Report on particle filtering

Author :
M. Romain DUSSOT

Supervisors :
Dr. Alexandru STANCU
Dr. Mario MARTINEZ GUERRERO
Dr. Eduard CODRES
Dr. Mohamed MUSTAFA

romain.dussot@ensta-bretagne.org - 0(+33)6.35.25.36.93 - Version 1.0
October 1, 2018

Résumé / Abstract

For my internship in the University of Manchester, I have worked alone on my project. The subject was not clear at the beginning, only the support and the theme were defined. My project was to apply firstly a particle filter on a ground robot. Then, if the time left is enough, I had to apply a SLAM method on the robot based on the particle filter previously done. In this report, I will explain first all the knowledge I have on the particle filter, then I will show some simple examples I created for a better understanding of the filter and its advantages. Finally, I will explain the application of my work and the future possibilities to go further. After this technical part, I will present briefly the Robotics department of the University of Manchester.

Contents

Chapter 1

Introduction

In this report, I will present the work I have done during my internship. The project itself has been created using a type of robot and a type of sensors. With a ground robot equipped with a raspberry Pi and a LIDAR, my supervisor proposed the subject:

" Research and application of a particle filter on the localisation of a ground tank-robot using a LIDAR and the ROS Operation System "

The first part of the report and of my internship is divided in two parts, on one hand some explanations on the theory of the particle filter, on the other hand some simple exmaples and some interesting results of such a filter. Then, on the second part of the report, I will introduce the way I used to apply my particle filter on the ground robot.

Chapter 2

Objectives and Context

2.1 Subject

As written in the introduction, the subject is:

" Research and application of a particle filter on the localisation of a ground tank-robot using a LIDAR and the ROS Operation System "

To go further in the subject and to explain the support I had during my internship, I will describe some parts of the subjects and show the resources I had.

2.2 Objectives

To be able to apply a particle filter, I had to make some researches on the subject. I will explain in the next part of my report the result of my researches. After some researches and a meeting with Mohamed MUSTAFA, I had a better understanding of the particle filter. I had to apply the theory to some simulation coded in Python. The goal was to translate some global representation of the algorithm to practical function and application. For example, the "Spreading" (explanation in chapter 2) step of the algorithm is not so clear to implement, so I had to think how to apply it in different contexts.

Afterwards, I had to apply my simulation in Python to a real robot and I had to translate my code in C++. Unfortunately, my knowledge in C++ was not enough to apply easily and fastly my simulation to the robot. Furthermore, I had to implement some new libraries in ROS to permit me to communicate properly in my robot.

2.3 Context

For my internship, I have worked in the University of Manchester and more specifically the "School of Electrical and Electronic Engineering".

The department has been created in 1905 by A. Schwarz and managed to produced a series of discovers, for example the first stored-program computer. Nowadays, the researchers are working on many different fields:

- Materials and devices;
- Electronique engineering for agriculture;
- resilient energy systems;
- Robotics for extrem environments;
- Sensing technologies for security.

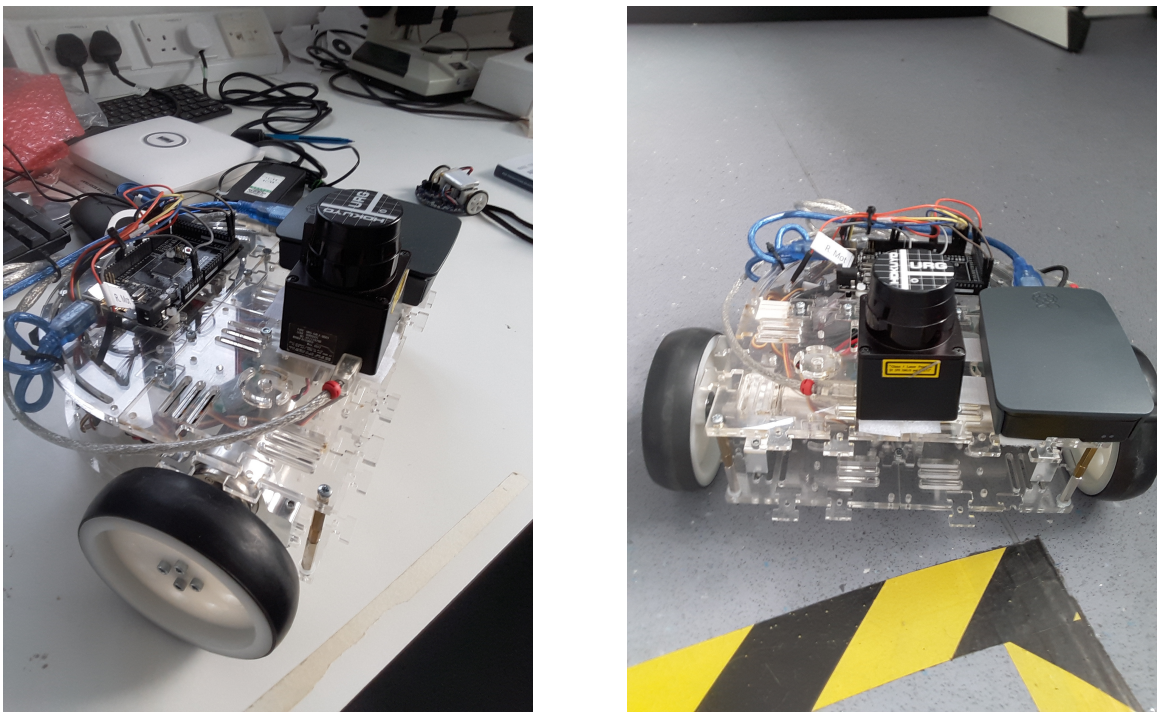


Figure 2.1: Pictures of the PuzzleBot

During my internship, I worked with the *Dr* Alexandru Stancu in the specialty Robotics for extrem environment. According to the multiple possiblities of robot types from submarines to drones, I could work on a ground robot. I have worked on a prototype of a PuzzleBot.

This robot is composed with:

- 1 Raspberry Pi 3 model B with ROS;
- 1 Arduino module to supply the power;
- 2 DC motors with encoders;
- 1 LIDAR Hokuyo;
- a "puzzle" support made from multiple pieces.

This robot is the perfect example for the economic part of the report.

2.4 Economy of the department

As a student, I did not have access to any kind of budget from my supervisors. They explained me that they had to sell their knowledge or projects to companies. The department offers some possibilities of advertising or project building. The PuzzleBit is one of the best example, a student wanted to create some basic pieces to build a robot quickly. He has created the PuzzleBot during his PhD and the department sold it to a company to have funds for other projects. This is basically the only way to have funds. The University of Manchester do not possess money directly but all the different departments of the University have their own economy and have to find some funds to work.

Chapter 3

The theory about particle filtering

3.1 Quick Explation and Principle

To be able to understand the equations, I firstly watched a video to understand how could the filter work, I used the video posted by Andreas Svennson on Youtube which is brief and concise. That was the perfect video as an introduction to the particle filter that I discovered as this moment. [5]

3.2 Explanation on particle filtering

This type of filter is a probabilistic approach to solve a state space problem. The algorithm is based on the Montecarlo methods and the law of large numbers to be applied. This filter can be applied on many different fields, this report will be focused on the localisation of a robot thanks to measurements such as distances with a LIDAR. All these steps has been designed thanks to many french and english articles. [2] [3] [6]

To be applied:

- There is something we want to know as a position;
- We measure something related to what we want to know as a measure;
- We know something about the relationship between the measurements and what we want to know with the datasheet of the sensor ot the hypothesis.

If we apply this model to a real state space problem:

- We want to know the location of a robot.
- We can measure the distances between the robot and the walls and obstacles.
- We know the map which means the position of the walls and obstacles, then we can figure out the position of the robot.

We define the dynamic state and the measures

$$X = \{x0, ..., xk, ...\} \ \& \ Y = \{y0, ..., yk, ...\}$$

.

The dynamic state is defined as :

- a known state;

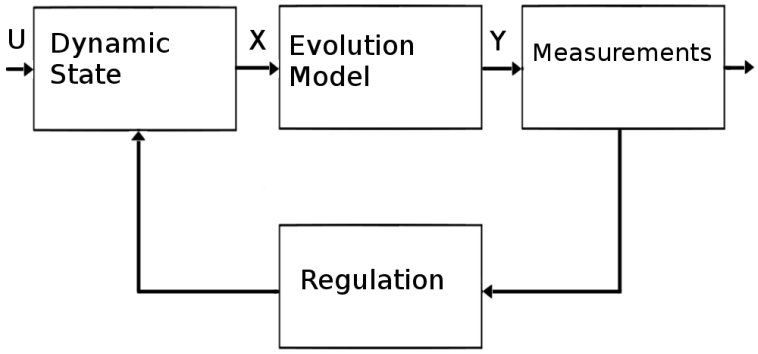


Figure 3.1: State of the robot

- the initialisation is : $p(x_0)$;
- the transitive equation is : $p(x_{k+1}/x_k)$;
- where x_k is a Markov chain.

The observation is defined as :

- a known state;
- the measure : $p(y_k/x_k)$
- thanks to the property of the Markov chain

$$p(y_k/x_k, y_{k-1}, y_{k-2}, ...y_0) = p(y_k/x_k)$$

Now that we have defined the context of the problem, we can consider the filter available in this case. Our problem is not a linear problem. So there are many filters available to solve the problem, as the Extended Kalman Filter (EKF). In this project, we will focus on the particle filtering and its way to solve non-linear problems. To apply it, we will use some hypotheses.

- The probability density, at a time k, a posteriori of the state knowing the measurements is approximated by a discrete random approximation;
- This discrete random approximation is defined by N particles placed in the interesting zones of the space (intrusting means places with no walls or obstacles at the beginning);
- The support of a particle is a possible trajectory of the state;
- The weight of a particle is the estimation of the density of likelihood.

3.3 Implementation of the particle filter

3.3.1 The steps

Initialisation: First step of the algorithm, we have to introduce all the types of the variables, sets, etc. We define the N particles. We use a uniform distribution of our particles on our map i.e. all the particles have a random x, y and θ and a weight of $\frac{1}{N}$. Furthermore, all the particles must have a possible initial state. A particle can neither start directly on a wall nor a obstacle hence the hypothesis on the perfeclty known map.

Spreading: In this step, we will move every particle. Because each particle is a possible state of the robot and also a possible "robot", we will apply the same evolution model and the same control than the robot itself. Nevertheless, we will add some noise to the control because we know that the reliability of the robot in relation to the control is not perfect. This noise will permit to reduce this error thanks to a certain number of particle.

Update weights: Now that the model has been applied to every particle, we will update the weights with a significance function S.

$$S(x_k/x_{0\to k-1}^{(i)}, y_{0\to k})$$

This function permits to find the evolution of the particle weight by using the comparison between the measurements from the robot and from the particle. For a better understanding, a gaussian function centered in 0 applied on the difference between the measurements is possible [4].

Estimation: At this point, every particles have a new weights that described the likelihood for a particle to be placed as the real robot. Thanks to these weights, we can calculate the estimate position of the real robot with a weighted average or with the calculation of the minimum mean square estimation (MMSE).

Redictribution: In this step, we will normalize our weight vector. The principle is to use the particle with the bigger likelihood to divide it into many particles with the same probability. A particle with a huge weight can be divided into a local uniform distribution of particles. Thanks to this step, at the end of the loop, we will have the same number of particles than at the beginning and we can restart the loop. We will compare the sum of the squared weight to a threshold under which we could lose some information or accuracy. Then we will store the old weights, resize the particle into many and put them into a local uniform area around the latest location of the particle.

3.3.2 The algorithm

Initialisation: We start with (complexity $O(N)$)

$$\{x_0^{(i)}\}_{i\in[1,N]} \simeq p(x_0)$$

Loop

Spreading (complexity $O(N)$)

$$\{x_k^{(i)}\}_{i\in[1,N]} \simeq S(x_k/x_{0\to k-1}^{(i)}, y_{0\to k})$$

Update Weights (complexity $O(N)$)

$$w_k^{(i)*} = w_{k-1}^{(i)} * \frac{p(y_k/x_k^{(i)} p(x_k^{(i)}/x_{k-1}^{(i)})}{S(x_k/x_{0\to k-1}^{(i)}, y_{0\to k})}$$

Estimation We use the minimum mean square estimation (complexity $O(N)$)

$$\hat{x}_{k,MMSE} = \sum_{i=1}^{n} w_k^{(i)} x_k^{(i)}$$

Redistribution (complexity $O(N)$)

$$If(\sum_i (w_k^{(i)})^2)^{-1} < T : \begin{cases} p(x_{0\to k}^{(i)new} = x_{0\to k}^{(j)old}) & = & w_k^{(j)old} \\ w_k^{(i)new} & = & \frac{1}{N} \end{cases}$$

Chapter 4

Simple examples

4.1 1D example

Firstly, for a better understanding of the particle filtering, we will work on a basic 1D example. We will localise a robot with the measure of the height of the ceiling above the robot. In this context, some of the problems of the particle filter can be shown. [1]

4.1.1 Context

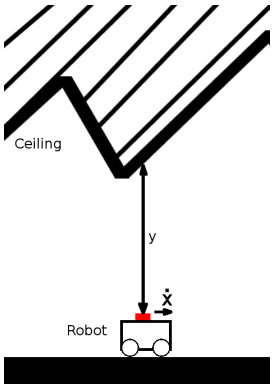


Figure 4.1: Context of the 1D context

On this example, we will apply the particle filter and change some parameters to see the behaviour of the localisation.

Here the hypothesis used on this example are:

- the robot can only move from left to right on the room;
- the ceiling is known;
- noise has been added to the measure from the sensor and for the simulated measure from the particle;

To understand all the figures, the left part represents the ceiling and the floor with blue lines, the position of the robot with the red dot and the position of particles with the black dots. On the left image, the balck dots reprensents only the position of all the particle excluding the different weights. The aim is to see if the filter manages to localise properly the robot. On the right side of the figures, the weights of all the particles is plotted, this is $w = g(x)$ where g is a function reprensenting the weight of each particle knowing the position of these particles. The way chosen to show the results is the point cloud.

4.1.2 With a non-symetric ceiling

We will define a ceiling define with $ceil(x) = 10 + x$ to apply the filter at the beginning and be sure that the filter works on this case.

On the figure, the left board represents the ground and the ceiling with a black point cloud representing the position of tall the particles. On the right board, the weights of the particles are represented function of the position.

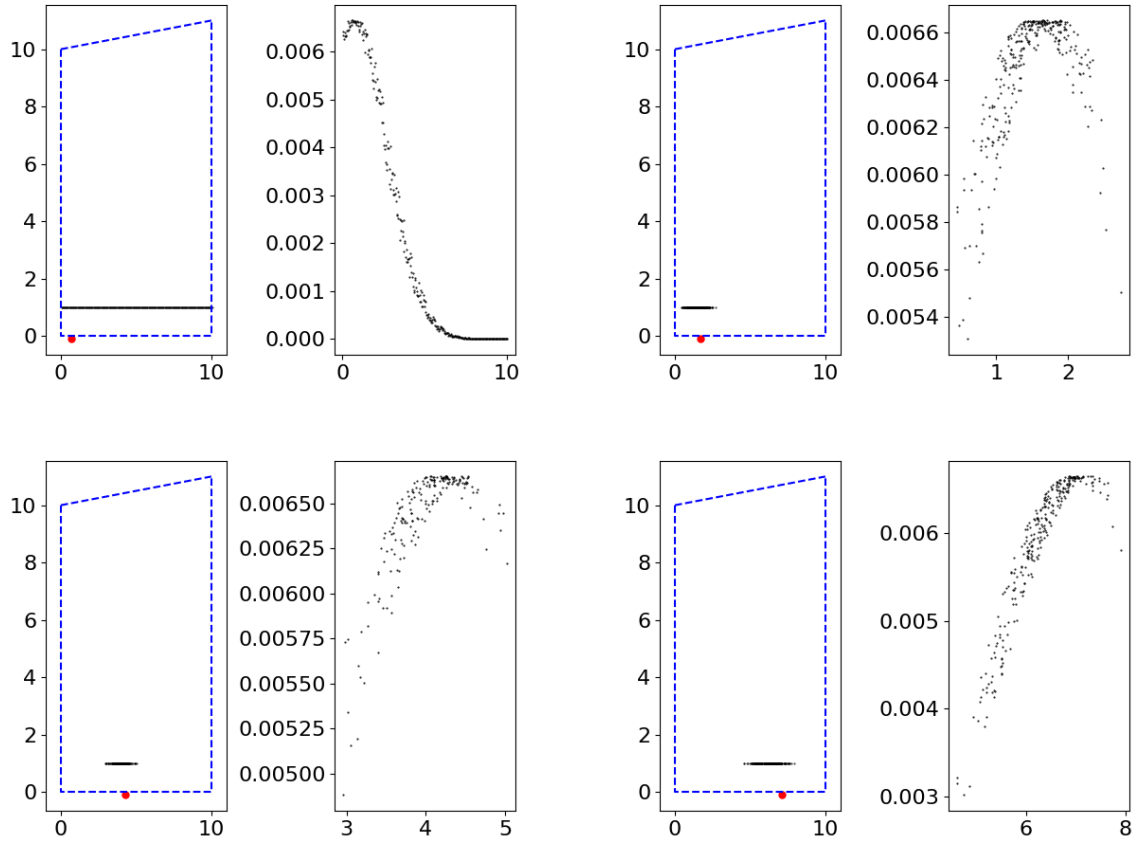


Figure 4.2: Particle filter result with $t = 0s$, $t = 5s$, $t = 18s$, $t = 32s$

Here, we can see the action of the particle filter which permits to follow the movement of the robot. The most valuable parameter to scale the accuracy of the filter is the spread parameter from the redistribution step.

4.1.3 With a symetric ceiling

Firstly, we have decided to build a linear increasing (or decreasing, it does not change anything) ceiling. Now we chose to build an other type of ceiling with a symetry on it. We will chose a curve for the shape with the function $ceil(x) = -0.1x^2 + x + 10$.

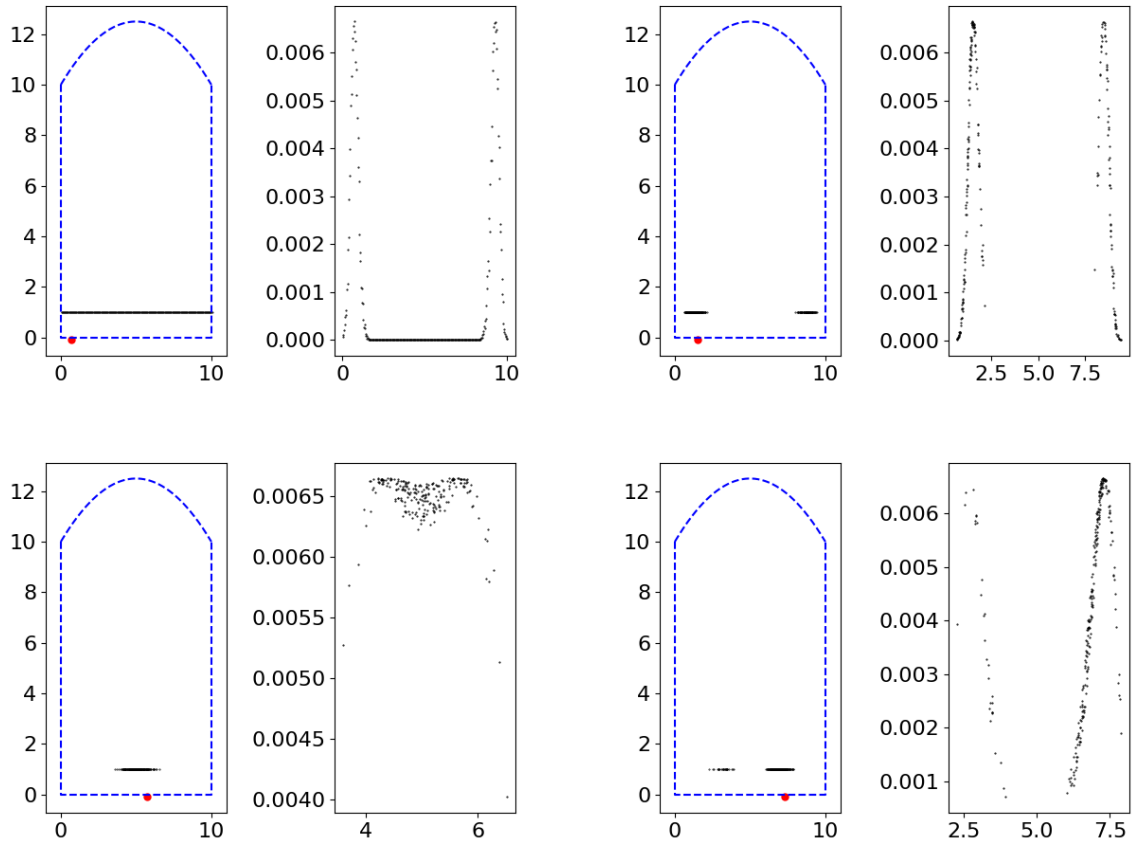


Figure 4.3: Particle filter result with $t = 0s$, $t = 4s$, $t = 25s$, $t = 33$

Now, we have some changes in the result, we can see that the particle filter does not manage to localise properly the robot because of the symetry. Furthermore, if the robot do not move fast enough, the spread parameter of the redistribution will erase overwhelm this movement. In this case, the robot has to wait until the middle of the ceiling to be able to be localised and to estimate the position.

This issue can be avoided if the spread is more tight after the redistribution, but there is a risk to lost the robot if he moves too quickly or if the calculation is too slow because of the number of particles.

4.1.4 With a periodic-shape ceiling as a sawtooth

Now, we will change the issue of the symetry already done and move to an issue base the periodicity of the shape. If the ceiling is periodic as a sawtooth function, $ceil(x) = 10 + 2 * (x - |x|)$.

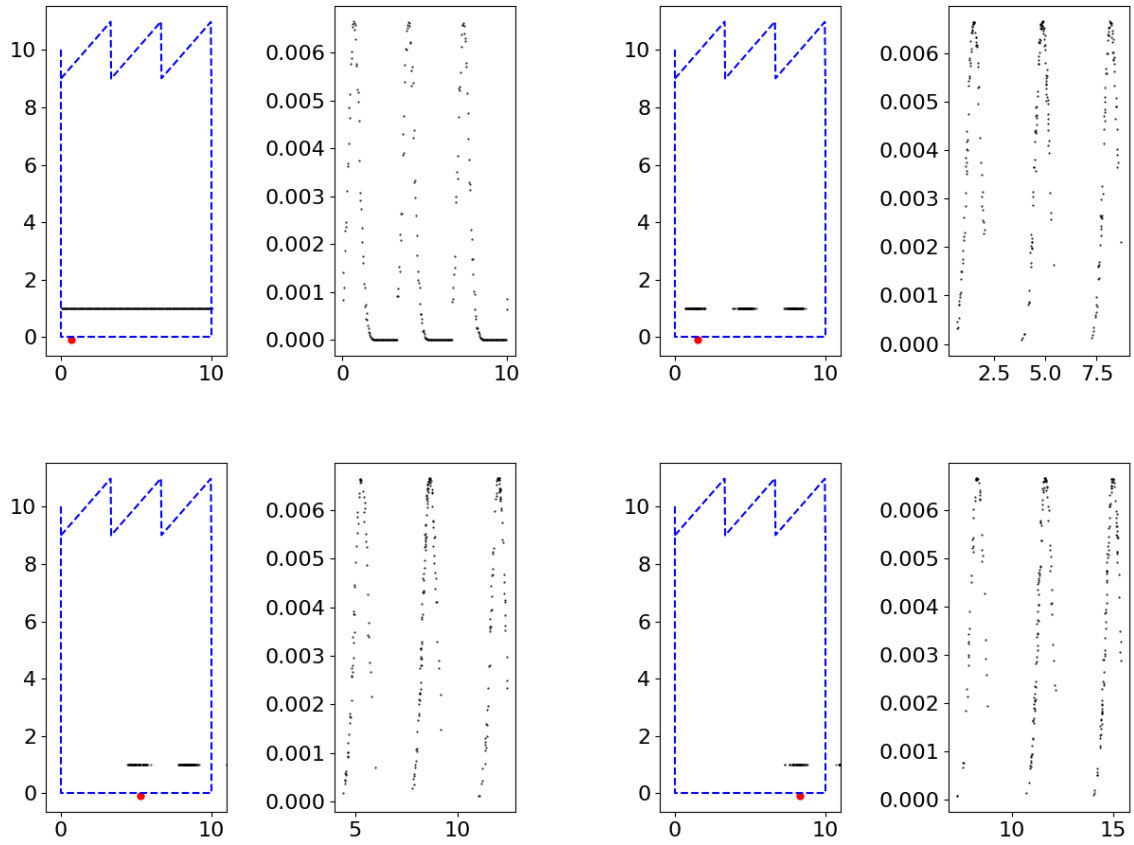


Figure 4.4: Particle filter result with $t = 0s$, $t = 4s$, $t = 23s$, $t = 38s$

Here, we can see the problem at the beginning and we do not know the position of the robot. In this case, we can admit that we do not know the location of the robot. Furthermore, in this simulation the ceiling is a function, and it not taken in account that the ceiling does not exist before and after the walls. That is why the curve representing the weight of the particle does not change in shape but only translate as the motion does.

4.1.5 With a flat-by-part ceiling

In this part, we want to find the an other cases with a special behaviour. Here, we define the ceiling with 3 diffenrent parts:

- 1 : decreasing (from left to right) $x \in [0, 2.5] ceil(x) = 12.5 - x$;
- 2 : constant $x \in [2.5, 7.5] ceil(x) = 10$;
- 1 : decreasing $x \in [7.5, 10] ceil(x) = 10 - (x - 7.5)$.

This choice for the part 1 and 2 is made to have clearly different parts to permit to the filter to find quickly the location at the beggining. z

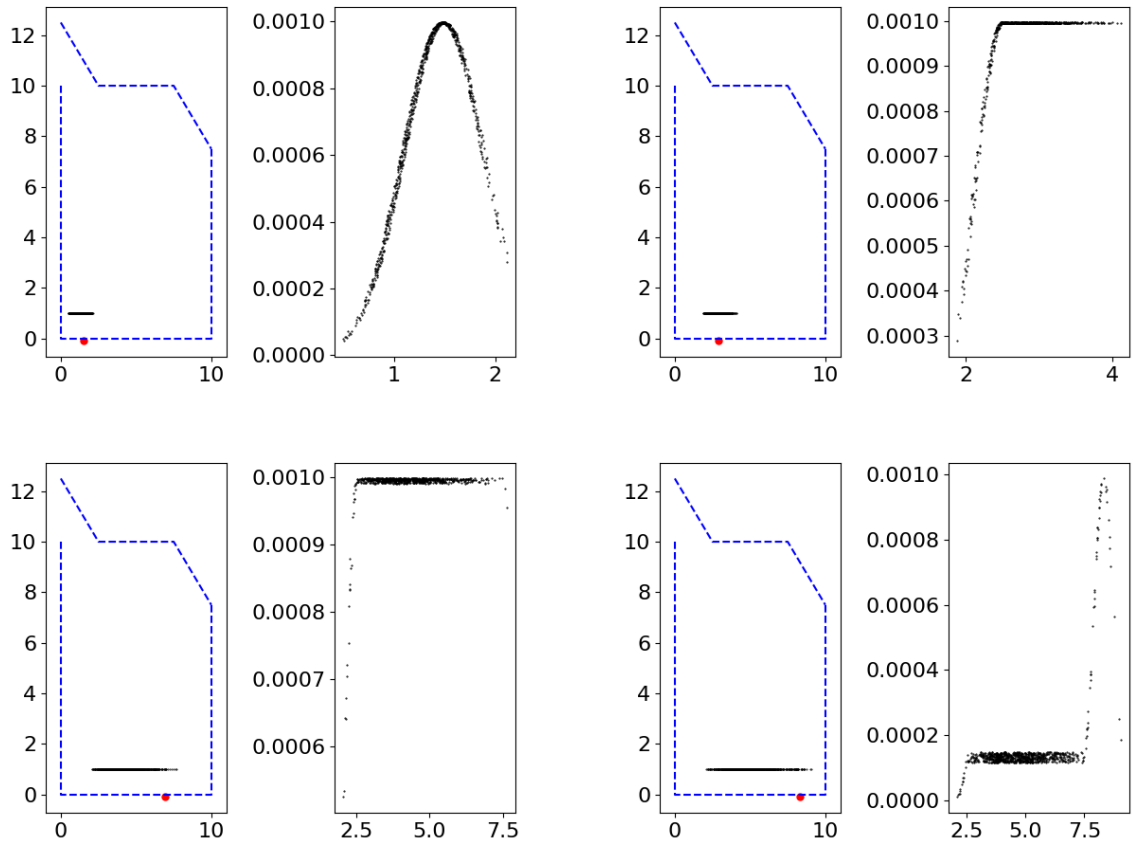


Figure 4.5: Particle filter result with $t = 4s$, $t = 11s$, $t = 31, t = 38s$

On these figures, the filter manages to estimate the location of the robot during the part 1. Then, on the part 2, he does not know where are the robot because there are not "data association" in this simulation. This subject will be introduced at the end of the report, this is not the main subject here. We can see the application on both the spreading permitting to the particle to be on the whole constant part. Finally, when the robot arrives at the part 3, the filter localise it.

The interesting information to keep from this example is the choice of the parameters of the filter. The choice made permits to scale the "spread" of the robot compared with the speed of this one. If the spread is slower than the speed, the filter will not manage to relocalise the robot after losing it. Furthermore, the noise added to the measure and the σ of the gaussian probality permit to scale the width of the curve reprensenting the position of the robot. It permits to chose if you want to be accurate in the localisation with a risk of losing the position or safety with a loss a accuracy.

Chapter 5

Application on a ground robot in C++

5.1 Context

The particle filter permits to localise a robot in a known map. THe first part of the project was to developp the theory. Now, we have a robot equipped with a Raspberry Pi for the control part and an Arduino module for the power supply and the application of order on motors. To be qble to communicate between the different parts of the robot (LIDAR, motors, electronic, ...), we are using ROS for Robotic Operating System. This software is designed for the communication between programs which runs in a parralel way.

Before continuing on the implementation and the way to make the code. We will explain some different ways to implement the particle filter in any kind of language. Furthermore, All the figures in this part represent a situation explaining the steps of the particle filter. The figures are just educational, the issue from the shape of the map, the symetry or the parallelism of the walls are not taken in account in the result.

5.2 Polygons or points

In this section, we will describe two different ways to implement a particle filter in a known map. The choices are:

- to use a multitude of points to describe the map.
- to use polygons to describe the map;

5.2.1 Points

Principle

In this case, we use points to descsribe the map, the map in represented as a binary matrix with a certain resolution, height and width. These will not be defined because this is not necessary at this point of the report. (Fig 4.1)

So now, on this map, we have one robot and N particles representing possible state of the robot. We will now apply the particle filter algorithm step by step supposing that the initialisation has been made correctly.

Spreading In this step, we apply to each particle the evolution model of the robot.

To avoid some errors, we add some noises to the control law for each particle because we do not know if the robot apply correctly the control. So to simulate some errors, we add some noises to every movement of the particles.

1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1

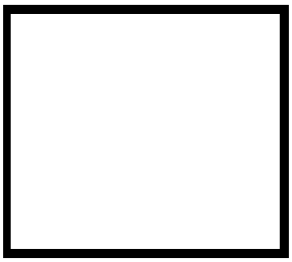


Figure 5.1: Equivalent matrix and real map

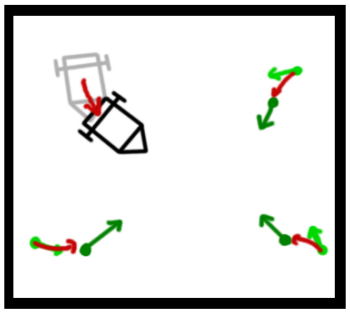


Figure 5.2: Spreading, application of the motion to the robot and to the particle

Update Weights Then, we take the measure from the robot and the measure from the particles. To compare it, we use a gaussian likelihood $N(0, 1)$ centered in 0 with a standart deviation of 1. We apply this likelihood to every the differences between the measures of all rays from particles and all rays from the robot. We obtain at the end a new weight reprensenting the reliability of the particle. To be able of doing such a calculation, we use the hypothesis that every measure are independant, so we can multiply the probability of every particles together.

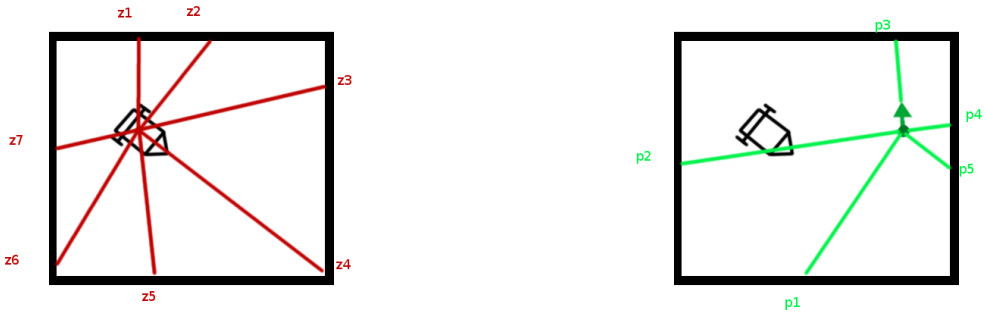


Figure 5.3: Update Weights, measures from the robot and from the particles

Estimation At this point, all the particles have a new weight reprensenting their reliability. Therefore, we can estimate the position of the robot.

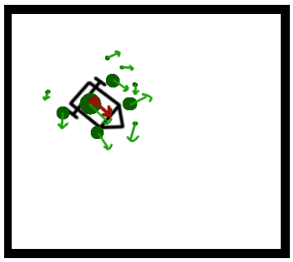


Figure 5.4: Estimation, many estimation methods work as the Minimum Mean Square Estimation

Redistribution At the end of the loop, the redistribution has to rebuild N particles with the same weight $1/N$. The thing is to divide all the particles into many using their actual weights until the number of new particles reachs N.

Figure 5.5: Redistribution

To correctly follow the random feature of the particle filter, if you divide a particle (mother) into a few (daughters). The daughters will placed randomly into an area near the position of the original particle. The parameter of the spreading of the distribution can be fixed or depending of the covariance of the estimation.

After a few steps, if there are not special cases, the filter would be able to localise the robot. The reliability of the estimation can be known with the covariance of this estimation.

Details

How to measure the distance between a particule and a point of the map In the step of updating the weights of the particles, we just said that we should compare the measure from the LIDAR and the distance from between the particle and a point of the map. But this is not a triavility. To be able to compare a ray and a distance, you have to be sure these two things are comparable. The thing is to follow the angle of the ray, beucause thanks to the LIDAR, we know the angle of each ray relatively of the head of the robot. So you have to follow the virtual ray starting from the particle with a angle known (absolutely and relatively to the head of the particle) until finding the first case of the map with an obstacle.
There are many choices possible to find it. Two possibilities are:

- successively move on this ray with a certain step until either find an obstacle (a 1 in the matrix) or leaving the defined map;
- defined the trajectory of the ray, find all the cases of the matrix which are on this line, then find the cases with the smallest distance with the robot.

Is it possible to lose the robot because no particles are relatively well placed ? The particle filter work with a certain number of particle, on a certain map with a certain resolution. All these parameters can scale the accuracy, performance and reliability of the filter. During the step of updating weights, the filter use measures depending of the position and the orientation of the particle relatively to the robot. With enough particle, the filter manages to localise the robot thanks to the law of large numbers. Furthermore, all the weights resulting of this step are all relativ, while the number are still usable on the program (depending on the type of the variable), the particles can be sorted using their weight and the algorithm can be applied. The filter will need few steps to be able to localise properly.

5.2.2 Polygons

Principle

In the first approach, the map was defined by a point cloud, every point representing a possible obstacle on reality and a 1 in the matrix. Now, the definition will change, instead of using points, we will use lines. Each obstacle will be defined as a filled polygon. Furthemore, each line will be defined in a parametrized way. We will define the ray as a parametrized line too, a ray will starting from the LIDAR to a certain angle relatively to the head of the robot.

- a *polygon* is defined as a set of *segments*
- a *segment* is defined as a *line* $S = p1 + (p2 - p1) * t$ where $p1, p2 \in \mathbb{R}^2, t \in [0, 1]$
- a *line* is defined as $L = p + v * t$ where $p, v \in \mathbb{R}^2, t \in [-\infty, \infty]$
- a *ray* is defined as $R = p + v * t$ where $p, v \in \mathbb{R}^2, t \in [0, \infty]$

With these definition, the intersection between 2 lines is computable. To know if the segments are crossing when the lines are crossing, we have to know the value of t. If $t \in [0, 1]$, the segments are crossing. The same logic works with an intersection between a ray and a segment.

With all the definition, we will now apply these hypothesis to our map.

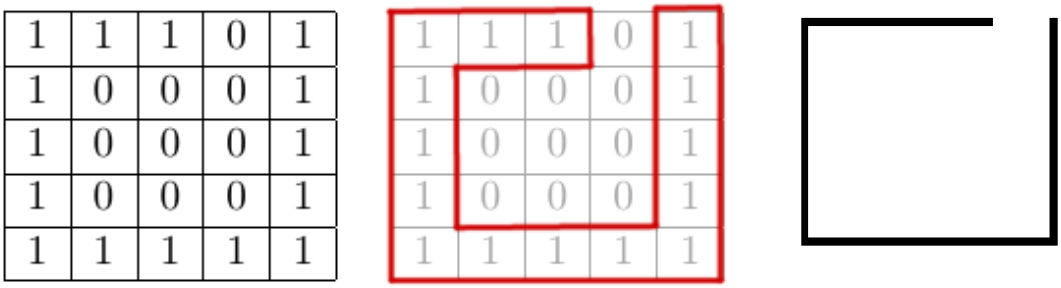


Figure 5.6: Principle and definition of the objects

In this part, we will define the method of particle filtering using polygons and lines. A huge part of the method is still the same as in the line method.

Here, this part is focused on defining the ways to update the weights of all the particles.

Update Weights To update the different weights for all the particle, we will find the intersection between a defined ray (known by its origin and its direction) and every segments defined. The obstacle will be defined as the segments which cross the ray with the smaller distance between the origin of the ray and the intersection point.

Details

How to measure the distance between the robot and the closest obstacle ? In this context, you cannot choose directly the closest obstacle. Here, we will measure the distance between the robot and all the lines created. The calculation to find one intersection between a ray and a line is fast, so we can calculate quickly the distance between a ray and all the lines created. After the calculation of all these distances, we choose the smallest and the distance is found.

What happend if the ray and the line are parrallel ? In this case, we use the library Eigen in C++, in this library, these calculation are already implemented. All the specific cases are taken in account.

What happens if the ray do not cross any lines ? You have two possiblities when you build a map; either the map is closed with an obstacle to do the perimeter so there will always be a intersection or the map is opened so if no intersection are found or really far, the solution is to choose the maximal range of the sensor.

How to create the polygons using only a matrix ? First, we have to define the hypothesis to create the polygons. When we use a binary matrix, we do not know the probability about the position of a obstacle in a square. We will assume that all the square can be an obstacle and also define that the polygon will follow the border of the block of 1. To follow this border, we create a cursor permitting us to move on the lines.
To initialize, we will find the first 1 of the matrix starting on top left corner. we will set up the cursor toward the right. Then we link the cursor to the case in front of it on the right.

Chapter 6

Conclusion

In this report, all the basic knowledge to implement a particle filter is explained. The algorithm is pretty simple and the bigger difficulty comes from the way to implement this algorithm in a program and to settle the values of the different parameters depending on the context.

Now, this report aimed to introduce this kinf of filter and not to show a perfect application. I tried to implement the filter in a ground robot using the C++ language. All the implementation is commented but it doesn't work for the moment, the ROS implemenation to create a communication between the LIDAR and the Raspberry Pi is made and works, but some parts of the full program are missing like linking the polygons created to the loop with the measures and the application of the gassian likelihood.

To go further in this project, some features are possible to developp. After debbuging and finishing the particle filter itself, the goal is to implement firstly a data association to solve issues like symetry or periodicity. Then, implement a SLAM method in a 2D plan will permit to use a robot in a unknown map and building it while discovering.

Bibliography

[1] ejkreinar. Basics of particle filter, October 2012. https://cwrucutter.wordpress.com/2012/10/04/basics-of-particle-filters/.

[2] François Legland. Filtrage particulaire.

[3] Nicolas Paul. Filtrage particulaire, March 2006. Séminaire de Statistiques, Conservatoire Nationale des Arts et Métiers.

[4] Cyril Stachniss. Short introduction to particle filters and monte carlo localization, December 2012.

[5] Andreas Svensson. Particle filter explained without equations. posted on youtube, October 2013. https://www.youtube.com/watch?v=aUkBa1zMKv4.

[6] Sebastian Thrun. Particle filter in robotics, 2000.

Appendix A

Markov Chains

This is the course I used to understand the Markov chain to link with the Monte-Carlo method in addition with the course in RobMooc by L. Jaulin.

Chaînes de Markov (et applications)

Raphael Lachieze-Rey*

25 janvier 2016

M1 Paris Descartes.

Table des matières

*lr.raphael@gmail.com

1 Chaînes de Markov homogènes

$(\Omega, \mathbb{P})$ est un espace probabilisé.

1.1 Exemples et définitions

Idée : Une chaîne de Markov est une suite d'événements aléatoires dans le temps ou **conditionnellement au présent, le futur ne dépend pas du passé**, ou autrement dit le futur ne dépend du passé que par le présent.

Exercice 1.
Parmi les exemples suivants, lesquels correspondent a une chaîne de Markov ?

- Les records du monde du 100m
- La population mondiale
- La position d'une voiture (car le passé nous renseigne sur sa vitesse, et donc sur sa position future)
- Le nombre de personnes dans une file d'attente
- Un marcheur aléatoire qui ne revient jamais sur ses pas.
- Le couple (position, vitesse) d'une voiture de course
- une marche aléatoire

Définition 1. *Formellement, soit E un espace fini ou dénombrable. Ce sera l'***espace d'états**. *Soit $X = \{X_n; n \geq 0\}$ une suite de variables aléatoires à valeurs dans E. On dit que X est une chaîne de Markov si, pour tout $x_1, \ldots, x_{n+1} \in E$, on a*

$$\mathbb{P}(\underbrace{X_{n+1} = x_{n+1}}_{\text{Le futur}} \mid \underbrace{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n}_{\text{Le passé (et le présent)}}) = \mathbb{P}(\underbrace{X_{n+1} = x_{n+1}}_{\text{Le futur}} \mid \underbrace{X_n = x_n}_{\text{Le présent}})$$

Cette propriété des chaînes de Markov est aussi connue comme **propriété de Markov**.

Exercice 2. Soit $R_n, n \geq 0$ des variables indépendantes à valeurs dans $E = \mathbb{N}$. Montrer que $S_n = \sum_{i=1}^{n} R_i$ et $P_n = \prod_{i=1}^{n} R_i$ sont des chaînes de Markov.

Une chaîne de Markov peut être vue comme un **système dynamique**, ce qui veut dire que $X_{n+1} = f_n(X_n)$, ou f_n est une "transformation aléatoire" indépendante du passé. Dans l'exemple précédent, $f_n(X_n)$ est la somme (ou le produit) de X_n avec R_{n+1}.

Si la transformation aléatoire f_n ne dépend pas de n, i.e. $X_{n+1} = f(X_n)$ pour tout n pour une certaine transformation f, on dit que X est une **chaîne de Markov homogène**.

Cela veut dire que si à un certain instant $n \geq 0$ la chaîne se trouve à l'état x ($X_n = x$), alors la probabilité qu'elle se trouve à l'état y au temps $n + 1$ est la même que si l'on était au temps initial.

Définition 2. *Une chaîne de Markov est homogène si pour tout $n \geq 0$, x et y dans E*

$$\mathbb{P}(X_{n+1} = y \,|\, X_n = x) = \mathbb{P}(X_1 = y \,|\, X_0 = x).$$

Dans ce cas, on pose

$$Q(x, y) = \mathbb{P}(X_1 = y \,|\, X_0 = x),\ x, y \in E.$$

Q est la **matrice de transition** *de la chaîne X.*

Dans ce cours, toutes les chaîne de Markov sont supposées homogènes.

Remarque 1. *C'est éventuellement une matrice infinie*

Une chaîne de Markov homogène "saute" donc aléatoirement d'états en états, et la probabilité de chaque saut est donnée par la matrice Q.

Exemple 1. Une grenouille monte sur une échelle. Chaque minute, elle peut monter d'un barreau avec probabilité $1/2$, ou descendre d'un barreau avec probabilité $1/2$. L'échelle a 5 barreaux. Si la grenouille arrive tout en haut, elle saute immédiatement en bas de l'échelle et recommence.

On appelle X_n la position de la grenouille sur l'échelle. L'espace d'états est donc $E = \{0, 1, 2, \ldots, 5\}$. Si a un instant n la grenouille est au niveau $x \in \{1, 2, 3, 4\}$ de l'échelle, alors à l'instant $n + 1$ elle sera

$$\begin{cases} \text{au barreau } x + 1 \text{ avec probabilité } 1/2, \\ \text{au barreau } x - 1 \text{ avec probabilité } 1/2, \end{cases}$$

ce qui se traduit par

$$\begin{aligned} \mathbb{P}(X_{n+1} = x + 1 \,|\, X_n = x) = 1/2 \quad (= \mathbb{P}(X_1 = x + 1 \,|\, X_0 = x)), \\ \mathbb{P}(X_{n+1} = x - 1 \,|\, X_n = x) = 1/2 \quad (= \mathbb{P}(X_1 = x - 1 \,|\, X_0 = x) \end{aligned}$$

Comme les probabilités ne dépendent pas de n, il semble que l'on tienne le bon bout pour avoir une chaîne de Markov homogène. Si c'est le cas, on peut écrire une partie de la matrice de transition :

$$Q = \begin{pmatrix} ? & ? & ? & ? & ? & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ ? & ? & ? & ? & ? & ? \end{pmatrix}$$

Si la grenouille se retrouve à l'état 5, alors elle peut soit passer à l'état 4, soit passer à l'état 0. Il faut donc remplacer la dernière ligne de la matrice par

$$(1/2, 0, 0, 1/2, 0),$$

(encore une fois cela ne dépend pas de l'instant n). Si la grenouille est à l'état 0, elle ne peut que passer à l'état 1. La première ligne de la matrice est donc

$$(0, 1, 0, 0, 0).$$

X_n est donc bien une chaîne de Markov homogène, avec matrice de transition Q.

Exercice 3. Introduisons un facteur de fatigue $f \in (0, 1)$, et imaginons qu'à chaque instant la grenouille reste à son état actuel avec probabilité f. X_n est toujours une chaîne de Markov ? Si oui, quelle est sa matrice de transition ?

Imaginons désormais que le facteur de fatigue $f = f_n$ dépend du temps. Que cela change-t-il ?

Si désormais le facteur de fatigue dépend de tout le chemin parcouru par la grenouille (nombre de barreaux montés et descendus), a-t-on toujours une chaîne de Markov ?

Exercice 4. Le nombre d'individus d'une population évolue de la manière suivante : A chaque instant, un individu nait avec la probabilité $p \in (0, 1)$, ou meurt avec la probabilité $q = 1 - p$.

Ecrire la matrice de transition.

Ecrire la chaîne de Markov en termes des variables introduites à l'exemple 2.

Comment corriger la matrice de transition pour qu'il n'y ait pas un nombre négatif d'individus ?

correction: On a en fait une marche aléatoire avec des variables de Rademacher R_n

$$\mathbb{P}(R_n = \pm 1) = 1/2.$$

Le problème est qu'on peut avoir un nombre négatif d'individus. Il faut donc corriger la MDT en remplaçant la 1re ligne par $(1, 0, 0, \dots)$ (et les nombres négatifs ne font plus partie de l'espace d'état.) □

Définition 3. *On dit qu'une matrice Q (éventuellement infinie) est* **stochastique** *ssi tous ses coefficients sont ≥ 0 et si la somme de chaque ligne fait 1 : $\forall x \in E$,*

$$\sum_{y \in E} Q(x, y) = 1.$$

On dit aussi **matrice markovienne**.

Remarque 2. *Les coefficients d'une matrice stochastique sont dans $[0, 1]$, ils peuvent donc représenter une probabilité...*

Proposition 1. *Si Q est la matrice de transition d'une chaîne de Markov, alors elle est stochastique.*

Démonstration. Soit $x \in E$. Alors

$$\begin{aligned}\sum_{y\in E} Q(x,y) &= \sum_{y} \mathbb{P}(X_1 = y \,|\, X_0 = x)\\ &= \mathbb{P}(\text{“}X_n \textbf{ aille quelque part après être allé en } x\text{...”})\\ &= 1.\end{aligned}$$

Plus formellement,

$$\begin{aligned}\sum_{y} \mathbb{P}(X_1 = y \,|\, X_0 = x) &= \sum_{y} \mathbb{E}(1_{X_1=y}|X_0 = x)\\ &= \mathbb{E}(\sum_{y\in E} 1_{X_1=y}|X_0 = x)\end{aligned}$$

Or

$$\sum_{y\in E} 1_{X_1=y} = 1$$

tout le temps car X_1 est égal à un et un seul des y. Donc

$$\sum_{y} \mathbb{P}(X_1 = y \,|\, X_0 = x) = \mathbb{E}(1|X_0 = x) = 1.$$

□

1.2 Loi des X_n

Le comportement d'une chaîne de Markov X dépend entièrement de sa matrice de transition Q, et de la position initiale X_0. On appelle μ_0 la **loi initiale** de X, définie par

$$\mu_0(x) = \mathbb{P}(X_0 = x).$$

Connaissant μ_0 et Q, on peut calculer directement la loi de X_n.

Proposition 2. *Pour toute suite $\{x_0, x_1, \dots, x_n\}$ dans E, on a*

$$\begin{aligned}&\mathbb{P}(X_0 = x_0, X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)\\ &\qquad = \mu_0(x_0)Q(x_0,x_1)Q(x_1,x_2)\dots Q(x_{n-1},x_n).\end{aligned}$$

Démonstration. On a (en utilisant la propriété de Markov)

$$\begin{aligned}&\mathbb{P}(X_0 = x_0, X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)\\ &\qquad = \mathbb{P}(X_n = x_n, \dots, X_1 = x_1 \,|\, X_0 = 0)\mathbb{P}(X_0 = x_0)\\ &\qquad = \mathbb{P}(X_n = x_n, \dots, X_1 = x_1 \,|\, X_0 = 0)\mu_0(x_0)\\ &\qquad = \mathbb{P}(X_n = x_n, \dots, X_2 = x_2 \,|\, X_1 = x_1, X_0 = 0)\mathbb{P}(X_1 = x_1|X_0 = x_0)\mu_0(x_0)\\ &\qquad = \mathbb{P}(X_n = x_n, \dots, X_2 = x_2 \,|\, X_1 = x_1, X_0 = 0)Q(x_0,x_1)\mu_0(x_0)\\ &\qquad = \mathbb{P}(X_n = x_n, \dots, X_2 = x_2 \,|\, X_1 = x_1)Q(x_0,x_1)\mu_0(x_0)\end{aligned}$$

en utilisant la **propriété de Markov**. En utilisant le même raisonnement, on montre que

$$\mathbb{P}(X_n = x_n, \dots, X_2 = x_2 \,|\, X_1 = x_1) = \mathbb{P}(X_n = x_n, \dots, X_3 = x_3 \,|\, X_2 = x_2)Q(x_1, x_2).$$
$$\mathbb{P}(X_0 = x_0, X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = \mathbb{P}(X_n = x_n, \dots, X_3 = x_3 \,|\, X_2 = x_2)Q(x_1, x_2)Q(x_0, x_1)\mu_0(x_0).$$

Par récurrence, on montre la formule proposée.

□

Notation 1. Pour une mesure μ_0 et une matrice Q, on note la mesure

$$(\mu_0 Q)(y) = \sum_{x \in E} \mu_0(x) Q(x, y).$$

Cela revient à multiplier (matriciellement) la mesure μ_0 vue comme un vecteur $\mu_0 = (\mu_0(x_1), \mu_0(x_2), \dots)$ par la matrice Q.

Proposition 3. *Si μ es la loi de X_0, alors (μQ) est la loi de X_1.*

Démonstration. Soit $y \in E$.

$$\begin{aligned}\mathbb{P}(X_1 = y) &= \sum_x \mathbb{P}(X_1 = y, X_0 = x) \\ &= \sum_x \mathbb{P}(X_1 = y | X_0 = x)\mathbb{P}(X_0 = x) = \sum_x \mu(x)Q(x, y) = (\mu Q)(y)\end{aligned}$$

□

Pour une même chaîne X, on considère souvent plusieurs lois initiales différentes. Dans ce cas on précise la loi utilisée en notant

$$\mathbb{P} = \mathbb{P}_\mu$$

dans chaque calcul de probabilité, et l'espérance est alors notée $\mathbb{E}_x$. Si la loi est un "dirac" $\mu = \delta_x$ pour un certain $x \in E$ (ce qui veut dire $X_0 = x$ p.s.), alors on note plus simplement $\mathbb{P}_{\delta_x} = \mathbb{P}_x, \mathbb{E}_{\delta_x} = \mathbb{E}_x$.

Exemple 2. Pour reprendre l'exemple de la grenouille,

$$\begin{aligned}&\mathbb{P}_0(X_1 = 1) = 1, \\ &\mathbb{P}_0(X_1 = 3) = 0, \\ &\mathbb{P}_2(X_1 = 3) = 1/2, \\ &\mathbb{P}_0(X_3 = 3) = (1/2)^3 = 1/8, \\ &\mathbb{P}_0(X_3 = 4) = 0, \\ &\dots\end{aligned}$$

On peut calculer directement la loi de X_n en utilisant le produit matriciel.

Exercice 5. Soit $a \in (0,1)$. On considère une chaîne de Markov dont la matrice de transition est la suivante :

$$\begin{pmatrix} a & 1-a \\ 1-a & a \end{pmatrix}$$

Calculer la probabilité de l'événement A_3 de passer pour $n \leq 5$ trois fois par l'état 2

1. Avec $\mu_0 = (1,0)$,
2. Avec $\mu_0 = (1/2, 1/2)$.

Correction :

Il faut dénombrer tous les chemins possibles qui contiennent trois fois l'état 2. En partant de 1 :

- $x^1 = (1,1,2,2,2)$
- $x^2 = (1,2,1,2,2)$
- $x^3 = (1,2,2,1,2)$
- $x^4 = (1,2,2,2,1)$

Pour le premier chemin, sa probabilité est, d'après la formule précédente,

$$\mathbb{P}_1(X = x^1) = Q(1,1)Q(1,2)Q(2,2)Q(2,2) = a(1-a)a^2 = a^3(1-a).$$

Avec d'autres calculs, on montre

$$\mathbb{P}_1(A_3) = a^3(1-a) + 2a(1-a)^3 + a^2(1-a)^2.$$

Pour la seconde question, il faut calculer $\mathbb{P}_2(A_3)$ de manière similaire, et on a

$$\mathbb{P}_{\mu_0}(A_3) = \mu_0(\{1\})\mathbb{P}_1(A_3) + \mu_0(\{2\})\mathbb{P}_2(A_3) = 1/2(...)$$

Proposition 4. *Pour tout n, la loi de X_n est μQ^n.*

Démonstration. D'après la proposition 3, la loi de X_1 est $\mu_1 = \mu Q$.

On peut alors "oublier" la variable X_0, et ne considérer que la chaîne qui part de X_1 (formellement, poser $X_0' = X_1, X_1' = X_2, etc...$). La matrice de transition est toujours Q, par contre la loi initiale n'est plus μ, c'est μ_1.

La loi de X_2 (i.e. X_1') est donc, en réutilisant la proposition 3,

$$\mu_2 = (\mu_1 Q) = ((\mu Q) * Q).$$

Comme le produit matriciel est associatif, $((\mu Q) * Q) = \mu * Q * Q = \mu * (Q * Q) = \mu * Q^2$.

La loi de X_2 est donc bien μQ^2, comme annoncé. En raisonnant par récurrence, on montre bien $\mu_3 = (\mu * Q^2) * Q = \mu Q^3$, ..., ... et $\mu_n = \mu Q^n$. □

Exercice 6. Une chaîne de Markov avec états $E = \{1, 2\}$ a la matrice de transition

$$Q = \begin{pmatrix} a & 1-a \\ 1-a & a \end{pmatrix}$$

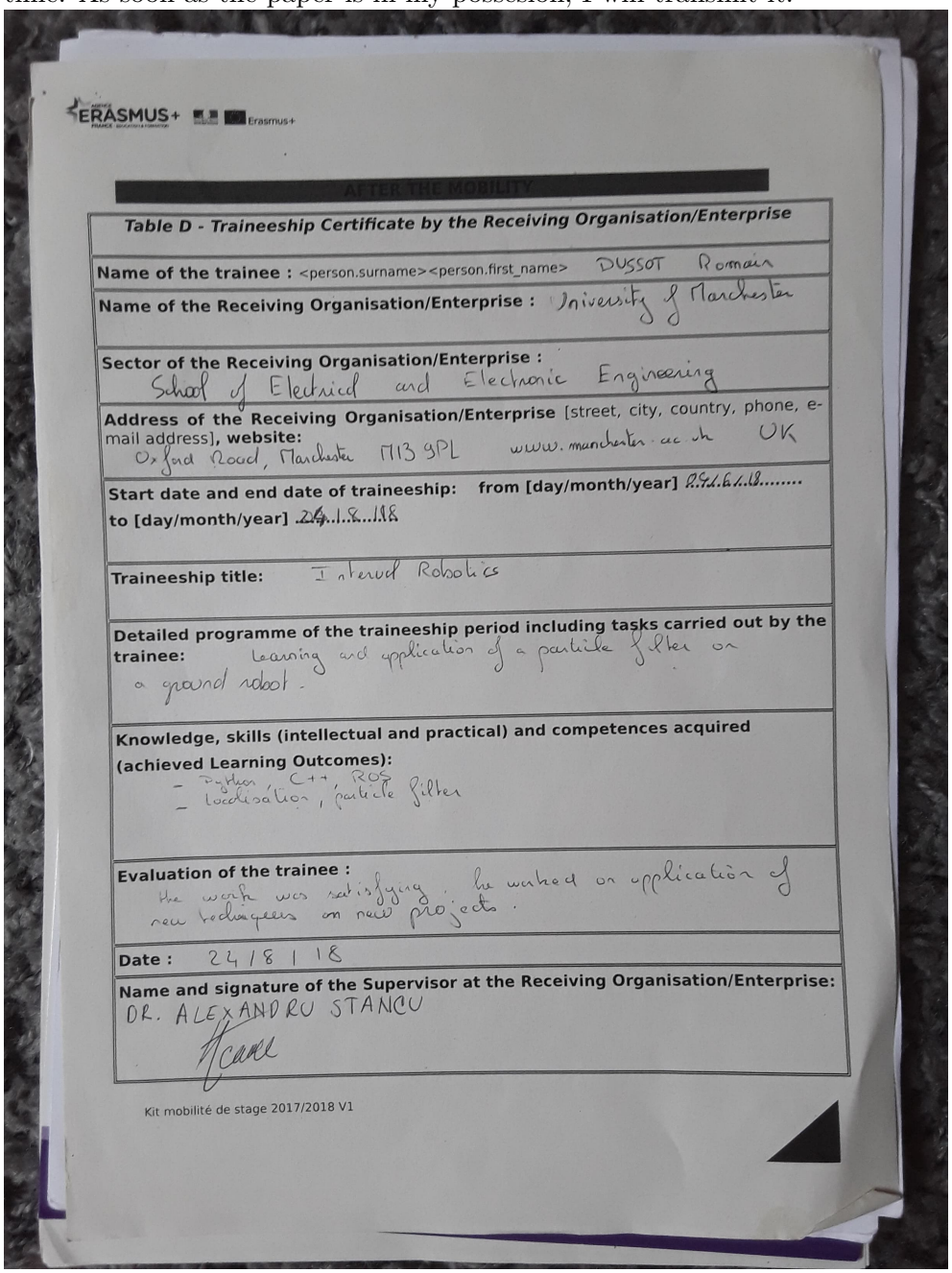
pour un nombre $a \in (0, 1)$.

Calculer la loi de X_n pour tout n, sachant que l'on part de l'état $X_0 = 1$. Donner par exemple $\mathbb{P}_1(X_n = 1)$ la probabilité de retour en 1 en n coups.

Appendix B

Evaluation by the supervisor

In this apendixe, the certificate of my after the mobility sent to ENSICAEN as a proof of the end of my internship instead of the assessment report from the ENSTA Bretagne because I did not get it on time. As soon as the paper is in my possesion, I will transmit it.



ERASMUS+ Erasmus+

AFTER THE MOBILITY

Table D - Traineeship Certificate by the Receiving Organisation/Enterprise

Name of the trainee : <person.surname><person.first_name> DUSSOT Romain

Name of the Receiving Organisation/Enterprise : University of Manchester

Sector of the Receiving Organisation/Enterprise :
School of Electrical and Electronic Engineering

Address of the Receiving Organisation/Enterprise [street, city, country, phone, e-mail address], **website:**
Oxford Road, Manchester M13 9PL www.manchester.ac.uk UK

Start date and end date of traineeship: from [day/month/year] 25/6/18
to [day/month/year] 24/8/18

Traineeship title: Interval Robotics

Detailed programme of the traineeship period including tasks carried out by the trainee: learning and application of a particle filter on a ground robot.

Knowledge, skills (intellectual and practical) and competences acquired (achieved Learning Outcomes):
- Python, C++, ROS
- localisation, particle filter

Evaluation of the trainee :
The work was satisfying, he worked on application of new technologies on new projects.

Date : 24/8/18

Name and signature of the Supervisor at the Receiving Organisation/Enterprise:
DR. ALEXANDRU STANCU

Kit mobilité de stage 2017/2018 V1