ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme und Regelungstechnik

151-0566-00 **Recursive Estimation** (Spring 2017)

Programming Exercise #2 Topic: Particle Filtering

Issued: May 10, 2017 Due: May 31, 2017

Michael Muehlebach (michaemu@ethz.ch), May 8, 2017

Particle Filter for Tracking two Vacuum Cleaning Robots

Your task is to design a Particle Filter (PF) that tracks two cleaning robots A, B that are moving at speeds $u_\mathrm{A}, u_\mathrm{B} > 0$ and headings $\theta_\mathrm{A}, \theta_\mathrm{B} \in [-\pi, \pi]$ in a rectangular room with side length L and $2L$. When a robot reaches a wall, it bounces off at a random angle that depends on the incident angle. At various times, the four sensors S_1, S_2, S_3, S_4, which are installed in the corners of the room, provide distance measurements to the robots. A sketch of the system is provided in Figure 1.

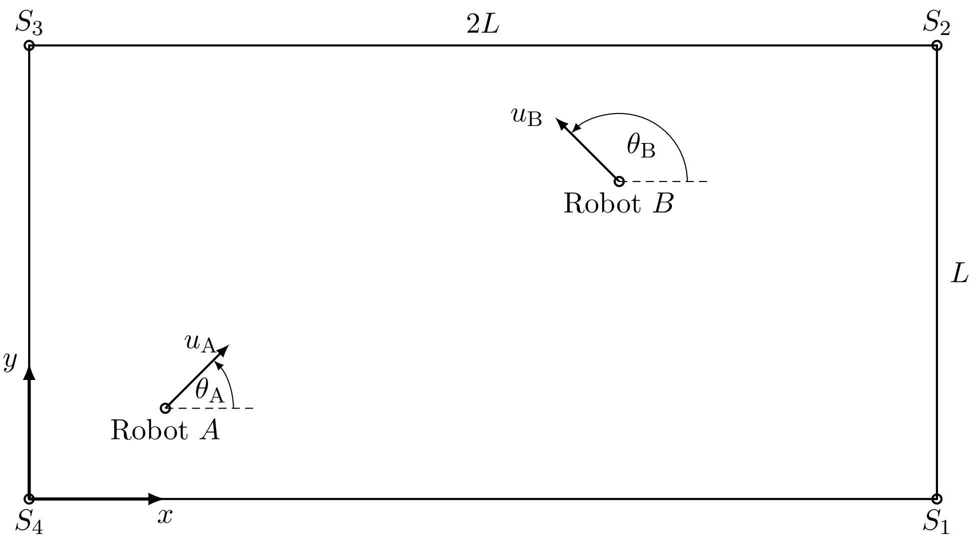


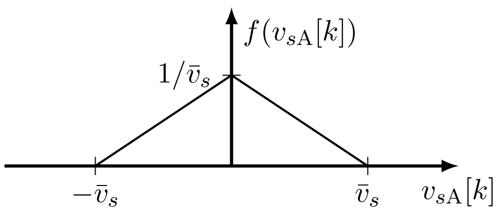
Figure 1: Sketch of the robots moving in the rectangular room that is equipped with four distance sensors, one in each corner.

System Dynamics

The positions of the robots at time t are given by the coordinates $x_\mathrm{A}(t), y_\mathrm{A}(t)$ and $x_\mathrm{B}(t), y_\mathrm{B}(t)$. The robot headings are $\theta_\mathrm{A}(t), \theta_\mathrm{B}(t)$, and are constant except during wall bounces. The robot velocities change at discrete time steps k. The discrete time k is related to the continuous time t by $t = kT_\mathrm{s}$, where T_s is a given, constant sampling time. The robot velocities are piece-wise constant during a sample time interval:

$$\begin{aligned} u_\mathrm{A}(t) &= u_\mathrm{A}[k-1](1 + v_{s\mathrm{A}}[k-1]), \quad \text{for } (k-1)T_\mathrm{s} \le t < kT_\mathrm{s} \\ u_\mathrm{B}(t) &= u_\mathrm{B}[k-1](1 + v_{s\mathrm{B}}[k-1]), \quad \text{for } (k-1)T_\mathrm{s} \le t < kT_\mathrm{s}, \end{aligned}$$

where $u_\mathrm{A}[k]$ and $u_\mathrm{B}[k]$ refers to the known (commanded) robot velocities, which are corrupted by the process noise $v_{s\mathrm{A}}[k]$, respectively $v_{s\mathrm{B}}[k]$. The process noise $v_{s\mathrm{A}}[k]$ has a triangular probability density function (PDF) with constant parameter $\bar{v}_s$:



The PDF of the process noise $v_{s\mathrm{B}}[k]$ is identical (triangular with constant parameter $\bar{v}_s$).
The continuous-time dynamics of the overall system in between wall bounces are then given by

$$\begin{aligned}
\dot{x}_\mathrm{A}(t) &= u_\mathrm{A}(t)\cos(\theta_\mathrm{A}(t)) & \dot{x}_\mathrm{B}(t) &= u_\mathrm{B}(t)\cos(\theta_\mathrm{B}(t)) \\
\dot{y}_\mathrm{A}(t) &= u_\mathrm{A}(t)\sin(\theta_\mathrm{A}(t)) & \dot{y}_\mathrm{B}(t) &= u_\mathrm{B}(t)\sin(\theta_\mathrm{B}(t)) \\
\dot{\theta}_\mathrm{A}(t) &= 0 & \dot{\theta}_\mathrm{B}(t) &= 0.
\end{aligned}$$

A robot bounces into a wall when its coordinates coincide with the walls shown in Figure 1, and its velocity vector is pointing into the wall. For example, a robot with coordinates $x(t), y(t)$; heading $\theta(t)$; and speed $u(t)$ bounces into the lower, horizontal wall if

$$y(t) = 0 \quad \text{and} \quad u(t)\sin(\theta(t)) < 0.$$

We define the pre-bounce angle α_j^- of the j-th bounce of the two robots as the nonnegative, acute angle between the robot velocity and the wall: $0 \leq \alpha^- \leq \pi/2$. This angle is illustrated in Figure 2 in an example of a robot bouncing off the lower wall.

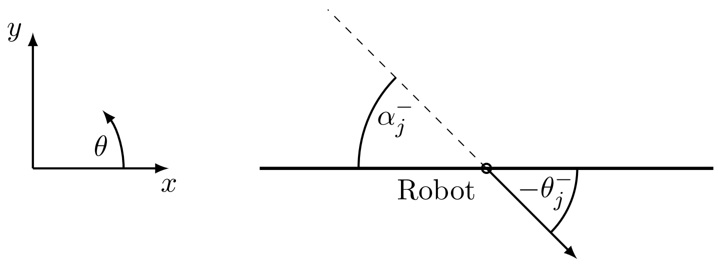


Figure 2: Illustration of the pre-bounce angle α_j^- in an example where a robot bounces off the lower wall with pre-bounce heading θ_j^-.

The speed of a robot is constant over a bounce. The nominal post-bounce angle $\bar{\alpha}_j^+$ is equal to the ideal reflection angle. The actual post-bounce angle α_j^+ is random:

$$\alpha_j^+ = \bar{\alpha}_j^+(1 + v_j)$$

where v_j is a continuous random variable whose sample space is the interval $[-\bar{v}, \bar{v}]$, where $0 \leq \bar{v} < 1$ is a constant. The probability density function (PDF) is

$$f(v_j) = \begin{cases} c\, v_j^2 & v_j \in [-\bar{v}, \bar{v}] \\ 0 & \text{otherwise} \end{cases}$$

for all bounces $j = 0, 1, 2, \ldots$ of the robots. The parameter c is a normalization constant. The post-bounce angle is illustrated in Figure 3 for the example of the robot bouncing off the lower wall.
Bounces on the other walls are analogous to the example shown in Figures 2 and 3. In the example shown in Figure 3, the post-bounce heading of the robot θ_j^+ is equal to α_j^+. However, in general, the actual post-bounce heading depends on the orientation of the wall and the pre-bounce heading θ_j^-. The same holds for the relation between the pre-bounce angle and the pre-bounce heading of the robot: In the example shown in Figure 2, $\alpha_j^- = -\theta_j^-$, however, in general, this relation depends on the orientation of the wall and the pre-bounce heading θ_j^-.

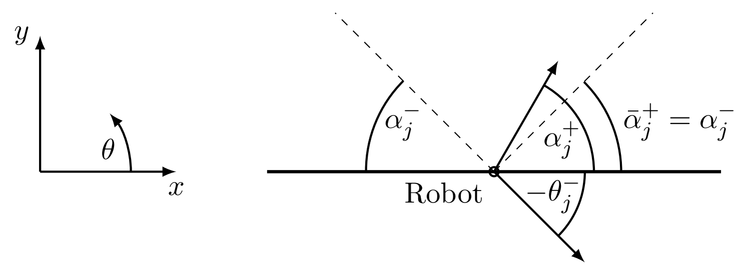
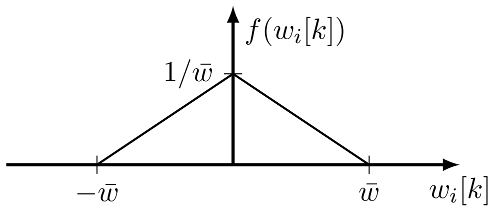


Figure 3: Illustration of the post-bounce angle α_j^+ for the example where a robot bounces off the lower wall.

For simplicity, we assume that the two robots do not collide (i.e. they are able to magically drive through each other).
At time $t = 0$, the robots randomly start in one of the corners of the room. Specifically, robot A starts out in one of the corners where the sensors S_1 and S_2 are located; and robot B starts out in one of the corners where sensors S_3 and S_4 are located. For each robot, both starting corners are equally likely. A robot's initial heading is random and uniformly distributed on the set of headings that keep the robot from driving out of the room. For example, the initial heading of robot A starting out in the corner of sensor S_1 is uniformly distributed on $\theta_A \in [\pi/2, \pi]$.

Measurement Model

Each sensor is calibrated for a specific robot: Sensors S_1 and S_2 are measuring distances to robot A, while sensors S_3 and S_4 are measuring distances to robot B. At each discrete time instant k, each sensor may or may not provide a measurement. The distances that the four sensors report are corrupted by additive measurement noise variables $w_i[k],\ i \in \{1,2,3,4\}$. The measurement noise variables have a triangular PDF with constant parameter $\bar{w}$, which is identical for all sensors:



With some constant probability $\bar{s} \in [0,1]$, each sensor detects the wrong robot and consequently measures the distance to the robot it was not calibrated for. This is captured by the random variables $s_i[k],\ i \in \{1,2,3,4\}$, which have the PDF

$$f(s_i[k] = 1) = 1 - \bar{s}, \quad \text{and} \quad f(s_i[k] = 0) = \bar{s}, \quad \text{for all } i \in \{1,2,3,4\}.$$

If $s_i[k] = 1$, the sensor measures the correct robot; if $s_i[k] = 0$, the sensor measures the wrong robot. The model for the measurements $z_i[k]$ corresponding to the sensors $S_i,\ i \in \{1,2,3,4\}$ then is

$$\begin{aligned}
z_1[k] &= s_1[k]d_{1\text{A}}[k] + (1 - s_1[k])d_{1\text{B}}[k] + w_1[k]\\
z_2[k] &= s_2[k]d_{2\text{A}}[k] + (1 - s_2[k])d_{2\text{B}}[k] + w_2[k]\\
z_3[k] &= s_3[k]d_{3\text{B}}[k] + (1 - s_3[k])d_{3\text{A}}[k] + w_3[k]\\
z_4[k] &= s_4[k]d_{4\text{B}}[k] + (1 - s_4[k])d_{4\text{A}}[k] + w_4[k]
\end{aligned}$$

where $d_{i\text{A}}[k]$ is the Euclidean distance of robot A to sensor S_i and $d_{i\text{B}}[k]$ is analogous. For example

$$d_{1\text{A}}[k] = \sqrt{(x_\text{A}[k] - 2L)^2 + y_\text{A}^2[k]}, \qquad \text{(recall that } x_\text{A}[k] := x_\text{A}(t = kT_\text{s})\text{).}$$

Hint: In order to calculate the likelihood of a measurement $z_i[k]$, you can use the total probability theorem as follows:

$$f(z_i[k]|p[k]) = \sum_{s_i[k]=\{0,1\}} f(z_i[k]|s_i[k],p[k])f(s_i[k]|p[k])$$

where $p[k]$ is the vector of robot positions $p[k] := (x_\mathrm{A}[k], y_\mathrm{A}[k], x_\mathrm{B}[k], y_\mathrm{B}[k])$.

Independence of Noise Variables

The random variables $x_\mathrm{A}(0), y_\mathrm{A}(0), \theta_\mathrm{A}(0), x_\mathrm{B}(0), y_\mathrm{B}(0), \theta_\mathrm{B}(0), \{v_{s\mathrm{A}}[\cdot]\}, \{v_{s\mathrm{B}}[\cdot]\}, \{v_0, v_1, v_2, \ldots\}, \{s_i[\cdot]\}$, and $\{w_i[\cdot]\}$, with $i \in \{1,2,3,4\}$, are mutually independent.

Objective

The objective is to design a PF that estimates the locations and headings of the two robots. Your estimator will be called at every discrete time step k. The estimator has access to the previous posterior particles, the control input $u[k-1]$, and possibly the measurements $z_i[k]$, $i \in \{1,2,3,4\}$. Furthermore, the constants L, T_s, $\bar{v}_s$, $\bar{v}$, $\bar{w}$, and $\bar{s}$ are known to the estimator.

Provided Matlab Files

A set of Matlab files is provided on the class website.

`run.m`	Matlab script that is used to simulate the actual system, run your estimator, and display and evaluate the results.
`Estimator.m`	Matlab function template to be used for your implementation of the estimator.
`KC.m`	Matlab class with constants known to the estimator. Use the class like a struct: for example, the statement `KC.ts` accesses the sample time parameter.
`UKC.m`	Matlab class with constants not known to the estimator.
`simulateRobots.p`	Matlab function used to simulate the motion of the robots and to generate the measurements. This function is called by run.m, and is obfuscated (i.e. its source code is not readable).

Task

Implement your solution for the PF in the file `Estimator.m`. Your code has to run with the Matlab script `run.m`. You *must* use *exactly* the function definition as given in the template `Estimator.m` for the implementation of your estimator.
The number of particles is not defined in the problem, and you should tune this number, together with a roughening method, to get acceptable performance of your estimator. Furthermore, it is possible with the given measurement model that all your particles have zero measurement likelihood and therefore, all particle weights are zero as well. Your PF must handle this case appropriately.
The file `run.m` also outputs a performance measure based on a mean distance error (see the code for how it is computed). To give you an idea about acceptable performance, Michael's implementation of the PF achieved an average mean error of 0.6 m with a standard deviation of 0.13 m in 200 calls of `run.m` for the given default parameters.
The number of particles will affect the computation time of your implementation of the PF. The file `run.m` also outputs an average computation time for a single update of your PF. Your implementation should run with an average computation time for a single update below 1 second on Michael's laptop[1]. As a reference, Michael's implementation takes 0.13 seconds for a single update, on average.

[1]Core i7 CPU running at 2.5GHz, with 8GB of RAM.

Evaluation

For evaluating your solution, we will test your PF on the given problem data. Moreover, we will make suitable modifications to the parameters in `KC.m` and `UKC.m` and also test your estimator on those.

Deliverables

Note: Your submission has to follow these instructions exactly, as grading is automated. Submissions that do not conform will have points deducted.

Up to three students are allowed to work together on the programming exercise. They will all receive the same grade.

As a group, you must read and understand the ETH plagiarism policy here:
`http://www.plagiarism.ethz.ch/` – each submitted work will be tested for plagiarism. You must fill out the *Declaration of Orignality*, available here:
`http://tiny.cc/ETHPlagiarismForm`.

Hand in a single zip-file, where the filename contains the names of all team-members according to this template (note that there are no spaces in the filename):
`RE17Ex2_Firstname1Surname1_Firstname2Surname2_Firstname3Surname3.zip`.

The zip-file contains only two files:

1. Your implementation of the PF in the single file `Estimator.m`, which has the exact same function definition as in the provided template.
2. A scan of the *Declaration of Originality*, signed by all team-members.

Send your zip-file in a single email:

- Use this *exact* subject: `programming exercise 2 submission`
- In the email body, include a list of all team-member names.
- Send the email to `re2017@ethz.ch`.

You will receive an automatic reply confirming receipt of the email. A human will not look at the email before the deadline expires, and you are ultimately responsible that we correctly receive your solution in time. Late submissions will not be considered, nor will submissions where the attachment was forgotten.