Autonomous Robot: Grid Localization

Guillaume Lemaître
Heriot-Watt University, Universitat de Girona, Université de Bourgogne
g.lemaitre58@gmail.com

I. Introduction

In this paper, we will present an implementation of grid localization allowing to localize the position and the motion of a robot. The implementation will be an example of a one-dimensional Mobile robot being moved, with constant velocity, in circular hallway. We will present the different steps that we followed to predict the position of the robot. First, we will present the update step. Then, we will introduce how we created the motion model histograms. We will conclude explaining the computation of the prediction using the motion model previously defined.

II. Updating state

During the implementation of the grid localization, the updating step was the first step realized. This step consists on corrected the predicted belief using the measurement. The environment is constituted of three different doors as shown on the figure 1.

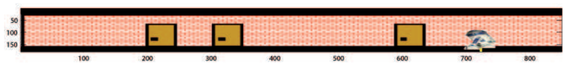

Figure 1. Representation of the environment

The sensor allowing measurements are considered like noisy. Hence, the probability density of the measurement is represented by Gaussians as shown in the figure 2.

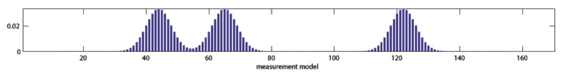

Figure 2. Probability density of the measurement

In this step, the predicted model was a uniform model. Updating model consists in two steps:

- If a measurement is taken, we have to correct the position of the robot using the measurement model shown in figure 1.
- Otherwise, we update the position of the robot using the prediction computed (this prediction will be computed in the last part).

This function is shown in the appendix A-C in the part *UPDATING*.

We can formalize this problem as follow:

```
if (measurement obtained) {
    $update_m = measurement_m \times previous_b$
    $normalize(update_m)$
}
else {
    else $update_m = predict_m$
}
```

Now, we will present the results obtained, using only updating and an initial uniform prediction model.

The result after the initialization is shown on the figure 3

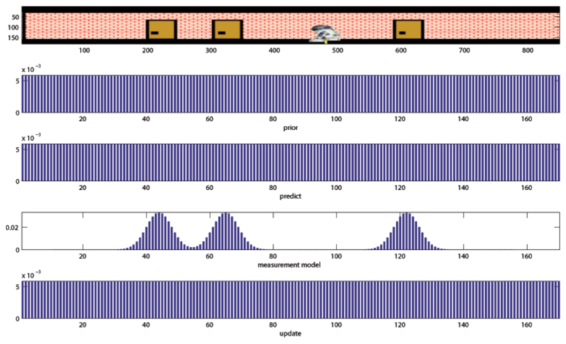

Figure 3. Initialization of the implementation

The result after the first measurement is shown on figure 4.

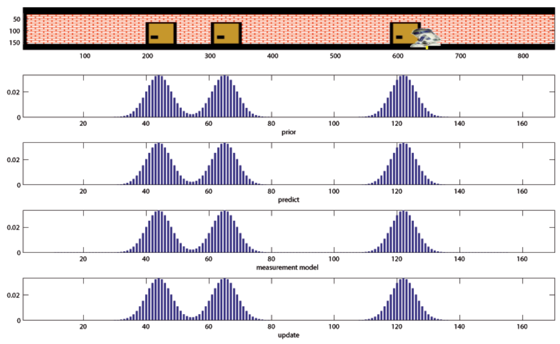

Figure 4. Updating after the first measurement

After several iteration, the result observed is as shown on figure 5.

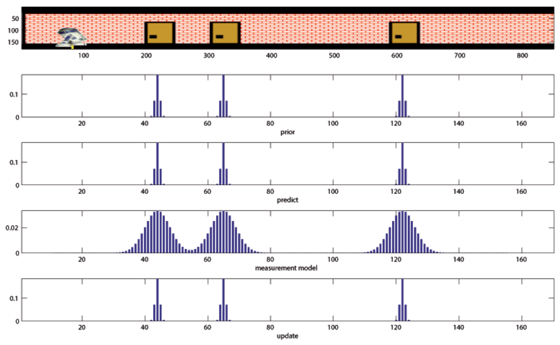

Figure 5. Updating after several measurements and updating

Iterations after iterations, the probability to be at the front of the doors is maximal and can explain because prediction is not still computed and only measurement model is influence the updating.

III. Motion model histograms

To compute the predicted belief we have to use a motion model. The velocity of the robot is normally constant. However, sensors giving the odometry are noisy. Hence, we have to construct different motion models depending of the different odometry measured which can appear. In this example, we considered that the noise added to the odometry measurement have a standard deviation of 5 cm. Hence, due to the *discrete factor*, five different motion model histograms have to be construct. The function allowing to construct the motion model histograms is shown in the appendix A-B. Figure 6 shows the different motion model histograms generated for different displacements u_k.

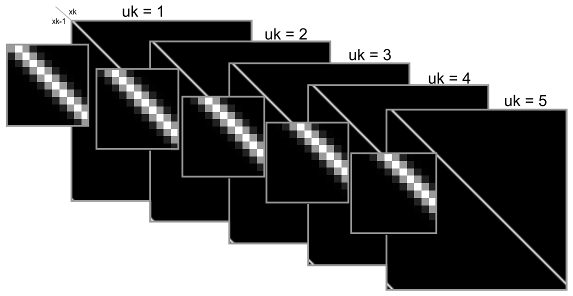

Figure 6. Different motion model histograms generated for different values of u_k

In figure 6, big figures show the different motion model obtained for value of u_k. The smallest figures show a zoom in on the top-right corner showing the shifting of the Gaussian of one cell between each consecutive motion model. For each motion model, x axis represents state x_k whereas y axis represents state x_{k-1}. Hence, each row represents:
$row_i = p(X_t|u_k = mod, X_{t-1} = x_i)$

IV. Prediction state

The prediction state is composed of two steps:

- The total probability theorem.
- The Bayes rule.

A. Total probability theorem

The first step of the prediction is to compute the total probability theorem. The total probability theorem is computed as follow:
$\bar{p_{k,t}} = \sum_i p(X_t|u_k = mod, X_{t-1} = x_i) \times p_{i,t-1}$
where $p(X_t|u_k = mod, X_{t-1} = x_i)$ is defined by the motion model defined in the previous section and $p_{i,t-1}$ is the previous belief.

This function is shown in the appendix A-C in the part *PREDICTION - Total probability theorem*.

B. Bayes rule

The second step of the prediction is to compute the Bayes rule. The Bayes is computed as follow:
$p_{k,t} = \eta p(z_t|X_t = x_k) \times \bar{p_{k,t}}$
where η is normalized number, $\bar{p_{k,t}}$ is the probability computed in the total probability theorem, $p(z_t|X_t = x_k)$ is given by the measurement model.

This function is shown in the appendix A-C in the part *PREDICTION - Bayes rule*.

C. Results of Grid localization

In this section, we will present and explain results obtained. Figure 3 shows the initial belief. After the first measurement, we can start to update and to know more precisely the position of the robot as shown on the figure 7.

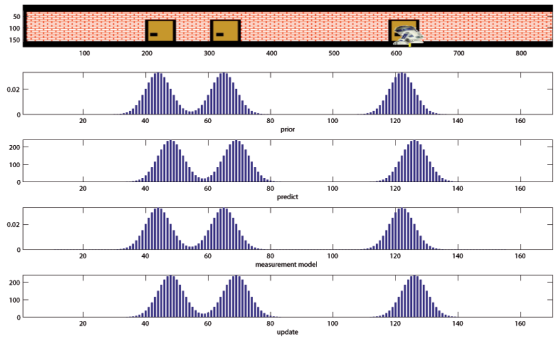

Figure 7. Localization after the first measurement

Before the first measurement, the position of the robot was unknown. After the phase of updating, as we explain in the section II, we multiply the prior belief by the measurement model. The prior belief was a uniform density model, so after this step, the predict model is equal to the measurement model.

The next step is a step of prediction. The position has to be predicted using the odometry. No measurements are given by the measurement sensors. Figure 8 shows the step of the prediction.

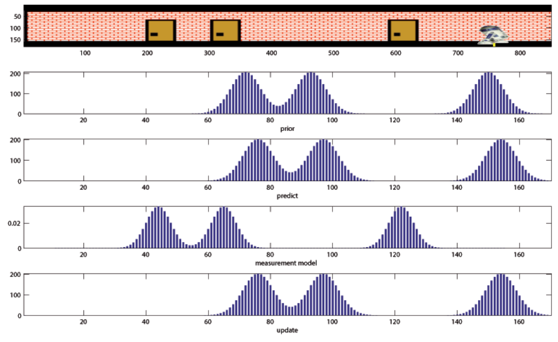

Figure 8. Prediction after the first measurement

The position is shifted using equation see in the section IV. Prediction position follows the real position of the robot. However, due to the incertitude of the odometry after each iteration the Gaussians are larger and larger as shown on the figure 9

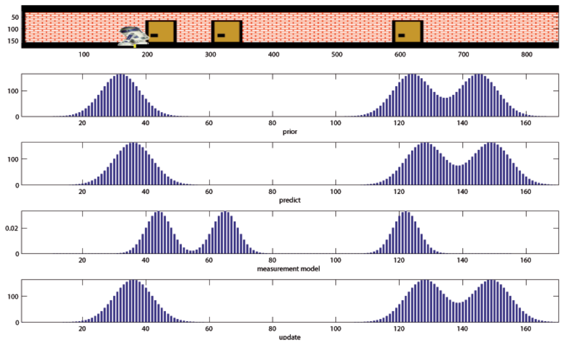

Figure 9. Prediction before the second measurement

During the next measurement, we are just corrected the predicted belief using equations of the section II. Figure 10 shown the prediction after the second measurement.

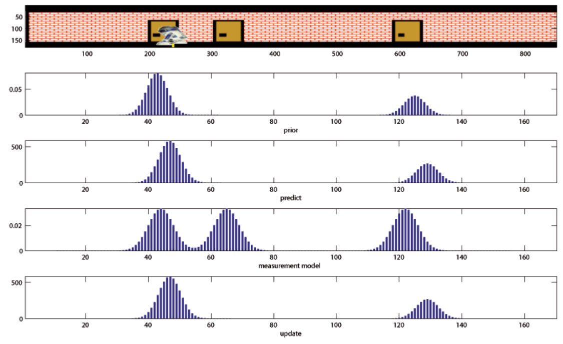

Figure 10. Prediction after the second measurement

The prediction starts to be better due to the updating. And after the third measurement, the prediction belief is exactly at the real position of the robot as shown on the figure 11.

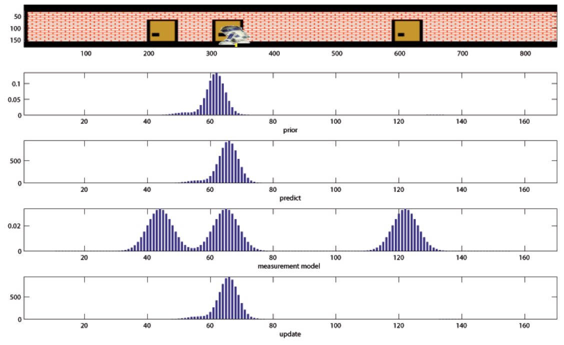

Figure 11. Prediction after the third measurement

After, we realized prediction step where we add noise due to the odometry as shown in the figure 12. Each update are used to correct the prediction as shown in the figure 13.

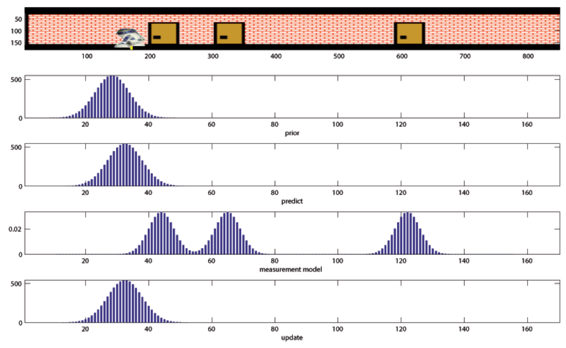

Figure 12. Prediction after convergence

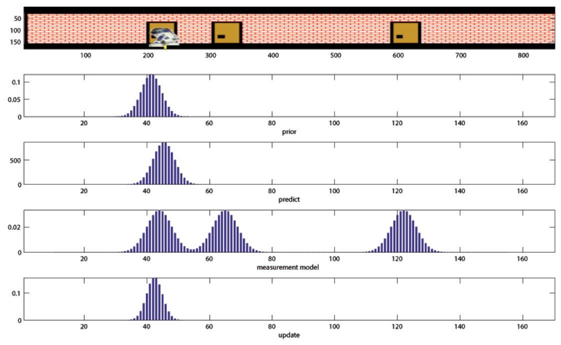

Figure 13. Updating to correct prediction

V. Conclusion

In this paper, we presented an implementation of the grid localization for an example of a one-dimensional Mobile robot being moved, with constant velocity, in circular hallway. First, we presented the updating step. Then, we introduce how we created the motion model. We concluded presenting the computation of the prediction model.

APPENDIX A
CODE

A. *GridLocalization1DOFRobotInTheHallway function*

```
function GridLocalization1DOFRobotInTheHallway
global frame

close all;

fprintf('Loading the animation data...\n');
load animation;
fprintf('Animation data loaded\n');


% Algorithm parameters
simpar.animate=1;                                       % 1: Draw the animation of the
%gaussian. 0: do not draw (speed up the simulation)
simpar.nSteps=500;                                      % number of steps of the algorithm
simpar.domain = 850;                                    % Domain size (in centimeters)
simpar.xTrue_0=[abs(ceil(simpar.domain*randn(1))); 20];
simpar.discrfactor = 5;                                 % Grid Resolution factor. The
%domain will be discretized to domain/discrfactor bins
simpar.numberOfCells=pos2cell(simpar.domain,simpar);
% Due to the odometry noise the sensed displacement uk will be not
% constant. a robot velocity of 20 cm/s and a sample time of T=1s means a
% displacement of 20 cm. Due to the 5 cm stdev noise in odometry we can get
% readings of 25, 15, 27, 13, hence assuming a cell of 5 cm (discr factor)
% we can use 5 uk control comands expressed in cells:
simpar.numberOfControlCommands=5;
simpar.wk_stdev=0;                                      % stddev of the noise used in
% accepleration to simulate the robot movement.
simpar.door_locations = [222,326,611];                  % Position of the doors
% (in centimetres). This is the Map definition.
simpar.door_cell_location = pos2cell(simpar.door_locations,simpar);
simpar.door_stdev=90/4;                                 % +-2sigma of the door observation is
% assumend to be 90 cm which is the wide of the door
simpar.door_cell_stdev=pos2cell(simpar.door_stdev,simpar);
simpar.odometry_stdev = 5;                              % Odometry uncertainty. Std.
% deviation of a Gaussian pdf. [cm]
simpar.T=1;                                             % Simulation sample time
simpar.bar_width = 0.5 ;                                % Width of the bars of the bar plots
% used to represent the histograms

xTrue_k=simpar.xTrue_0;

% Initial Robot belief is a flat histogram
belief_hist = ones(1,simpar.numberOfCells)/simpar.numberOfCells;

% STATE TRANSITION PROBABILITY HISTOGRAM
motion_model_hist = create_state_transition_table(simpar.numberOfCells,simpar.odometry_stdev,
 simpar);

% MEASUREMENT MODEL HISTOGRAM
% The histogram of measure model are three doors of 90cm aprox
door1_measurement_model=gaussian2histogram(simpar.door_cell_location(1), simpar.door_cell_stdev,
 simpar.numberOfCells);
bar(door1_measurement_model);title('door1_measurement_model, pres anykey');pause
door2_measurement_model=gaussian2histogram(simpar.door_cell_location(2), simpar.door_cell_stdev,
 simpar.numberOfCells);
bar(door2_measurement_model);title('door1_measurement_model, pres anykey');pause
door3_measurement_model=gaussian2histogram(simpar.door_cell_location(3), simpar.door_cell_stdev,
 simpar.numberOfCells);
bar(door3_measurement_model);title('door3_measurement_model, pres anykey');pause
measurement_model_hist=(door1_measurement_model'+door2_measurement_model'+
door3_measurement_model')/3;
bar(measurement_model_hist);title('Mesurement Model Histogram, pres anykey');pause

% The localization algorithm starts here %%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1:simpar.nSteps

     DrawRobot(xTrue_k(1), simpar); %Plots the robot

     xTrue_k_1=xTrue_k;
     xTrue_k=SimulateRobot(xTrue_k_1,simpar);  %Simulates the robot movement

     %Aplies the discrete bayes filter to localize the robot and to draw
     %the histograms

     uk=get_odometry(xTrue_k,xTrue_k_1,simpar);
     zk=get_measurements(xTrue_k(1),xTrue_k_1(1),simpar);

     fprintf('zk=%d uk=%f\n',zk,uk);

     belief_hist=Grid_localization(belief_hist,measurement_model_hist,motion_model_hist,uk,zk
     ,simpar);

end
% The Localization Algorith ends here %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B. *create_state_transition_table*

```
function motion_model_hist=create_state_transition_table(cells, odometry_stdev, simpar)

%%% Preallocation
motion_model_hist = zeros(cells,cells,simpar.numberOfControlCommands);

%%% Creation of motion model histogram
for j = 1:simpar.numberOfControlCommands
    for i = 1:cells
        motion_model_hist(i,:,j) = gaussian2histogram(pos2cell((i*simpar.discrfactor)
        +((j)*simpar.discrfactor),simpar), pos2cell(odometry_stdev,simpar),cells);
    end
end
```

C. *Grid_localization function*

```
%Aplies the discrete bayes filter and plots the results
function updated_belief_hist=Grid_localization(priorbelief_hist,measurement_model_hist,
 motion_model_hist, uk,zk,simpar)

% This lines must be replaced by your solution to the grid localization
% problem. They are provided only to allow for the execution of the program
% before solving the lab.

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PREDICTION
%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Preallocation
predict_hist = zeros(1,simpar.numberOfCells);

%%% Prediction model
%%% Allocation of probability vector
probability = zeros(1,simpar.numberOfCells);
%%%Convert uk in number of cell
ukn = pos2cell(uk,simpar);
disp('uk in number of cells: ')
disp(ukn)
%%% Number to compute n
prob = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PREDICTION - Total probability theorem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:simpar.numberOfCells
    for j = 1:simpar.numberOfCells
        probability(i) = probability(i) + (motion_model_hist(j,i,ukn)*priorbelief_hist(j));
        if i == j
            prob = prob + (motion_model_hist(j,i,ukn)*priorbelief_hist(j));
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PRDICTION - Bayes Rule
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Compute n
n = 1/prob;

%%% Update predictbelief
for i = 1:simpar.numberOfCells
    predict_hist(i) = probability(i)*n;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% UPDATING
%%%%%%%%%%%%%%%%%%%%%%%%%%


if (zk == 1)
    updated_belief_hist = measurement_model_hist.*priorbelief_hist;
    updated_belief_hist = normalize(updated_belief_hist);
else
    updated_belief_hist = predict_hist;
end


% plotting the histograms for the animation
DrawHistogram(2,'prior',priorbelief_hist,simpar);
DrawHistogram(3,'predict',predict_hist,simpar);
DrawHistogram(4,'measurement model',measurement_model_hist,simpar);
DrawHistogram(5,'update',updated_belief_hist,simpar);
```

D. Others functions already implemented

```
% Simulate how the robot moves %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function xTrue_kNew=SimulateRobot(xTrue_k, simpar)
%We will need to update the robot position here taking into account the
%noise in acceleration
wk=randn(1)*simpar.wk_stdev;
xTrue_kNew = [1 simpar.T; 0 1]* xTrue_k + [simpar.Tˆ2/2; simpar.T] * wk;

% We want the position of the robot to be in the range 0-850 (circular corridor)
xTrue_kNew(1) = mod(xTrue_kNew(1),850);

%Simulates the odometry measurements including noise%%%%%%%%%%%%%%%%%%
function uk=get_odometry(xTrue_k,xTrue_k_1,simpar)
uk=mod(xTrue_k(1)-xTrue_k_1(1)+simpar.odometry_stdev*rand(1),850);

%Simulates the detection of doors by the robot sensor %%%%%%%%%%%%%%%%
function sensor=get_measurements(xTrue_k,xTrue_k_1,simpar)
i=1;
```

```
sensor = 0;
while(i≤length(simpar.door_locations) && sensor ≠ 1)
   if ((xTrue_k(1) ≥ simpar.door_locations(i) && ...
    simpar.door_locations(i)≥ xTrue_k_1(1)) || ...
     (xTrue_k(1) ≤ simpar.door_locations(i) && ...
      simpar.door_locations(i)≤ xTrue_k_1(1))) && ...
       (abs(xTrue_k(1)-xTrue_k_1(1))<180)
      sensor = 1;
   end
   i = i + 1;
end

% Draws the robot
function DrawRobot(x, simpar)
global frame

if simpar.animate
    x = mod(x,simpar.domain);
    i=x*332/simpar.domain;

    % keep the frame within the correct boundaries
    if  i<1, i=1; end;
    if i>332, i=332; end;    if  i<1, i=1; end;

    subplot(5,1,1);
    image(frame(ceil(i)).image);  %axis equal;
end
drawnow;

%Plots a Gaussian using a bar plot %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function DrawHistogram(sp,label,pdf,simpar)

if simpar.animate
    subplot(5,1,sp);

    bar(pdf,simpar.bar_width);  axis tight;
    xlabel(label);
end

% Converts a gaussian pdf to its corresponding histogram %%%%%%%%%%%%%%
function histogram = gaussian2histogram(mean, stdev, cells)
%A little trick to simulate the circular corridor. I make a longer corridor
%and then, I recompose to the original size.
r2ps=sqrt(2*pi*stdev);
sq2=2*stdev^2;
histogram=zeros(cells,1);
for i=1:3*cells
    histogram(i) = exp((-(i-(cells+mean))^2)/sq2)/r2ps;
end
histogram=histogram(1:cells)+histogram(cells+1:2*cells)+histogram((2*cells)+1:3*cells);
histogram=normalize(histogram);



% Converts a position into a cell %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function cell=pos2cell(position,simpar)
cell=fix(position/simpar.discrfactor);

%Normalizes a histogram %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function nhistogram = normalize(histogram)
%We normalize the histogram
%We compute the normalizing factor
nfactor = 1/(sum(histogram));
nhistogram = histogram * nfactor;
```