Magnetic mapping using a trailed magnetometer

Quentin Brateau[1,✉]

[1]ENSTA Bretagne, Brest

The realization of a magnetic map of a terrain is a powerful tool especially used in mine warfare. This cartography can be done with a magnetometer, but the task is quite slow. This is why it is useful to use robots to make this cartography. The problem induced by this solution is the addition of magnetic disturbances related to the structure of the robot and its actuators. It is then possible to drag the magnetometer on a sled behind the robot.

Magnetic | Mapping | Control | Interval Analysis
Correspondence: ***quentin.brateau@ensta-bretagne.org***

Formalism

A. Mathematical context. The system evolve on a field which should be mapped with the magnetometer. It follows the evolution describe by EQUATION 1.

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t)) & (evolution\ equation) \\ \mathbf{y}^i = g(\mathbf{x}(t_i)) & (observation\ equation) \end{cases} \tag{1}$$

where $\mathbf{x}$ is the state of the robot, $\mathbf{u}$ is the input of the system. As $\mathbf{x}$ and $\mathbf{u}$ continuously evolve with time, we define the trajectories of this system denoted by $\mathbf{x}(\cdot)$ and $\mathbf{u}(\cdot)$. We could also define tubes enclosing these trajectories at every time denoted by $\forall t, [\mathbf{x}](t)$ and $\forall t, [\mathbf{u}](t)$. Measurements are available at each t_i and are denoted by $\mathbf{y}^i$. Therefore it is assumed that $\forall t, \mathbf{u}(t) \in [\mathbf{u}](t)$ and that for each measurements $\mathbf{y}^i \in [\mathbf{y}^i]$. This last conditions means that each measurement is bounded which is an assumption made frequently when we are dealing with constrain programming.

B. State equations. The system is composed of a vehicle that will drag a sledge that transports a magnetometer. The towing vehicle is a tank type vehicle, and the sled is attached to it with a rope. Assuming that the rope is constantly under tension, we are able to find the equations describing the dynamics of the system.

$$f(\mathbf{x}(t), \mathbf{u}(t)) = \begin{cases} \dot{\mathbf{x}}_1 = \frac{\mathbf{u}_1 + \mathbf{u}_2}{2}.cos(\mathbf{x}_3) & \textbf{(2a)} \\ \dot{\mathbf{x}}_2 = \frac{\mathbf{u}_1 + \mathbf{u}_2}{2}.sin(\mathbf{x}_3) & \textbf{(2b)} \\ \dot{\mathbf{x}}_3 = \mathbf{u}_2 - \mathbf{u}_1 & \textbf{(2c)} \\ \dot{\mathbf{x}}_4 = -\frac{\mathbf{u}_1 + \mathbf{u}_2}{2}.sin(\mathbf{x}_4) - \dot{\mathbf{x}}_3 & \textbf{(2d)} \end{cases}$$

Here $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are respectively the abscissa, the ordinate and the heading of the trailing robot, $\mathbf{x}_4$ is the angle of the sled to the tractor vehicle, as shown in FIGURE 1. We can also notice that $\frac{\mathbf{u}_1 + \mathbf{u}_2}{2}$ is related to the speed control of the char robot and that $\mathbf{u}_2 - \mathbf{u}_1$ is the angular acceleration control.

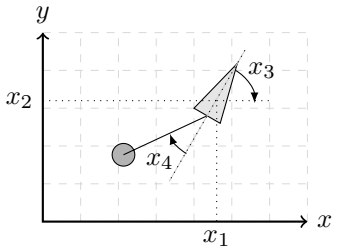

Fig. 1. Diagram representing system state variables

Constrain Network

C. Decomposition. These equations could be broken down into the following set of elementary constrains :

$$\begin{cases} (i) & \mathbf{v}(\cdot) = f(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \\ (ii) & \dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \\ (iii) & \dot{\mathbf{v}}(\cdot) = \mathbf{a}(\cdot) \\ (iv) & \mathbf{d}(\cdot) = \mathbf{x}_{1,2}(\cdot) - \mathbf{x}_{\mathbf{m}}(\cdot) \\ (v) & \mathbf{t} = \mathbf{x}_3 + \mathbf{x}_4 \\ (vi) & \mathcal{L}_{polar}\left(\mathbf{d}(\cdot),\ [L],\ \mathbf{t}\right) \\ (vii) & \mathcal{L}_{eval}\left(t_i,\ [\mathbf{y}_{1,2}],\ \mathbf{x}_{1,2}(t_i),\ \mathbf{v}_{1,2}(t_i)\right) \\ (viii) & \mathcal{L}_{eval}\left(t_j,\ [\mathbf{y}_{4,5}],\ \mathbf{v}_{1,2}(t_j),\ \mathbf{a}_{1,2}(t_j)\right) \end{cases} \tag{3}$$

These constrains involve intermediate variables introduced for ease of decomposition: $a \in \mathbb{R}$, tubes $\mathbf{v}(\cdot) \in \mathbb{R} \to \mathbb{R}^4$ and $\mathbf{d}(\cdot) \in \mathbb{R} \to \mathbb{R}^2$

Simulator

A python simulation was realized using VIBES in order to test the behavior of the system. A class *Tank* has been created to instantiate a vehicle with its sled. Then the script integrates the evolution equation using Euler's method in order to obtain the state of the system x according to the u inputs. We could see that the behavior of the system seems correct and the model is faithful to reality. Moreover, the GNSS sensor and an accelerometer are simulated in order to enclose the real position in a box.

Reliable set of sled's angle

Considering that the rope remains continuously under tension while the robot is moving, then the evolution of the angle x_4 of the sled relative to the towing vehicle is described by the *Differential Equation* **??**.

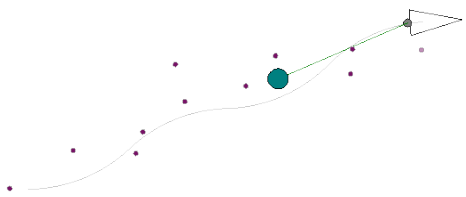

Fig. 2. Simulation of the system

We are able to find the trajectory of the sled by computing the interval containing the angle x_4, considering that initially x_4 belongs to $[-\pi/2;\pi/2]$. Then by applying the *Differential Equation* **??** on this interval, knowing the control vector u, we end up obtaining a fine interval framing the real angle x_4, independently of the initial angle as we can see on the FIGURE 3.

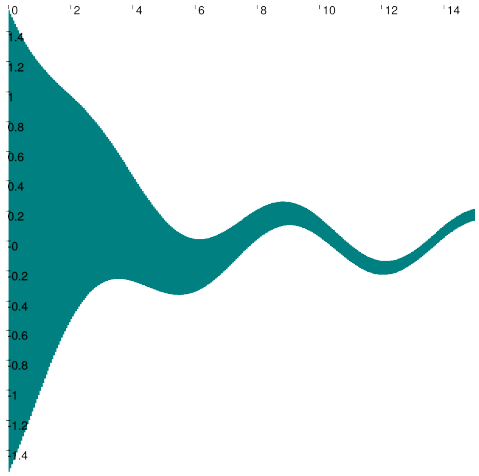

Fig. 3. Integration example using intervals

Sled localization

By having a box enclosing the trailing robot, and an interval framing the angle x_4, we are able to obtain a box containing the sled, knowing the length of the rope.
This box has the shape of a pie sector. *Equation* 4 presents the position of the magnetometer as a function of the state of the system x_1, x_2 and x_4. Here all these variables are intervals. By applying a polar contractor to these intervals, we are able to obtain the intervals containing x_m and y_m.

$$\begin{cases} x_m = x_1 - L.cos(x_4) \\ y_m = x_2 - L.sin(x_4) \end{cases} \quad \textbf{(4)}$$

Magnetometer range of measurement

By knowing a box containing the magnetometer and its measuring range, we are able to find all the points seen for sure and all the points that could be seen by the magnetometer. This allows us to recover the coverage of the area by the magnetometer.

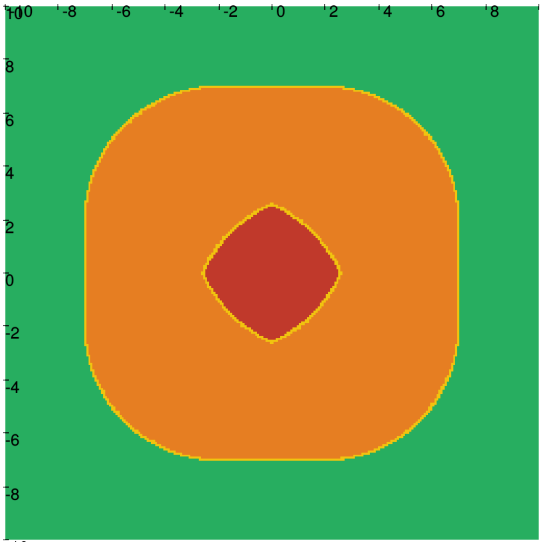

Fig. 4. Measurement range of the magnetometer using Thicksets

The FIGURE 4 shows us these two sets. The green set is the unseen area, the orange set is the area maybe seen and the red set is the area seen for sure by the magnetometer. The yellow set forms the uncertain boundary between the other sets, which can be adjusted. Thus we have a way to know precisely the coverage of the mapping during the mission.

Magnetometer Mapping

With a moving box enclosing the magnetometer at any time, this method allows us to recover the coverage of the area by the magnetometer. Thus we have a way to know precisely the coverage of the mapping during the mission, which depends on the uncertainty on the position of the magnetometer and on the maximum distance of each point of the field to be measured. An example of the covered area of a magnetometer on a field is presented by the FIGURE 5.

Fig. 5. Coverage of the map by the magnetometer using Thicksets

Application example

In this part we will see on a practical case what we are able to do with this implementation. We place ourselves in the same conditions as before, i.e. we are in the case of a magnetometer dragged by a robot with tank type evolution equations and the rope must remain tight all along the mission.
Here we want our robot to follow the following Lissajous trajectory:

$$\forall t \in \mathbb{R}, \qquad \mathcal{T}(t) = \begin{cases} x(t) = 20 \times sin\left(\frac{2 \times t}{20}\right) & \textbf{(5)} \\ y(t) = 20 \times sin\left(\frac{3 \times t}{20}\right) & \textbf{(6)} \end{cases}$$

$$\forall t \in \mathbb{R}, \qquad \frac{d\mathcal{T}(t)}{dt} = \begin{cases} \frac{dx(t)}{dt} = 2 \times cos\left(\frac{2 \times t}{20}\right) & \textbf{(7)} \\ \frac{dy(t)}{dt} = 3 \times cos\left(\frac{3 \times t}{20}\right) & \textbf{(8)} \end{cases}$$

$$\forall t \in \mathbb{R}, \qquad \frac{d^2\mathcal{T}(t)}{dt^2} = \begin{cases} \frac{d^2x(t)}{dt^2} = -\frac{4}{20} \times sin\left(\frac{2 \times t}{20}\right) \\ \frac{d^2y(t)}{dt^2} = -\frac{9}{20} \times sin\left(\frac{3 \times t}{20}\right) \end{cases} \textbf{(9)}$$

(10)

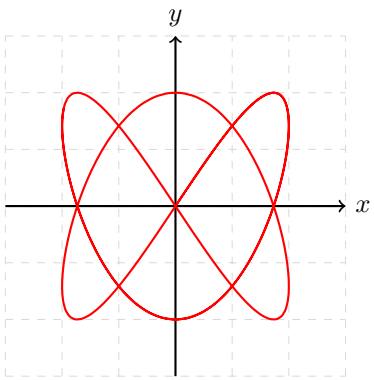

Fig. 6. Lissajous trajectory

The *control.py* file is able to generate the controls to drive the trailing robot. These commands are generated by feedback linearization by controlling the robot in speed and heading. These controls leads to the trajectory in the simulator as we could see on the FIGURE 7
Then the module *mapping.py* allows to process the tubes including the trajectory of the trailing robot and the sledge. The tubes are then displayed in order to check that they seem to stick to the trajectory of the system.

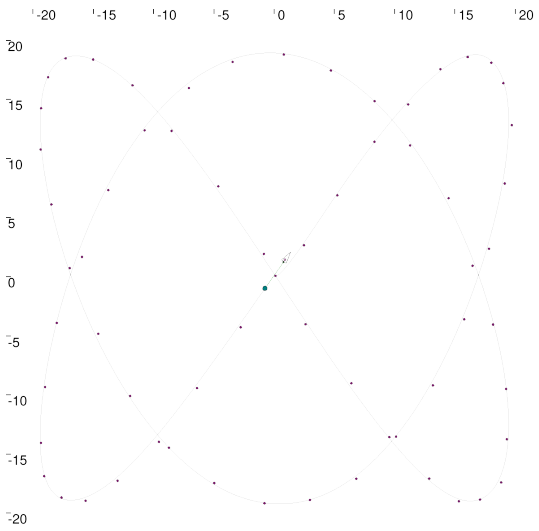

Fig. 7. Simulator's trajectory of the system

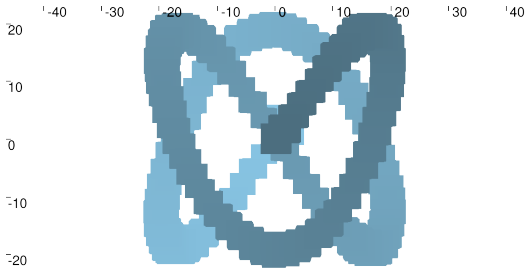

Fig. 8. Trailing vehicle enclosing tube

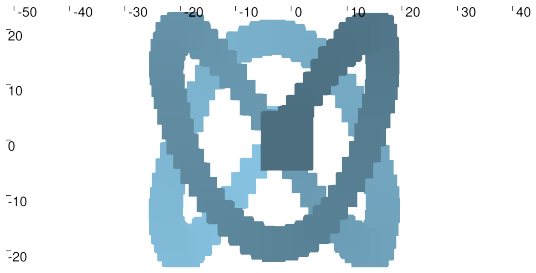

Fig. 9. Magnetometer enclosing tube

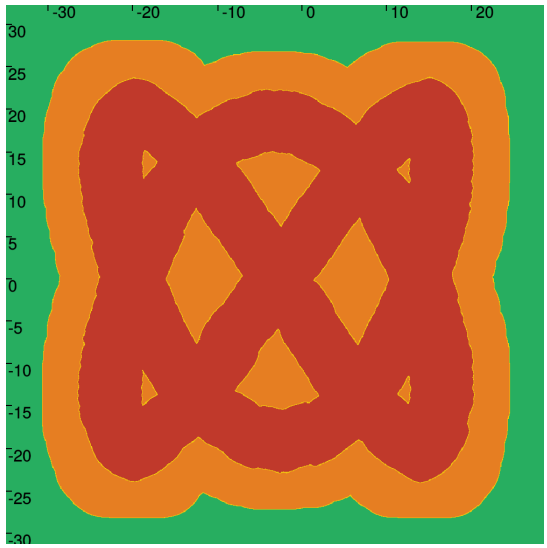

Fig. 10. Magnetometer enclosing tube