_x \\ F_y\end{pmatrix} = \begin{pmatrix}-\dfrac{\partial V}{\partial x} \\ -\dfrac{\partial V}{\partial y}\end{pmatrix}$$

Which gives us the following expressions for the repulsive forces:

$$\begin{cases}F_{Oi,x} = -k_i \dfrac{\partial}{\partial x}((x_0 - x)^2 + (y_0 - y)^{-1/2}) \\ F_{Oi,y} = -k_i \dfrac{\partial}{\partial y}((x_0 - x)^2 + (y_0 - y)^{-1/2})\end{cases}$$

$$\begin{cases}F_{Oi,x} = -k_i(x_0 - x)((x_0 - x)^2 + (y_0 - y)^{-3/2}) \\ F_{Oi,y} = -k_i(y_0 - y)((x_0 - x)^2 + (y_0 - y)^{-3/2})\end{cases}$$

$$\begin{cases}F_{Oi,x} = \dfrac{-k_i(x_0 - x)}{r^3} \\ F_{Oi,y} = \dfrac{-k_i(y_0 - y)}{r^3}\end{cases}$$

And for the attractive force:

$$\begin{cases}F_{Oi,x} = -k_i \dfrac{\partial}{\partial x}((x_0 - x)^2 + (y_0 - y)^{1/2}) \\ F_{Oi,y} = -k_i \dfrac{\partial}{\partial y}((x_0 - x)^2 + (y_0 - y)^{1/2})\end{cases}$$

$$\begin{cases} F_{Oi,x} = k_i(x_0 - x)((x_0 - x)^2 + (y_0 - y)^{-1/2}) \\ F_{Oi,y} = k_i(y_0 - y)((x_0 - x)^2 + (y_0 - y)^{-1/2}) \end{cases}$$

$$\begin{cases} F_{Oi,x} = \dfrac{k_i(x_0 - x)}{r} \\ F_{Oi,y} = \dfrac{k_i(y_0 - y)}{r} \end{cases}$$

Note: Here we can make a link to the potential field we created earlier. In those equations of the forces, we have two different terms. One of them gives the projection of the force on the x or y axis. If we pay attention to this projection term, we realize the core part is $(x_0 - x)$*. This value is normalized by r, which means that one* r^{-1} *is included in the "projection" term.*

Therefore, the intensity of an attractive force is constant regarding k and does not vary regarding the distance, when the repulsive force is decreasing following a $\frac{1}{r^2}$ *function. Also, the attractive force is positive, which means it is from the vehicle to the target, when the repulsive force is negative (from the obstacle to the vehicle).*

Now that we have the expression of the forces, we can sum them all together to have the resultant of it:

$$F_{TOT} = F_{O,1} + F_{O,2} + F_T$$

Once we have the x and y component of the force, we can get its direction and therefore define the expression of the steering angle φ. We call this angle α and it is defined by:

$$\alpha = tan^{-1}\left(\frac{F_{TOT,y}}{F_{TOT,x}}\right)$$

C- Control system

Now that we have the angle the robot needs to aim at, we need to control the positioning of the wheel so that the steering angle φ match the given angle α.

To do so, we will design a feedback control system. The system will take the position of the robot (at each new position processed) to compute the potential field and the different force that apply on it. Therefore, the global force is updated as long as the angle α. Then, this angle is compared to the actual orientation of the robot, θ. This error signal is then processed by a proportional controller with a gain K. Which gives the robot the actual angle he needs to point to regarding its actual orientation. The system's dynamics allow us to compute the variation of x, y and θ that we can use to get a position at a time and therefore continue the process until we reach the target. This can be represented as the following sketch:

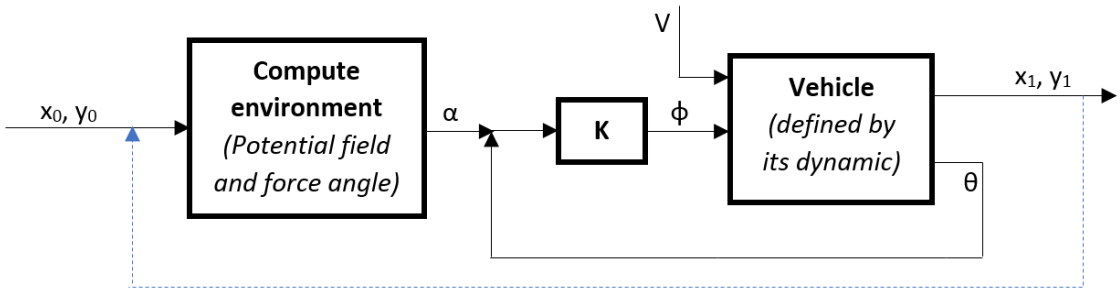

Figure 4: Sketch of the control system

Following this scheme, we have the relation between the command angle $\alpha$and the value that is given to the system:

$$\varphi = K(\alpha - \theta)$$

We will start with K=3.

MATLAB simulation

The most important part to discuss now is the simulation of the vehicle position over the time. To do this, we use ode23 to approximate the system's dynamic solution (first order derivative). Ode23 is a single step solver using Runge-Kutta (2,3) method.

Regarding the MATLAB's help, ode23 needs several arguments:

- The expression of the state variables $\dot{x}, \dot{y}$ and $\dot{\theta}$:

What we give here to ode23 is not exactly the expression of the state variables. Instead we implement a function that is going to calculate the sum of the forces at the robot actual position, use it to calculate a steering angle α, process φ, and gives the expression regarding the newly calculated φ. The thing is, ode23 calls the function for each new solving of the system's equation. Therefore, every time we have the new value of the steering angle regarding the new position, and regarding the eventual modifications of the space.

- The time span of the experiment:

This represent the interval of integration, formulated as a vector (t_0 t_f). t_0 corresponds to the time at which we have the initial conditions.

- The initial conditions of the system.
- Some options:

The function we put there is defined using odeset(). We use it to stop the process once the vehicle has reached the target.

We want the robot to start from the south-west of the area (0, 0) with an orientation angle of $\theta = {}^{\pi}/_{3}$. After running the simulation with a time span from 0 to 100, we got the following positions of the robot over the area:

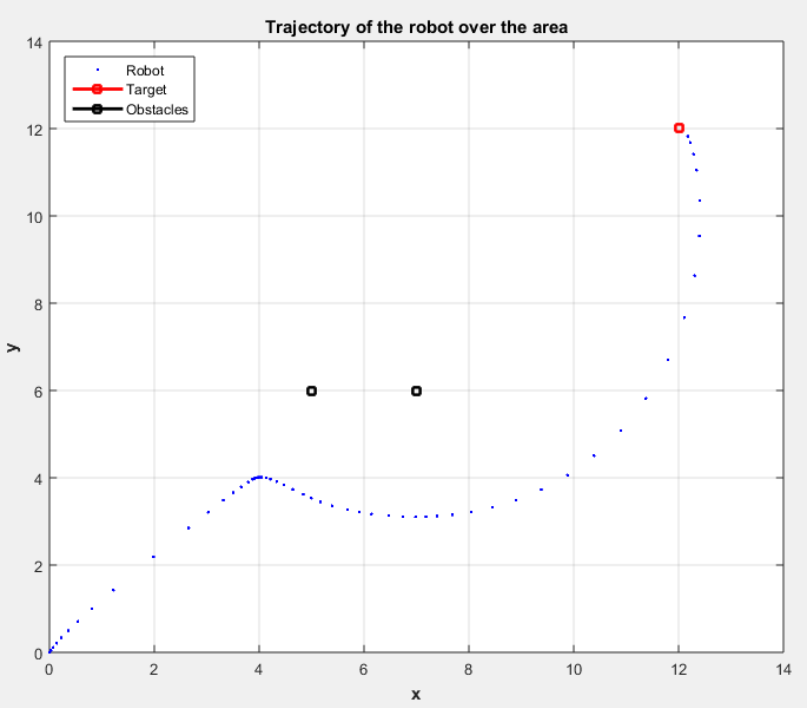

Figure 5: Trajectory of the vehicle in the area from the initial conditions (bottom left) to the target (top right).

This plot is pretty interesting since we can see how the speed of the robot is changing over the time when the obstacles force makes the steering angle change (which change the variation of x and y over time). This is represented by the space we have between two points. Since the time has a constant step, a variation of the distance induces a variation of the speed (a bigger space means an acceleration where a smaller one describes a deceleration).

We can also draw the continuous trajectory over the potential field by calculating the potential value for each simulated position of the robot and drawing them on the 3D plot of part 1:

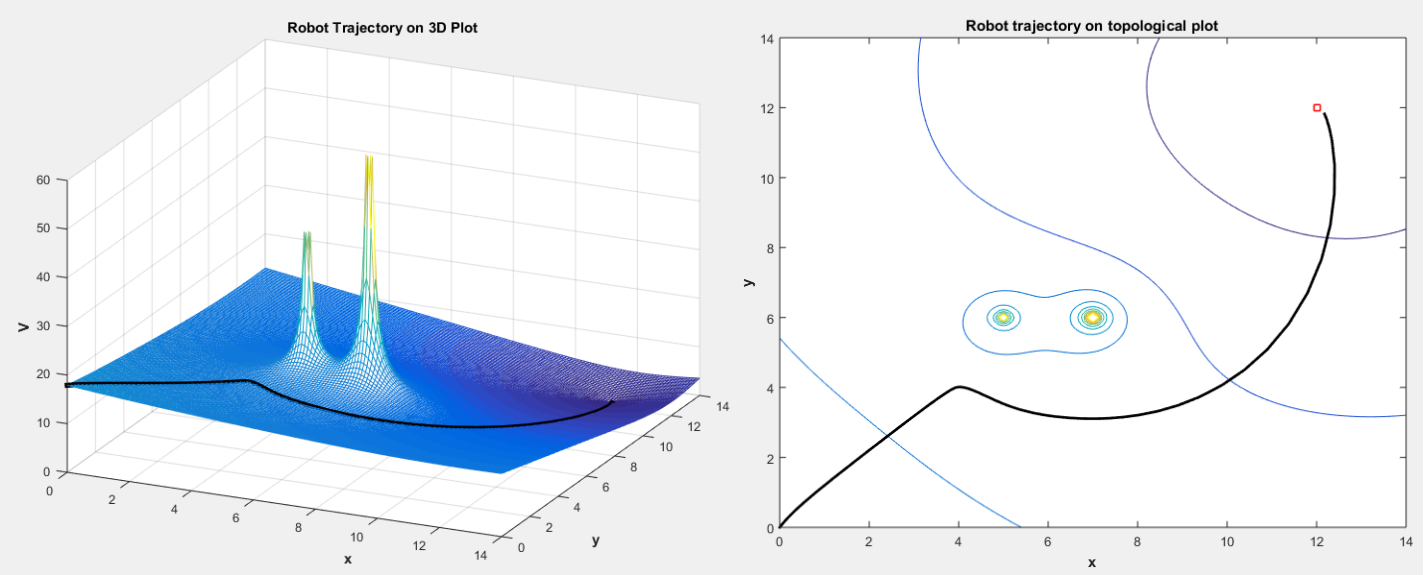

Figure 6: Trajectory of the vehicle over the potential field that maps the area, in 3D plot on the left and on a topological graph on the right.

The simulation is automatically stopped after the robot reached the target.

Conclusion

By mapping the environment as a potential field and describing the dynamics of the vehicle, we are able to give simple rules to the robot to avoid any obstacles on his way to reach the target. Using this gradient method allow several advantages in terms of tunability, since all of the factors that are use either to create an understandable environment or to control the robot trajectory are tunable. Indeed, by adjusting the gain on the obstacles or by changing raising the resolution of the map we can create more precise obstacles and shape them to fit a real situation. It is also possible to consider the importance of one obstacle and the degree of avoidance we need. A point that most of the pathfinding method does not complete correctly is the update of the environment. Since we compute it at each new position, the robot has a real-time mapping of the area with a good rate since the process is pretty fast.

The next step is obviously to generate a map that changes itself with the time, with obstacles or target moving in the area. For this situation, the challenge would be to design the controller so that the robot anticipates what's ahead of him to avoid troubles if any obstacles is rushing towards its direction for example. Then, creating a system where couple of robots are moving together to one target can be an interesting point, where the squadron is following one leader. Deeper studies on this part are mostly about the managing issues as in the case where the leader is not able to fulfill is role anymore, or if one agent of the squad is not operational anymore.