SIMULTANEOUS LOCALIZATION AND MAPPING IN ROBOTICS (SLAM)

Since the 1990s, SLAM (Simultaneous Location and Mapping) has been one of the core Robotics problems, which have given rise to the highest number of contributions. Considering a mobile robot in an unknown environment, it consists in building a model of the world, and, concurrently, in localizing the robot in this map. The obtained metric map can be used for multiple purposes: navigation, etc.

SLAM combines data from proprioceptive sensors which provide information about the robot's movement (wheels encoders…), as well as exteroceptive sensors (vision, laser rangefinder…) which bring information on its environment. The most elementary version of the algorithm exploits geometrical features which are unambiguously identifiable between two perceptions. Using a maritime terminology, these primitives are termed "landmarks". When SLAM is based on image sequences, these landmarks can be door frames, quadrangular posters, etc.

Many approaches to SLAM have relied on Bayes filtering: non-linear extensions of Kalman filtering, information filters, particle filtering, etc. The state vector x to estimate is naturally made up of the location r of the robot and the positions $\mathsf{m}_1, \ldots, \mathsf{m}_M$ of the M landmarks, these vectors being expressed in the reference, world-related, frame. The prior dynamics of r can be based on the odometry of the robot, that is, on the reconstruction of its global motion from its kinematic model and wheel movements. The landmarks being static, the state subvectors $\mathsf{m}_1, \ldots, \mathsf{m}_M$ do not evolve over time. The vector of observations z to be assimilated by the filter is built from the raw relative measurements collected by the robot of its environment. Each subvector of z must be associated with the right landmark in $\mathsf{m}_1, \ldots, \mathsf{m}_M$, *i.e.*, with its counterpart inside x. This problem can be significantly simplified (or even eliminated) if the landmarks are properly designed.

The following cycle is then iteratively run: *[1. move the robot]* $\rightarrow$ *[2. match the sensed data]* $\rightarrow$ *[3. update the model]*. Step *[1.]* serves as the basis to the prediction of x between two successive time instants $k-1$ and k, prior to the assimilation of a measurement $\mathsf{z}_k = z_k$ at time k; step *[2.]* consists in associating z_k with the right sub-vector(s) m_m inside x, $m \in \{1, \ldots, M\}$; M (and the size of x) must be increased if the landmark relative to z_k is not already in x; step *[3.]* consists in assimilating z_k. The dimension of x thus grows over time as long as new landmarks appear. The size of z is likely to vary over time.

In general, the problem is strongly nonlinear. In the sequel, an "academic toy case study" is considered. The 2D world is fitted with the reference frame $(O, \overrightarrow{x}, \overrightarrow{y})$. Therein, a pointwise robot R starts from O and undergoes an approximatively circular motion around a known center C; this motion is subject to disturbances (slippage, etc.). M motionless landmarks $A_1, \ldots, A_M$ are laid in the environment. Their number M is known. The unknown absolute coordinates of the robot and the landmarks are respectively termed (u^R, v^R) and $(u^1, v^1), \ldots, (u^M, v^M)$.

The sensor placed on the robot (somewhat idealized, because it has no real physical equivalent), delivers, up to a measurement error, the entries in the frame $(R, \overrightarrow{x}, \overrightarrow{y})$ of vectors $\overrightarrow{RA_1}, \ldots, \overrightarrow{RA_M}$ which separate the robot from the landmarks, *provided the landmarks are visible.* In other words, the sensor delivers the set of components $l_m = \overrightarrow{RA_m}.\overrightarrow{x}, m_m = \overrightarrow{RA_m}.\overrightarrow{y}$ for any $m \in \{1, \ldots, M\}$ such that A_m is inside its field-of-view. The robot embedded sensor is assumed to be able to perfectly detect and label all the landmarks located its field-of-view.

The aim is to estimate (u^R, v^R) along time as well as $(u^1, v^1), \ldots, (u^M, v^M)$ from the sequence of past and current measurements. As M is given, the size of the hidden state vector is known. A prior knowledge is assumed available at initial time 0. Figure (1) illustrates the problem.

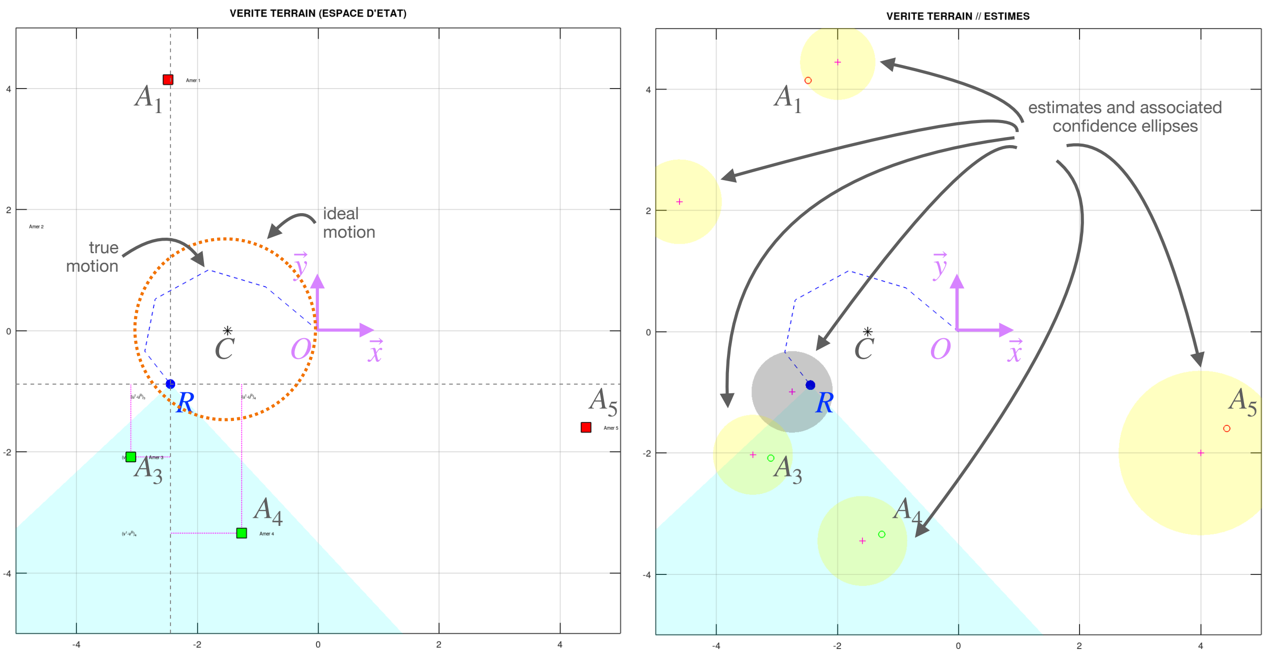

Figure 1: SLAM Toy Problem

Notations and Data

$\mathbb{I}$ and $\mathbb{O}$ respectively term the identity matrix and the matrix full of zeros, whose sizes can be derived from the context. The notation blkdiag$(A_1, \ldots, A_L)$ consists of the block-diagonal matrix made up with $A_1, \ldots, A_L$. The transpose operator is noted $'$. Finally, $\mathsf{x} \sim \mathcal{N}(\bar{x}, P)$ means that the (vector) random variable x follows a multivariate Gaussian distribution with mean $\bar{x}$ and covariance P. The associated probability density function $p_{\mathsf{x}}(x)$ is denoted by $\mathcal{N}(x; \bar{x}, P)$.

Let $\mathsf{r} = (u^R, v^R)'$, $\mathsf{m}_1 = (u^1, v^1)'$, $\ldots$, $\mathsf{m}_M = (u^M, v^M)'$. Let $\mathsf{x} = \big((\mathsf{r})', (\mathsf{m}_1)', \ldots, (\mathsf{m}_M)'\big)' \in \mathbb{R}^{2+2M}$ be the full hidden random state vector. At time k all of them are respectively denoted by r_k, u_k^R, v_k^R, $\mathsf{m}_{1,k}$, u_k^1, v_k^1, $\ldots$, $\mathsf{m}_{M,k}$, u_k^M, v_k^M, x_k.

If between two consecutive time steps the robot's trajectory moved along one eighth of a perfect circle centered on the point C of coordinates c, then the following equation would hold:

$$\mathsf{r}_{k+1} - c = F^R(\mathsf{r}_k - c), \quad \text{with } F^R = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}. \tag{1}$$

The landmarks are supposed to admit a zero dynamics, so they satisfy the equation

$$\forall m \in \{1, M\},\ \mathsf{m}_{m,k+1} = \mathsf{m}_{m,k}. \tag{2}$$

Due to unavoidable phenomena (slippage, etc.), which prevent the robot from perfectly following the ideal trajectory described by (1), the complete state vector x obeys a discrete-time stochastic state equation of the form

$$\mathsf{x}_{k+1} - b = F(\mathsf{x}_k - b) + \mathsf{w}_k, \tag{3}$$

where: the $(2+2M) \times (2+2M)$-dimensional dynamic matrix F writes as $F = \text{blkdiag}(F^R, \mathbb{I}, \ldots, \mathbb{I})$; the $(2+2M)$-dimensional dynamic noise w is white stationary and verifies $\forall k$, $\mathsf{w}_k \sim \mathcal{N}(0, Q_w)$, with $Q_w = \text{blkdiag}(Q_w^R, \mathbb{O}, \ldots, \mathbb{O})$ given $Q_w^R \in \mathbb{R}^2$; b is an adequate $(2+2M)$-dimensional column vector built on the basis of c and zeros.

The robot starts from O, so that the random state vector at initial time $k = 0$ is set to

$$\mathsf{x}_0 = (0, 0)'. \tag{4}$$

Let l_k^1, m_k^1, …, l_k^M, m_k^M be the (real-valued) measurements delivered by the sensor at any time $k \geq 1$, and z_k the vector gathering them. Depending on the values of k, the visibility of the landmarks changes, and so does the size of z_k. The measurement stochastic equation which links z and x writes as

$$\mathsf{z}_k = H_k \mathsf{x}_k + \mathsf{v}_k, \tag{5}$$

with $\mathsf{v}_k \sim \mathcal{N}(0, R)$ the measurement noise. The dimensions of the (given) observation and noise covariance matrices H_k and R may change with k. For all k, k', $\mathsf{w}_{0:k}$, $\mathsf{v}_{0:k'}$ are assumed jointly Gaussian and mutually independent.

The MATLAB function `dataSimulation.m` is provided, which prototype is described in Figure 2. This function simulates a random experiment. In addition to the number of time instants, the time horizon, the total number of landmarks and the random measurement vector sequence `Z`, it also outputs the genuine hidden random state vector sequence `X`, as well as many other useful matrices and vectors. It should be noted that if the experiment were conducted on a real robot, only `N`, `T` and `Z` would be given (although one could get the genuine `X` by means of a motion capture system by for example).

The function `ellipse` is also provided. According to its prototype in Figure 3, it plots the minimum volume confidence set (ellipsoid) of a bivariate Gaussian distribution, corresponding to a probability of 99 %.

WORK TO BE DONE

Simulation and Modeling

1. Run `dataSimulation`.
   (a) Explain the information provided by the graphic animation.
   (b) Give a meaning to all output variables.
   (c) Recall why the Kalman filter provides an exact solution to the Bayes estimation of the hidden state vector at any time on the basis of the available observations.
   (d) Implement this exact solution. *If you have not dealt with this question in 2ASRI, you are encouraged to devote some time to it in addition to the forthcoming questions specific to this lab!*
2. Write the closed-form mathematical expressions for
   (a) the probability density function (pdf) of the initial state (or "initial pdf") $p(\mathsf{x}_0)$;
   (b) the prior dynamics pdf $p(\mathsf{x}_k|\mathsf{x}_{k-1})$; *if necessary, a slightly modified version of this pdf will be used, in which the (unknown) position of the landmarks is assumed constant on average but subject to small random variations around this average value*;
   (c) the measurement pdf $p(\mathsf{z}_k|\mathsf{x}_k)$ (depending on the visibility of the landmarks).

Implementation of a SIS particle filter

3. Write the Sequential Importance Sampling (SIS) particle filter algorithm for the considered problem. The importance function will be set to the prior dynamics, possibly assuming that the (unknown) landmark positions are constant on average but subject to small random variations around their expected values.

4. Implement this algorithm for the initialization as well as the noise hypotheses identical to those output in `dataSimulation`. Run the filter on a sequence of measurements.

   (a) Deduce the Monte Carlo estimates of the genuine posterior means and covariances $\hat{x}_{k|k}$ and $P_{k|k}$ of the state vector x_k, conditioned on $\mathsf{z}_{1:k} = z_{1:k}$, that would be provided by the Kalman filter. Observe the evolution of these quantities.

   (b) On this basis, determine and plot the minimum-volume sets in which the outcomes of the vectors r_k, $\mathsf{m}_{1,k}$, ..., $\mathsf{m}_{M,k}$ should ideally lie with a given probability. For example, (and this is all the more true as we are in a linear Gaussian context),

   - at each instant k, on the basis of $\hat{x}_{k|k}$ and $P_{k|k}$, define for each component of the (hidden) state vector x_k, a "corridor" in which its outcome must lie with a theoretical probability of 99.7 % ;
   - (better!) at each time k, on the basis of $\hat{x}_{k|k}$ and $P_{k|k}$, define the confidence ellipses in which the outcomes of r_k, m_1, ..., m_M must lie with a theoretical probability of 99% (`ellipse` function).

5. Test the algorithm for various selected numbers of particles. What difficulties can be observed? Check the degeneracy of the weighted cloud by calculating the effective sample size.

SIR particle filter

6. Complete the above SIS algorithm with a resampling step, to be activated when the number of effective particles gets below a threshold (to be defined), in order to obtain the Sampling Importance Resampling (SIR) algorithm.

7. Test this new algorithm. To observe the "loop closure" phenomenon, where the observation of a landmark already seen in the past enables (thanks to correlations established a posteriori by the filter between the entries of x) to reduce the uncertainty on the position of other landmarks!

8. Compare the exact values of the moments $\hat{x}_{k|k}$ and $P_{k|k}$ of the filtering pdf computed by the Kalman filter with their estimates delivered by the particle filter on several "runs" of the latter, for various numbers of particles.

9. Propose an alternative definition of the importance function, enabling to sample the particles by taking into account both the prior dynamics of the robot and an information brought by the measurements.

Advanced Strategy

10. Implement and test a Rao-Blackwellized particle filter, such that: the position of the robot is estimated by means of a point-mass pdf; to each particle entering in this marginal pdf law is associated the conditional (Gaussian) pdf of the position of each landmark.

```
function [N,T,M,Z,arrayH,arrayR,in_fov,F,B,CC,Hfull,mX0,PX0,Qw,Rv,X] = dataSimulation(plot_p);
%DATASIMULATION
%  Simulation of a random experiment: robot motion, measurement collection, etc.
%  Syntax: [N,T,M,Z,arrayH,arrayR,in_fov,F,B,CC,Hfull,mX0,PX0,Qw,Rv,X] = dataSimulation(plot_p);
%
%  Input:
%  . plot_p : if 1, plots an animation, otherwise does nothing
%
%  Outputs:
%  . N: number of time samples
%  . T: vector of time samples (1xN)
%  . M: number of landmarks
%  . Z: 2MxN (2D) array of the outcomes of the measurement random process
%       (NaN entries correspond to unperceived landmarks; time is along the 2nd dimension)
%  . arrayH: 2Mx(2+2M)xN (3D) array of observation matrices
%            (NaN entries correspond to unperceived landmarks; time is along the 3rd dimension)
%  . arrayR: 2Mx2MxN (3D) array of measurement noise covariance matrices
%            (NaN entries correspond to unperceived landmarks; time is along the 3rd dimension)
%  . in_fov: 2MxN (2D) array of landmark visibility indexes
%            (NaN indexes correspond to unperceived landmarks; time is along the 2nd dimension)
%  -> as this is simulation, the following variables are also output
%     . F, B, CC: matrices/vectors involved in the prior dynamics of the full state vector
%                 (this (2+2M)x1 vector being made up with absolute coordinates of robot+landmarks)
%     . Hfull: 2Mx(2+2M) observation matrix if all landmarks were in the sensor fov
%     . mX0: (2+2M)x1 expectation vector of the initial state vector (at time 0)
%     . PX0: (2+2M)x(2+2M) covariance matrix of the initial state vector (at time 0)
%     . Qw: (2+2M)x(2+2M) covariance matrix of the (stationary) dynamics noise
%     . Rv: 2Mx2M covariance matrix of the (stationary) measurement noise if all landmarks were in the sensor fov
%     . X: (2+2M)xN (2D) array of the outcomes of the hidden state random process
%          (time is along the 2nd dimension)
%
%  Hint
%  . To suppress all lines full of NaN of a given matrix X, call X(any(~isnan(X),2),:);
%  . To suppress all lines and columns full of NaN for a given matrix X,
%    call X(any(~isnan(X),2),any(~isnan(X),1));
%
%  (c) Toulouse III Paul Sabatier University - P. Danès
```

Figure 2: Prototype of `dataSimulation.m`

```
function h = ellipse(mx,Px,color)
%ELLIPSE
% Draw the ellipse containing 99% of the realizations of a 2D Gaussian
% random variable with mean mx and covariance Px.
```

Figure 3: Prototype of `ellipse.m`