Politecnico di Milano - Faculty of Engineering
Computer Science - Sound and Music Engineering
Model Identification and Adaptive Systems 2014/2015

MIAS Project Report

Indoor Tracking using Extended Kalman Filter

Marco Ciccone 818260

Abstract

The problem we want to address is the tracking of the trajectory of a person moving in an indoor environment using informations given by sensors disposed in fixed position of the environment.
First of all we're going to describe the problem of tracking and the methods that we can use to solve it. We will see a brief introduction to the trilateration approach and the we will concentrate on the Kalman Filter and its implementation for the tracking of a target in 2D. Then we will present our results and some future steps to improve the estimate.

Problem statement

The problem of tracking the trajectory of an object is very important for several applications. For instance we can think of tracking planes using radar measurements, or tracking a vehicle using GPS information, or also tracking moving objects in a video surveillance system.
In this specific case we want to track a moving person in an indoor environment using some sensors that are in fixed positions in the room. The sensors

are radio beacons and the target wears a receiver so that each beacon gives the distance of the target from itself. We are interested only in tracking the position of the target in the room so will focus only on the 2-D problem trying to identify the (x, y) coordinates.
First we want to consider a simpler problem: we simulate the case in which the sensors give their Euclidean distance from the target. This permits us to concentrate on the problem of tracking avoiding issues and errors due to the modeling of the channel of transmission. We refer to the i-th beacon as B_i and we indicate its position in the plane with (x_i, y_i). The distance formula of a point in the space from the i-th beacon is:

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}, \qquad (i = 1, 2, \dots n)$$

where i denotes the beacon number and n is the total number of beacons

The trilateration method

A classic approach in solving this positioning problem is to treat the coordinates of the target as the point of intersection of several circles (spheres in a 3D environment) whose centers are the locations of the n beacons. The equation of any circle is

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2$$

So what we have to do is solving a system of n nonlinear equations, but this is hard. Linearizing the system of equations geometrically converts the problem into one of finding the point of intersection of several lines (planes in 3D). We first consider the case of absence of noise. When the exact distances from three beacons are available, the solution of the linear system of equations is completely determined. In fact, in a 2D environment the theoretical minimum number of beacons is 3. When approximate or noisy distances are used, the position that is calculated by the direct solution of the linear equations is no longer acceptable. We need to use a Least Squares approach.

Consider that (x_1, y_1),(x_2, y_2),...,(x_n, y_n) are the beacons locations and (x, y) is a the target position. Since each beacon gives the distance from the

target we have n circle equations:

$$\begin{aligned}
&(x_1 - x)^2 + (y_1 - y)^2 = d_1^2, \\
&(x_2 - x)^2 + (y_2 - y)^2 = d_2^2, \\
&\dots \\
&(x_n - x)^2 + (y_n - y)^2 = d_n^2,
\end{aligned} \tag{1}$$

We can arbitrarily use one of the equations as linearizing tool, so we choose the n-th one. Subtract the n-th equation from each of the other equations we obtain:

$$\begin{aligned}
&(x_1 - x)^2 - (x_n - x)^2 + (y_1 - y)^2 - (y_n - y)^2 = d_1^2 - d_n^2, \\
&(x_2 - x)^2 - (x_n - x)^2 + (y_2 - y)^2 - (y_n - y)^2 = d_2^2 - d_n^2, \\
&\dots \\
&(x_i - x)^2 - (x_n - x)^2 + (y_i - y)^2 - (y_n - y)^2 = d_i^2 - d_n^2,
\end{aligned} \tag{2}$$

for each equation $i < n$. We can rearrange the terms in order to obtain linear equations:

$$\begin{aligned}
&2(x_n - x_1)x + 2(y_n - y_1)y = (d_1^2 - d_n^2) - (x_1^2 - x_n^2) - (y_1^2 - y_n^2), \\
&2(x_n - x_2)x + 2(y_n - y_2)y = (d_2^2 - d_n^2) - (x_2^2 - x_n^2) - (y_2^2 - y_n^2), \\
&\dots \\
&2(x_n - x_i)x + 2(y_n - y_i)y = (d_i^2 - d_n^2) - (x_i^2 - x_n^2) - (y_i^2 - y_n^2),
\end{aligned} \tag{3}$$

We can rewrite the earlier equations in matrix form $Ax = b$ where

$$A = \begin{bmatrix} x_n - x_1 & y_n - y_1 \\ x_n - x_2 & y_n - y_2 \\ \vdots & \vdots \\ x_n - x_i & y_n - y_i \end{bmatrix}, x = \begin{bmatrix} x \\ y \end{bmatrix},$$

$$b = \begin{bmatrix} 0.5[(d_1^2 - d_n^2) - (x_1^2 - x_n^2) - (y_1^2 - y_n^2)] \\ 0.5[(d_2^2 - d_n^2) - (x_2^2 - x_n^2) - (y_2^2 - y_n^2)] \\ \vdots \\ 0.5[(d_i^2 - d_n^2) - (x_i^2 - x_n^2) - (y_i^2 - y_n^2)] \end{bmatrix}$$

The solution of the system $x = A^{-1}b$ exist only when the number of equations equals the number of coordinates of the target. When we have more sensors than unknown we can solve the system using the least squares method:

$$x = (A^\top A)^{-1} A^\top b$$

The position estimation using the trilateration approach assumes that the target position lies at the point of intersection. In ideal condition the solution set will exist only if all the three or four circles intersect at only one point. Then the solution set will be existing, but in real scenarios, the circles do not intersect due to noise or insufficient measurements.

Tracking using Kalman Filter

In a real environment the measurements of the distances given by the sensors are noisy. At each sampling time we have new measurements, in practice we acquire new informations about the target that we can use to estimate its position: the state of the system. This is a classic application of the well known Kalman Filter. In mathematical terms we can describe the dynamical system as a pair of equations:

1. Process equation

$$x_{k+1} = F_k x_k + w_k \tag{4}$$

where F_k is the transition matrix taking the state x_k from time k to $k+1$. In the tracking problem this equation embodies the dynamics of the motion of the target, this is also called *motion equation*. Depending on the target to track, we need to choose the right equation that describe the motion of the object. In this work we will see two different type of motion equations and we will compare the results in the two cases. In classic Kalman filtering problem the process noise w_k is assumed to be additive, white, and Gaussian, with zero mean and with covariance matrix defined by:

$$E[w_n w_k^\top] = \begin{cases} Q_k, & for\, n = k \\ 0, & otherwise \end{cases}$$

2. Measurement equation

$$z_k = H_k x_k + v_k \tag{5}$$

where z_k is the vector of observable variables, or measurements, at time k, and H_k is the *measurement equation*. Also the measurement noise w_k is assumed to be additive, white, and Gaussian, with zero mean and with covariance matrix defined by:

$$E[v_n v_k^\top] = \begin{cases} R_k, & for\, n = k \\ 0, & otherwise \end{cases}$$

Unfortunately we can't directly apply the Kalman filter to the problem described before, because its classic formulation can be used only for linear dynamical system. Our tracking problem involves the use of nonlinear measurement equations since the observed variables are the Euclidean distance between the target and each beacon. To address the problem we need to use a linearized version of the Kalman filter called Extended Kalman Filter (EKF). The model of the system that we consider is in the form:

$$x_{k+1} = \mathbf{f}(k, \mathbf{x}_k) + w_k \tag{6}$$

$$z_k = \mathbf{h}(k, \mathbf{x}_k) + v_k \tag{7}$$

where as before, w_k and v_k are independent zero-mean white Gaussian noise processes with covariance matrices R_k and Q_k respectively.
The basic idea of the extended Kalman filter is to linearize the model equations at each time instant around the most recent state estimate of the state. We can obtain the linearized dynamical system finding the matrix F and H as the Jacobian of the function f and h respectively.

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}(k, \mathbf{x}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} \tag{8}$$

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}(k, \mathbf{x}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} \tag{9}$$

The following steps describe the EKF algorithm:

Algorithm 1: Extended Kalman Filter

- **Prediction**

1.Project the state ahead

$$\hat{x}_k = F\hat{x}_{k-1} + Gu_{k-1}$$

2. Project the covariance ahead

$$P_k = HP_kH^\top + WQ_kW^\top$$

- **Correction**

1. Compute the Kalman Gain

$$K_k = P_kH^\top(HP_kH^\top + VRV^\top)^{-1}$$

2. Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}_k + K_k(z_k - h(\hat{x}_k))$$

3. Update the error covariance

$$P_k = (I - K_kH)P_k$$

The matrices W and V are the Jacobian matrix of partial derivatives of state and output functions with respect to the process noise and measurement noise, respectively. P , R and Q are the covariance matrix of the error in the state estimate, measurement noise, and process noise, respectively.

Choosing the motion equation

Random walk

The simplest way of modeling the motion is to use a *random walk* model for each coordinate. In a random walk the states $\mathbf{x}$, update according to

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{w}_k$$

where $\mathbf{w}_k$ is a random process. Usually $\mathbf{w}_k$ is considered independent for each state, and depending on the distribution and variance , different random walk processes are obtained. How $\mathbf{w}_k$ should be distributed is determined by the intrinsic behavior of the states. Since we're using the Kalman Filter to estimate the state, an obvious choice is a normal distribution with an appropriate variance. Such a random walk is called Gaussian random walk, and if σ^2 is very small it resembles a Brownian motion. We will also refer to this model as Position (P) model.

Position Velocity Model

We can consider the linear time-invariant system

$$\dot{x}(t) = Ax(t) + Bu(t) + Gw(t)$$

and we discretize it sampling at the times $t_k = kT$. As we know the solution of the system of linear differential equation is:

$$x(t) = e^{A(t-t_k)}x(t_k) + \int_{t_k}^{t} e^{A(t'-t_k)}(Bu(t') + Gw(t'))dt'$$

we now assume that the input u_k changes slowly relative to the sampling period so that $u(t) \simeq u(t_k)$ on $[t_k, t_{k+1})$. This gives the discrete linear system

$$x(t_{k+1}) = Fx(t_k) + \bar{B}u(t_k) + w_k \tag{10}$$

where

$$F = e^{AT} \tag{11}$$

$$\bar{B} = \int_0^T e^{A(T-t')}Bdt' \tag{12}$$

$$w_k = \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-t')}Gw(t')dt' \tag{13}$$

If w_t is a white noise process the $\{w_k\}$ is a sequence with

$$Q_k = cov(w_k) = \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-t')}GQ_w(t')G^\top e^{A^\top(t_{k+1}-t')}dt' \tag{14}$$

and since w_t is also Gaussian, then $w_k \sim N(0, Q_k)$.
For our problem we consider only a 2nd-order model without any external force acting on the system $u(t) = 0$, so we have:

$$x_{k+1} = Fx_k + w_k \tag{15}$$

$$F = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ \frac{1}{2}T^2 & T \end{bmatrix} \sigma_w^2 \tag{16}$$

In practice the vector of the states is given by the position of the target and its velocity. It's very straightforward to extend from the one coordinate case

to the two coordinates case:

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} \frac{1}{3}T^3 & 0 & \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{3}T^3 & 0 & \frac{1}{2}T^2 \\ \frac{1}{2}T^2 & 0 & T & 0 \\ 0 & \frac{1}{2}T^2 & 0 & T \end{bmatrix} \sigma_w^2, \qquad x_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} \tag{17}$$

Finding the measurement equations

Euclidean distance model

At each time step k each beacon gives the measured distance between the target and the location of the beacon itself. For each beacon we have a measurement equation

$$z_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2} + v_{1,k} \tag{18}$$

$$z_2 = \sqrt{(x - x_2)^2 + (y - y_2)^2} + v_{2,k} \tag{19}$$

$$\vdots \tag{20}$$

$$z_n = \sqrt{(x - x_n)^2 + (y - y_n)^2} + v_{n,k} \tag{21}$$

where each $v_{i,k}$ is a measurement noise. As we said, since the equations are non linear, in order to apply the extended Kalman filter, we need to linearize them. We compute the Jacobian, considering only the position model in 2 dimensions, for the position-velocity model it needs just to add two columns of zeros for the partial derivatives with respect to the velocity components:

$$H = \begin{bmatrix} \frac{\partial z_1}{\partial x} & \frac{\partial z_1}{\partial y} \\ \frac{\partial z_1}{\partial x} & \frac{\partial z_1}{\partial y} \\ \vdots & \vdots \\ \frac{\partial z_n}{\partial x} & \frac{\partial z_n}{\partial y} \end{bmatrix} \tag{22}$$

The partial derivatives are :

$$\frac{\partial z_i}{\partial x} = \frac{x - x_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2}} \tag{23}$$

$$\frac{\partial z_i}{\partial y} = \frac{y - y_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2}} \tag{24}$$

Where each partial derivative is evaluated in the last estimate of the state. We make the assumption that the measurement noise associated with the distance measurements of a beacon is independent among the the beacons. This will result in diagonal covariance matrix for the measurement noise.

Dealing with missing measurements

In a real application the beacons have a specific operating area that could be in terms of a few meters. This means that if the target is not in the range of a beacon, the measure of the distance will not be available. For this reason, in order to distinguish the case in which we don't have information from the case in which the target is exactly in the position of the beacon, we need to adapt the corresponding i-th row of the matrix H setting it to zero when we don't have the measurement from the i-th beacon.

$$H = \begin{bmatrix} \frac{x-x_1}{\sqrt{(x-x_1)^2+(y-y_1)^2}} & \frac{x-x_1}{\sqrt{(x-x_1)^2+(y-y_1)^2}} \\ \frac{x-x_2}{\sqrt{(x-x_2)^2+(y-y_2)^2}} & \frac{x-x_2}{\sqrt{(x-x_2)^2+(y-y_2)^2}} \\ \vdots & \vdots \\ 0 & 0 \\ \vdots & \vdots \\ \frac{x-x_n}{\sqrt{(x-x_n)^2+(y-y_n)^2}} & \frac{x-x_n}{\sqrt{(x-x_n)^2+(y-y_n)^2}} \end{bmatrix} \xleftarrow{i-th\ beacon\ measurement\ is\ absent} \tag{25}$$

The Jacobian is evaluated in the last state estimate $\hat{x}_k$.

RSSI model

We want to consider also a more realistic case using RSSI sensors. An RSSI sensor (Received Signal Strength Indicator) gives a measure of the distance between the transmitter and the receiver using a measure of the attenuation of the power of radio-signal in the communication channel. Using a model of the attenuation of the radio communication channel the measurement equation assume the form of :

$$z_k = h_k(x) = P_{d_0} - 10\alpha \log_{10}\left(\frac{R}{R_0}\right) - wallLoss \tag{26}$$

Here $h(x)$ is the power receiver in decibel, P_{d_0} is the received power at the reference distance or initial RSSI value at one meter distance in dBm, α is

path loss exponent, it can be varied from 2 to 6 depending on the propagation environment, R_0 is the reference distance (1m). These are the characteristic parameters of the channel. $R = \sqrt{(x - x_i)^2 + (y - y_i)^2}$ is the distance between transmitter and receiver. Finally, $wallLoss$ is the sum of the losses introduced by each wall. This factor depends on the building layout, construction material, reflecting surfaces, local based infrastructures and object movement. Also in this case we need to linearize the measurement equations in order to use the EKF algorithm. Rewriting $h(x)$ we can compute the partial derivatives:

$$h(x) = P_{d_0} - 5\alpha \log_{10}\left((x - x_i)^2 + (y - y_i)^2\right) - wallLoss \tag{27}$$

$$\frac{\partial h(x)}{\partial x} = \frac{-10\alpha(x - x_i)}{\ln(10)((x - x_i)^2 + (y - y_i)^2))} \tag{28}$$

$$\frac{\partial h(x)}{\partial y} = \frac{-10\alpha(y - y_i)}{\ln(10)((x - x_i)^2 + (y - y_i)^2))} \tag{29}$$

$$\tag{30}$$

for each beacon $i = 1, 2, \ldots n$. So H becomes:

$$H = \begin{bmatrix} \frac{-10\alpha(x-x_1)}{\ln(10)((x-x_1)^2+(y-y_1)^2))} & \frac{-10\alpha(y-y_1)}{\ln(10)((x-x_1)^2+(y-y_1)^2))} \\ \frac{-10\alpha(x-x_2)}{\ln(10)((x-x_2)^2+(y-y_1)^2))} & \frac{-10\alpha(y-y_2)}{\ln(10)((x-x_2)^2+(y-y_2)^2))} \\ \vdots & \vdots \\ \frac{-10\alpha(x-x_n)}{\ln(10)((x-x_n)^2+(y-y_n)^2))} & \frac{-10\alpha(y-y_n)}{\ln(10)((x-x_n)^2+(y-y_n)^2))} \end{bmatrix} \tag{31}$$

Results

In this section we explain the results of the tracking obtained in two different simulation environment comparing the motion and measurement models that we have described above. For the sake of this project we don't consider the trilateration method aforementioned but we focus only on the performance of the EKF tracking. We have mentioned it only to show an alternative to the Kalman Filter approach, moreover the trilateration method requires that we have at least 3 measurements in order to estimate the position of the target and this is not always possible in a real environment.
We will show the results of the EKF tracking with P and PV model using the Euclidean Distance, and the result of the same experiment considering the RSSI model.

In order to evaluate the results we use the following metrics:

$$AvgDistanceError = \frac{\sum \sqrt{(x-\hat{x})^2 + (y-\hat{y})^2}}{N} \tag{32}$$

$$\tag{33}$$

where the summation is over all the points that have been estimated by the algorithm, (x, y) are the real coordinates of each point, $(\hat{x}, \hat{y})$ are the estimated coordinates, N is the number of point that have been estimated. Another metric that was used is the Root Mean Square Error (RMSE). This metric computes the error in localization separately each coordinate and squares it. The sum of all errors are computed, divided by the number of estimates, and the square root is taken of the resulting value. The benefit of the RMSE is that the error in localization of the X and Y coordinates is available. The X and Y RMSE values can be combined and result in the Net RMSE that describes the net error.

$$RMSE_x = \sqrt{\frac{\sum (x-\hat{x})^2}{N}} \tag{34}$$

$$RMSE_y = \sqrt{\frac{\sum (y-\hat{y})^2}{N}} \tag{35}$$

$$RMSE_{Net} = \sqrt{RMSE_x^2 + RMSE_y^2} \tag{36}$$

For the experiments we have simulated two trajectories (T_1 and T_2) in a room of dimensions 30×10 m. The experiments are performed with 9 sensors sparse in the room in the following positions:

Table 1: Positions of the beacon in the simulated room 30×10 m

	Position
B_1	(7.5 , 3.5)
B_2	(15.0 , 5.0)
B_3	(20 , 3.5)
B_4	(20 , 7.5)
B_5	(7.5 , 7.5)
B_6	(10.0 , 10.0)
B_7	(10.0 , 2.0)
B_8	(17.5 , 10.0)
B_9	(17.5 , 2.0)

The sensors gives the measurements with a sampling time $\Delta T = 5$.Now we compare and discuss the results obtained by each method with the two trajectories.
For the simulation of the measurement, we compute the distance between the target and each beacon, and if it's greater than the operative area we don't consider it in the computation. Then we add a white gaussian noise $v_k \sim WGN(0, 0.1)$ to the measurements.
Each experiment has been simulated for 100 times and the performance indexes has been averaged.

Table 2: Performance trajectory T_1 using P-model and varying the operative area of the beacons

	5m	**6m**	**7m**
$DistanceErrorAvg$ **[m]**	0.70974	0.34939	0.29523
$RMSE_x$ **[m]**	0.8013	0.28297	0.23005
$RMSE_y$ **[m]**	0.91782	0.30549	0.25425
$RMSE_{net}$ **[m]**	1.2349	0.41975	0.34304

Table 3: Performance trajectory T_2 using P-model and varying the operative area of the beacons

	5m	**6m**	**7m**
$DistanceErrorAvg$ **[m]**	1.8917	0.62085	0.37193
$RMSE_x$ **[m]**	2.3206	0.39743	0.2505
$RMSE_y$ **[m]**	1.5675	0.84435	0.37869
$RMSE_{net}$ **[m]**	2.9265	0.93497	0.45449

Table 4: Performance trajectory T_1 using PV-model and varying the operative area of the beacons

	5m	**6m**	**7m**
$DistanceErrorAvg$ **[m]**	0.84004	0.4643	0.42446
$RMSE_x$ **[m]**	0.88099	0.43865	0.41433
$RMSE_y$ **[m]**	0.93928	0.33284	0.26985
$RMSE_{net}$ **[m]**	1.3316	0.55229	0.49479

As expected the performances increase if we increase the operative area of the sensors. In fact, a broader range permits to have more beacons measurements and so a more accurate estimate. The tracking performance of

Table 5: Performance trajectory T_2 using PV-model and varying the operative area of the beacons

	5m	**6m**	**7m**
$DistanceErrorAvg$ **[m]**	2.2113	0.78882	0.65402
$RMSE_x$ **[m]**	3.1149	0.68468	0.60607
$RMSE_y$ **[m]**	1.0982	0.62407	0.44732
$RMSE_{net}$ **[m]**	3.3291	0.93325	0.75463

trajectory T_2 are lower with respect to T_1. This is due to the fact that T_2 simulates the case in which the path of the target is almost at the limit of the operative area of the sensors, causing that we have less measurements available for the estimate.

Now we consider the case of RSSI measurements. As we can see from the results, the estimate with RSSI model is in general less accurate than the one with the simple euclidean distance, in fact attenuations and approximation are involved in this model. Moreover it depends on the characteristic of the radio communication channel and varying the parameters can affect the performance.
We tested two cases, the first one with P model and measurement noise with variance $\sigma_2 = 0.1$. The parameters of the channel used for the simulation are the same used by Peerapong et.Al in *"Hybrid Technique for Indoor Positioning System based on Wi-Fi Received Signal Strength Indication"*.

Table 6: Summary of the parameters used for PV model with RSSI measurements

Name	Symbol	Value
RSSI at reference distance	P_{d_0}	3dBm
Reference distance	R_0	1m
Path loss exponent	α	3

For the second case we tested the PV model with RSSI measurement using the parameters of the work of Peerapong et Al. We decided to try a different variance for the measurement noise to understand if the method can work also in other conditions. Moreover the value of the variance $\sigma^2 = 4.35$ comes directly from long observations in an office environment so it resembles a more realistic situation. The covariance matrix of the error measurement

Table 7: Performance trajectory T_1 using P-model and RSSI measurements varying the operative area of the beacons

	5m	**6m**	**7m**
$DistanceErrorAvg$ **[m]**	0.96169	0.5075	0.28793
$RMSE_x$ **[m]**	1.3898	0.77734	0.29988
$RMSE_y$ **[m]**	1.071	0.375	0.18228
$RMSE_{net}$ **[m]**	1.7835	0.88721	0.35133

Table 8: Performance trajectory T_2 using P-model and RSSI measurements varying the operative area of the beacons

	5m	**6m**	**7m**
$DistanceErrorAvg$ **[m]**	2.4672	1.4126	0.47993
$RMSE_x$ **[m]**	3.3588	1.6666	0.41993
$RMSE_y$ **[m]**	1.6665	1.2445	0.36086
$RMSE_{net}$ **[m]**	3.7502	2.1571	0.55439

defined by Peerapong et Al. is

$$R = I_{4\times4} \times 10^{4.565}$$

A critical parameter is the covariance matrix of the process noise. In order to improve the performance we need to choose it in a proper way. A lot of papers estimate the covariance matrix from data or by trials and simulations to see which values fit best. Also here some tests have been made trying different scaled version of the Q matrix defined for the PV model until we obtained sufficient results.

Table 9: Performance trajectory T_1 using PV-model and RSSI measurements varying the operative area of the beacons

	5m	**6m**	**7m**
$DistanceErrorAvg$ **[m]**	2.2327	2.0477	2.0429
$RMSE_x$ **[m]**	2.3896	2.2609	2.335
$RMSE_y$ **[m]**	1.7418	1.5737	1.519
$RMSE_{net}$ **[m]**	2.9877	2.7854	2.8109

We can see that with a stronger measurement noise the performance are more or less the same even if we increase the operative area of the sensors.

Table 10: Performance trajectory T_2 using PV-model and RSSI measurements varying the operative area of the beacons

	5m	**6m**	**7m**
$DistanceErrorAvg$ **[m]**	1.7464	1.5768	1.486
$RMSE_x$ **[m]**	1.3057	1.2531	1.2415
$RMSE_y$ **[m]**	1.4956	1.3146	1.2008
$RMSE_{net}$ **[m]**	1.9974	1.8239	1.7381

Conclusions and future works

The experiments with the PV model show that the EKF displays relatively large errors when there's a change of direction in the trajectory. The Kalman filter's estimates require a few iterations before it can correctly track the position of the target after a change in the direction. This may be due to the fact that the characteristics of the process noise of the simulated motion of the target are not really Gaussian as assumed by the Kalman filter. For this reason it may be interesting to test the performance of other algorithm such as the Particle Filter.
In the case of the P model, the assumption that velocity and acceleration are just random noise allows it to take these turns with less error. The PV model is forced to maintain rigidly to the Kinematic equations. We can see from figures that the P model seems to make many small errors at each step, the estimate is jerky with small oscillation around the true value, whereas the PV estimate is smoother but makes large errors, typically when we have a change of direction.
The results that we have obtained agree with the result of Shareef and Zhu in *"Localization Using Extended Kalman Filters in Wireless Sensor Networks"*.
We need also to consider errors due to the linearization of the system. In fact the EKF algorithm is a suboptimal method with respect to the classic Kalman filter because of the approximation of the linearization.

These results shows that in the case of the tracking of a person in a indoor environment where we have a lot of turning and change of direction because of walls and obstacles, a Position-Velocity model may not be the best choice. A future work may be to investigate other models that suits better to the human motion.

The results with the RSSI measurements have ample room for improvement, in fact in order to use the EKF with RSSI sensors in a real application it's necessary to tune correctly the parameters of the communication channel model. As mentioned before, it could be interesting to compare the performance of the EKF with the Particle Filter following the reference work of Fredrik and Martin Karlsson : *"Sensor Fused Indoor Positioning Using Dual Band WiFi Signal Measurements"*.

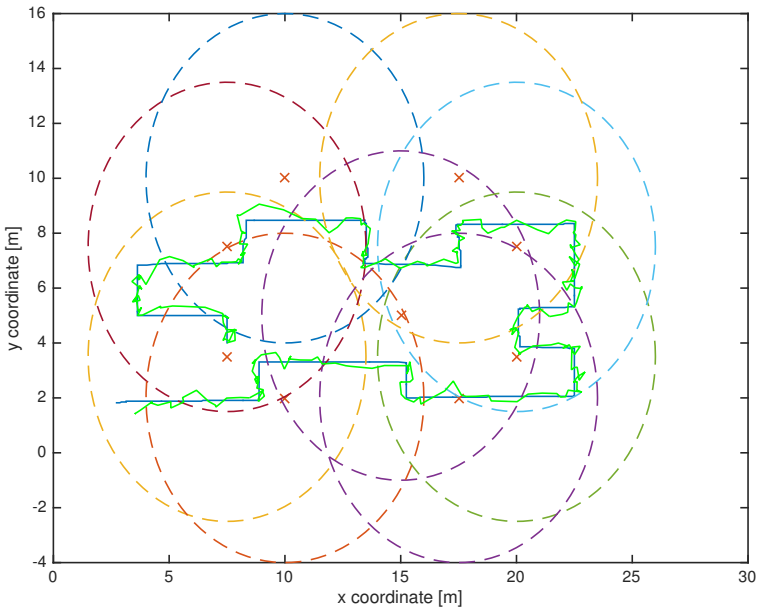

Figure 1: P model Trajectory T_1

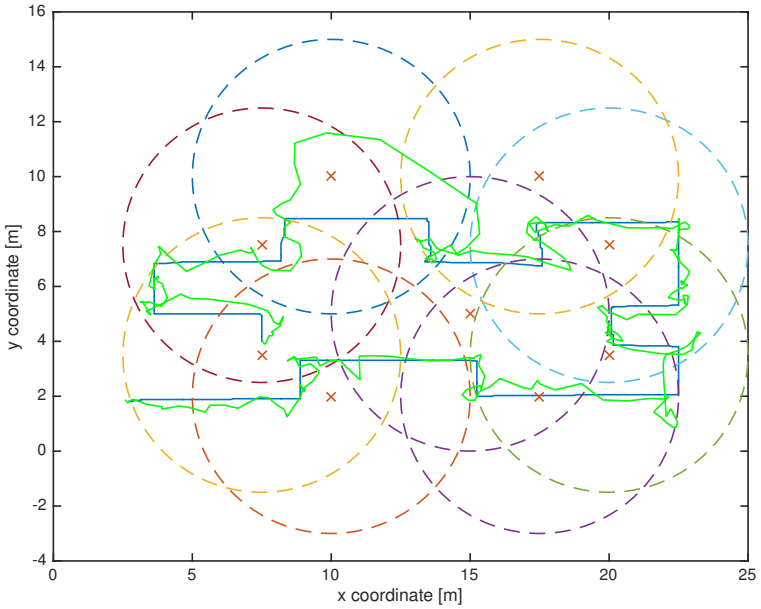

Figure 2: PV model Trajectory T_1

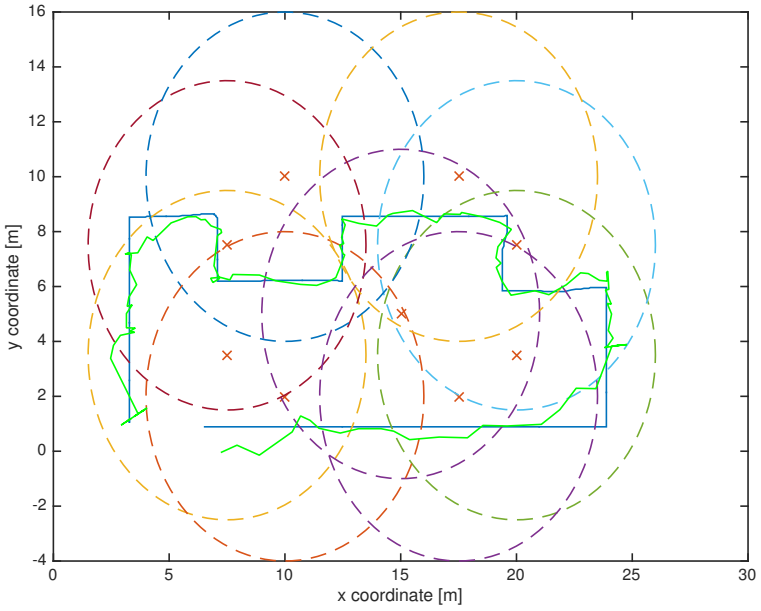

Figure 3: P model Trajectory T_2

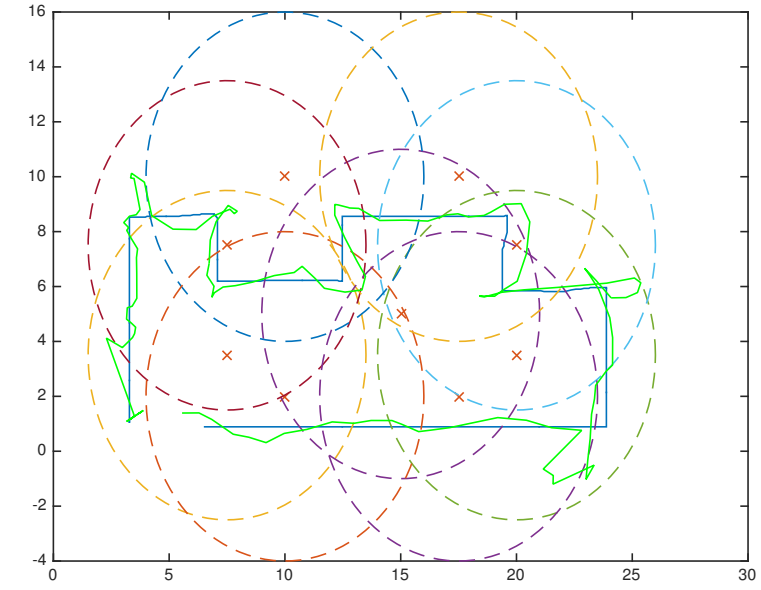

Figure 4: PV model Trajectory T_2

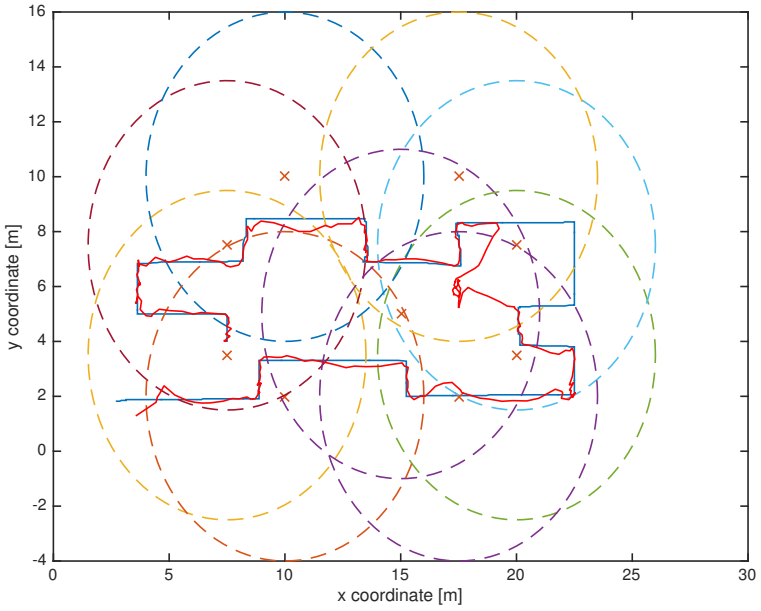

Figure 5: P model with RSSI measurement of Trajectory T_2

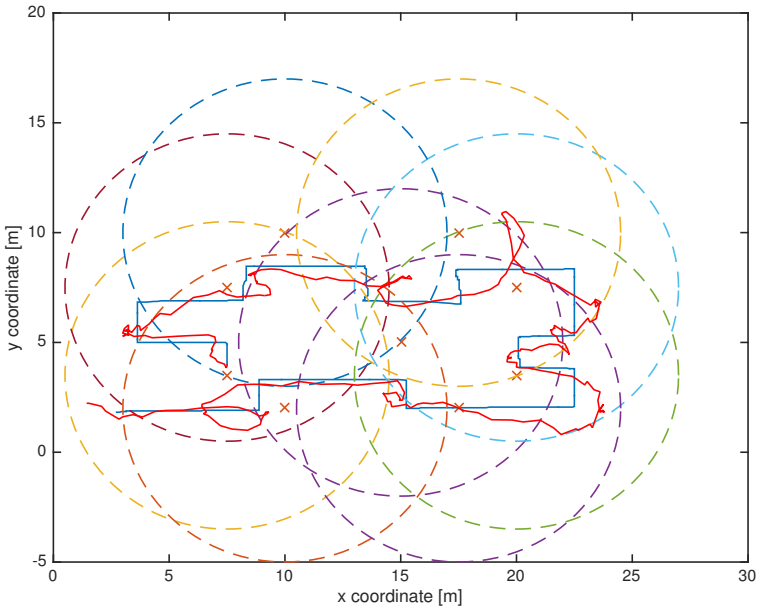

Figure 6: PV model with RSSI measurement of Trajectory T_2

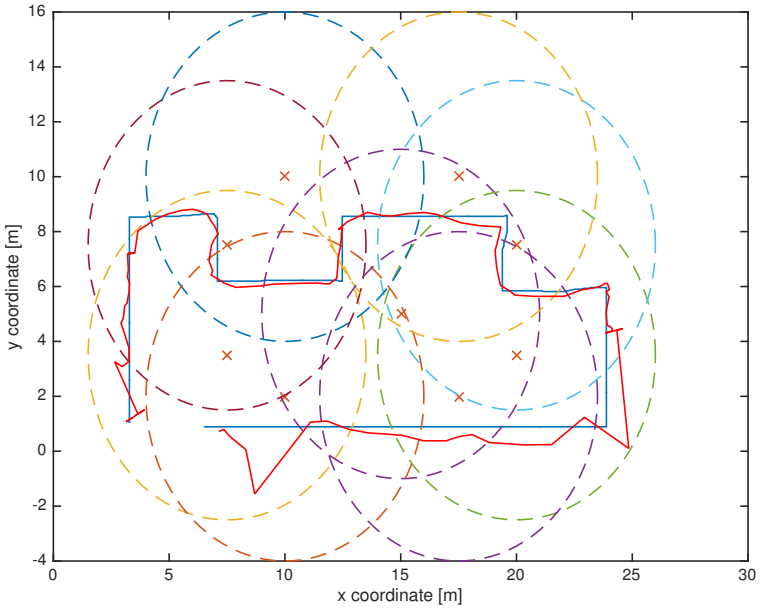

Figure 7: P model with RSSI measurement of Trajectory T_2

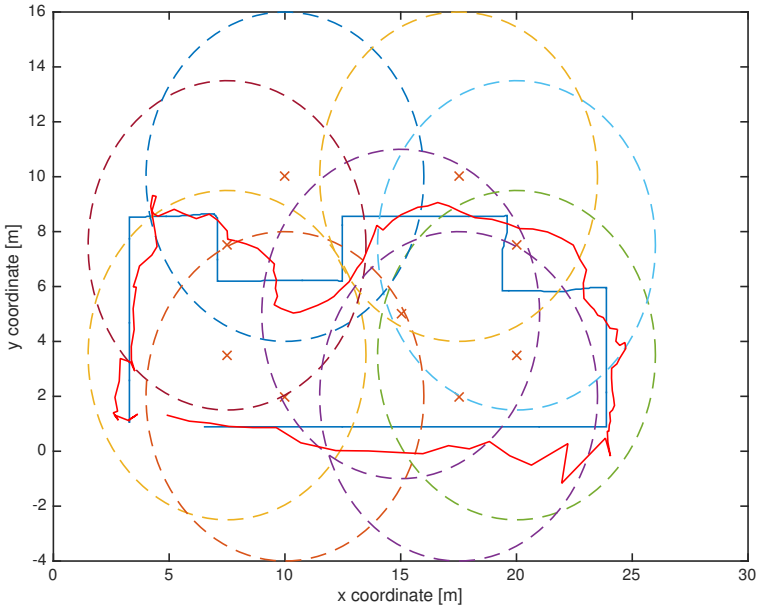

Figure 8: PV model with RSSI measurement of Trajectory T_2