12th International Conference on Information Fusion
Seattle, WA, USA, July 6-9, 2009

Information Fusion for Indoor Localization*

Pierre Blanchart
Telecom ParisTech
46 rue Barrault
75013 Paris, France
pierre.blanchart@gmail.com

Liyun He, François Le Gland
INRIA Rennes
campus de Beaulieu
35042 Rennes, France
liyun.he@gmail.com
legland@irisa.fr

Abstract – *This paper describes a fusion approach to the problem of indoor localization of a pedestrian user, in which PNS measurements, cartographic constraints and ranging or proximity beacon measurements are combined in a particle filter approximation of the Bayesian filter. Some critical issues are also addressed, such as taking the constraints into account, monitoring the degeneracy of the weights and the sample depletion in terms of the effective sample size, detecting track loss, and recovering from a detected loss.*

Keywords: information fusion, pedestrian navigation system (PNS), cartographic constraints, ranging beacon, proximity beacon, particle filtering

1 Introduction

In general terms, information fusion aims at combining different sources of information to reach a given objective, where each single source of information alone would fail to reach the assigned objective. Here, the objective is to estimate the position of one or several individual pedestrian users walking inside a building, so that a satellite–based solution, such as the GPS (global positioning system), could not be used in this context. Three different sources of information are used here:

- a pedestrian navigation system (PNS) unit provides drifting measurements of the pedestrian heading, as well as noisy measurements of the walked distance,
- noisy measurements of the distance between the user and a ranging beacon, or merely a binary detection information when the user is within some (small) distance of a proximity beacon: it is assumed that beacons are well–localized and well–identified, but there could be a limited number of these beacons in the whole building, so that these measurements are not frequently available,
- finally, information of a different nature provided by a map of the building, which lists obstacles (essentially walls) to the user walk: this really brings some useful information and should not be considered as a nuisance, and in some extreme cases, taking these constraints into account properly could already be sufficient to reach the localization objective, even with no beacon available around.

*This work has been supported by French National Research Agency (ANR) through Telecommunication program (project FIL, no. ANR-07-TCOM-006)

2 Modelling

The localization problem considered here can be described as follows. The state vector $x_k = (r_k, \theta_k)$ at time t_k is defined as the user 2D–position r_k and its orientation, represented as an angle θ_k or equivalently as the unit 2D–vector $u(\theta_k)$ where

$$u(\theta) = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} .$$

Let

$$d_k = |r_k - r_{k-1}| \quad \text{and} \quad \alpha_k = (\theta_k - \theta_{k-1}) ,$$

denote the true walked distance and direction change in the time interval between t_{k-1} and t_k. Clearly, the state vector x_k is related to the state vector x_{k-1} and to the pair (d_k, α_k) by the relation

$$r_k = r_{k-1} + d_k\, u(\theta_k) \quad \text{and} \quad \theta_k = \theta_{k-1} + \alpha_k .$$

In practice, the true walked distance and direction change are not known, but noisy PNS measurements $(\widehat{d}_k, \widehat{\alpha}_k)$ are provided instead, from which PNS estimates $(\widehat{r}_k, \widehat{\theta}_k)$ are obtained as

$$\widehat{r}_k = \widehat{r}_{k-1} + \widehat{d}_k\, u(\widehat{\theta}_k) \quad \text{and} \quad \widehat{\theta}_k = \widehat{\theta}_{k-1} + \widehat{\alpha}_k .$$

These position and orientation estimates, based on PNS measurements only, are know to diverge from the true position and orientation, and additional measurements should be used. To merge the different sources of information, a Bayesian approach is adopted, and the idea

is to exploit the PNS measurements $(\widehat{d}_k, \widehat{\alpha}_k)$ in a different way, so as to obtain a random model for the evolution of the unknown position and orientation. Indeed, $(\widehat{d}_k, \widehat{\alpha}_k)$ are noisy measurements of the true walked distance and direction change (d_k, θ_k), with the following random model for the error

$$d_k = \widehat{d}_k + w_k^{\text{walk}} \quad \text{and} \quad \alpha_k = \widehat{\alpha}_k + b_k + w_k^{\text{turn}} \ ,$$

with additive noises w_k^{walk} and w_k^{turn}. The bias b_k could be modelled either as a constant or as a Gauss–Markov random sequence, in which case it should be incorporated into the state vector. Only the simplest case where $b_k \equiv 0$ has been considered in the numerical experiments presented in Section 5. Therefore, the model for the evolution of the unknown state vector is

$$\begin{aligned} r_k &= r_{k-1} + (\widehat{d}_k + w_k^{\text{walk}})\ u(\theta_k) \ , \\ \theta_k &= \theta_{k-1} + \widehat{\alpha}_k + w_k^{\text{turn}} \ . \end{aligned} \tag{1}$$

Next, if a ranging beacon located at position a is active at time t_k, then it provides a noisy measurement of the distance between the user and the beacon, as

$$z_k = |r_k - a| + v_k^{\text{range}} \ , \tag{2}$$

with additive noise v_k^{range}. If a proximity beacon located at position a is active at time t_k, then it provides a binary detection characterized by the probability of detection

$$\mathbb{P}[\text{user is detected} \mid |r_k - a| = d] = P(d) \ . \tag{3}$$

as a function of the distance between the user and the beacon. Ideally, $P(d) = 1$ if $d \leq d_0$ and $P(d) = 0$ otherwise, where d_0 denotes the range of the proximity sensor, but a less sharp form of the function $P(d)$ could be used alternatively to take mis–detection into account.

Clearly, the true position, and also the transition between two successive true positions, do respect the constraints. This information should also be incorporated in the localization procedure, by enforcing that the transition between two successive unknown positions should also respect the constraints. How to take these constraints into consideration is precisely the purpose of the next Section.

3 Map–based Bayesian filtering

Estimating the user 2D–position and its orientation, based on sensor measurements and on constraints, can be formulated as a Bayesian filtering problem. In full generality, MMSE (minimum mean–squere error) estimates could be obtained in terms of the posterior probability distribution $p(x_k \mid z_{0:k})$ of the state x_k at time t_k given a sequence $z_{0:k} = (z_0, \cdots, z_k)$ of past sensor measurements. Here, it is assumed that the sequence of hidden states forms a Markov chain, characterised by its initial probability distribution $p(x_0)$, which represents the uncertainty about the initial hidden state x_0 at time t_0, and by its transition probability distributions $p(x_k \mid x_{k-1})$, which represent the uncertainty about the hidden state x_k at time t_k, if the hidden state x_{k-1} at time t_{k-1} would be known exactly. It is also assumed that the measurements are mutually independent, if the hidden states would be known exactly, and the sensor model is characterized by the probability distribution $p(z_k \mid x_k)$, which is called the likelihood function, seen as a function $x_k \mapsto p(z_k \mid x_k)$ of the hidden state.

The Bayesian filter satisfies the following recurrence equations: the *prediction* equation

$$p(x_k \mid z_{0:k-1}) = \int p(x_k \mid x_{k-1})\ p(x_{k-1} \mid z_{0:k-1})\ dx_{k-1} \ ,$$

and the *correction* or *update* or *filtering* equation

$$p(x_k \mid z_{0:k}) \propto p(z_k \mid x_k)\ p(x_k \mid z_{0:k-1}) \ ,$$

which is simply the Bayes rule, providing the posterior distribution as the normalized product of the prior distribution and the likelihood function.

The different terms involved in these equations can be made more explicit for the model introduced in Section 2. In view of (2), the likelihood function associated with ranging mesurements is easily defined in terms of the probability distribution of the additive noise v_k^{range}, and in view of (3), the likelihood function associated with proximity detection is explicitely defined as the probability of detection. In view of (1), the transition probability distributions implicitely depend on the PNS measurements and are easily defined using

$$\begin{aligned} p(x_k \mid x_{k-1}) &= p(r_k, \theta_k \mid r_{k-1}, \theta_{k-1}) \\ &= p(r_k \mid r_{k-1}, \theta_{k-1}, \theta_k)\ p(\theta_k \mid r_{k-1}, \theta_{k-1}) \\ &= p(r_k \mid r_{k-1}, \theta_k)\ p(\theta_k \mid \theta_{k-1}) \ , \end{aligned}$$

in terms of the probability distributions of the additive noises w_k^{walk} and w_k^{turn}. For example, if w_k^{walk} and w_k^{turn} are independent Gaussian random variables with zero mean and variance σ^2_{walk} and σ^2_{turn} respectively, then $p(\theta_k \mid \theta_{k-1})$ is a 1D Gaussian distribution with mean $\theta_{k-1} + \widehat{\alpha}_k$ and variance σ^2_{turn}, and $p(r_k \mid r_{k-1}, \theta_k)$ is a 2D Gaussian distribution with mean $r_{k-1} + \widehat{d}_k\ u(\theta_k)$ and degenerate covariance matrix $\sigma^2_{\text{walk}}\ u(\theta_k)\, u^*(\theta_k)$, non-degenerate in the direction $u(\theta_k)$ only. Note that these transition probability distributions provide a model for a user walking in unconstrained space, i.e. without obstacles, and do not take constraints into account.

Several different models are presented below, that take constraints into account, in the sense that invalid

transitions (x_{k-1}, x_k), that do not respect the constraints, are not possible under these alternate models. In other words, the idea is to replace the unconstrained transition probability distributions $p(x_k \mid x_{k-1})$ with map–consistent transition probability distributions $p(x_k \mid x_{k-1}, \texttt{map})$.

3.1 Restriction to a Voronoi graph

In the approach proposed in [8], a model is directly specified on a graph, which by construction respects the constraints. In the problem considered here, the PNS provides information of a quite different nature, namely a walked distance and a heading. This 2D information is indeed not necessarily consistent with restricting the transitions to a graph, which is a 1D structure. Of course, a transition could be first proposed in the unconstrained space and then projected back onto the graph, but the resulting transition could differ too much from the true transition walked by the user, since the graph edges are not necessarily oriented like the transition provided by the PNS. This would result in a prior probability distribution not consistent with the likelihood function, and a loss in accuracy.

3.2 Direct rejection (hard constraint)

In view of the discussion above, the preferred approach is to propose transitions in the unconstrained space, and then to reject invalid transitions. This could be achieved using a hard acceptance/rejection procedure, where a transition from $x = (r, \theta)$ to $x' = (r', \theta')$ is called invalid if the straight line joining the initial position r to the terminal position r' crosses an obstacle, as shown in Figure 1. The resulting map–consistent transition probability distributions are defined as

$$p(x_k \mid x_{k-1}, \texttt{map}) \propto \begin{cases} p(x_k \mid x_{k-1}) & \text{valid transition} \\ 0 & \text{invalid transition} \end{cases}$$

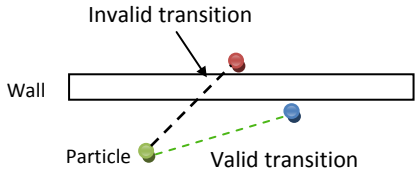


Figure 1: Invalid transition

However, there is a risk that the probability of acceptance is too small, i.e. that too many proposed transitions will be declared invalid, with the result that the number of particles alive will decrease dramatically.

3.3 Shortest path control (soft constraint)

Another simple way to measure whether a transition from $x = (r, \theta)$ to $x' = (r', \theta')$ does respect the constraints is to look at the shortest path a user would need to walk through within the environment represented by the map, to go from the initial position r to the terminal position r'. This path is conditioned on local cartography, that is, walls or other obstacles or objects the user has to go around. A simple acceptance/rejection procedure is to look whether the shortest path length $s(x, x')$ in the environment, between the initial position r and the terminal position r', is consistent with the measured walked distance $\widehat{d}_k$ provided by the PNS, as shown in Figure 2. One simple and sound way to achieve this objective is to use a cost function depending of the difference between $s(x, x')$ and $\widehat{d}_k$, associated with a Gaussian assumption for instance. The resulting map–consistent transition probability distributions are defined in this case as

$$p(x_k \mid x_{k-1}, \texttt{map}) \propto \exp\{-\frac{(s(x_{k-1}, x_k) - \widehat{d}_k)^2}{2\,\sigma^2}\}\, p(x_k \mid x_{k-1}) \ . \tag{4}$$

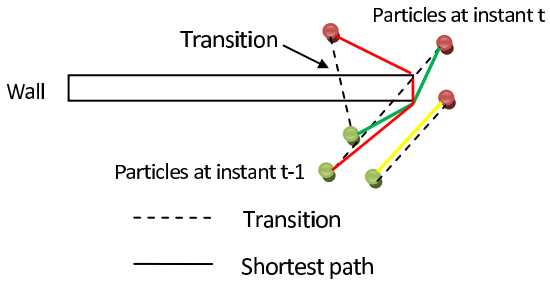


Figure 2: The shortest path illustration

How to implement the shortest path approach will be explained later in Section 5.2.

4 Particle filtering

The key idea behind particle filtering is to use weighted samples, also called particles, to approximate the posterior probability distribution $p(x_k \mid z_{0:k})$ of the state x_k at time t_k given a sequence $z_{0:k} = (z_0, \cdots, z_k)$ of past sensor measurements. In other words

$$p(x_k \mid z_{0:k}) \approx \sum_{i=1}^{N} w_k^i \, \delta_{\xi_k^i} \quad \text{with} \quad \sum_{i=1}^{N} w_k^i = 1 \ ,$$

where $(\xi_k^1, \cdots, \xi_k^N)$ denotes the particle positions, and $(w_k^1, \cdots, w_k^N)$ denotes the particle (positive) weights. In its simplest and very intuitive version, these particles propagate according to the state equation and constraints are easily taken into consideration, and as new

measurements arrive, the particles are re–weighted to update the estimation of the state. Beyond just weighting the samples, the weights could also be used more efficiently to resample, i.e. to select which samples are more interesting than others and deserve to survive and get offsprings at the next generation, and which samples are behaving poorly and should be discarded. There are many different ways to generate an independent N–sample from a weighted empirical distribution, which all reduce to specifying how many copies (or clones, replicas, offsprings, etc.) will be allocated to each particle. The simplest method is to sample independently, with replacement, from the weighted empirical distribution, which is done efficiently by simulating directly order statistics associated with a uniform N–sample. Other resampling procedures which reduce the variance are stratified sampling or residual resampling.

For the model introduced in Section 2, the particles positions and weights are updated as follows. If the hard constraint approach is used, then for any $i = 1, \cdots, N$, a particle ξ_k^i is proposed according to the probability distribution $p(x_k \mid \xi_{k-1}^i)$ in the unconstrained space, and the transition (ξ_{k-1}^i, ξ_k^i) is accepted if it is valid, otherwise it is rejected, which results in the loss of the corresponding particle.

If the soft constraint approach is used, then for any $i = 1, \cdots, N$, a particle ξ_k^i is proposed according to the probability distribution $p(x_k \mid \xi_{k-1}^i)$ in the unconstrained space, with the corresponding weight

$$\exp\{-\frac{(s(\xi_{k-1}^i, \xi_k^i) - \widehat{d}_k)^2}{2\,\sigma^2}\} \ .$$

The weights just act like weights coming from a sensor model, through the evaluation of a likelihood function, and they are used in a resampling step, to select the more realistic transitions.

When a ranging beacon is active, the corresponding likelihood function is evaluated at each particle position to incorporate the ranging measurement into the particle weight, i.e.

$$w_k^i \propto w_{k-1}^i \, p(z_k \mid x_k^i) \ , \tag{5}$$

where w_{k-1} is the weight of the i–th particle at time t_{k-1}. If the additive noise v_k^{range} is Gaussian, with zero mean and variance σ^2_{range}, then

$$w_k^i \propto w_{k-1}^i \ \exp\{-\frac{(z_k - d_k^i)^2}{2\,\sigma^2_{\text{range}}}\} \ , \tag{6}$$

where z_k is the ranging measurement and $d_k^i = |r_k^i - a|$ is the distance between the active ranging sensor located at position a and the i–th particle $\xi_k^i = (r_k^i, \theta_k^i)$ located at position r_k^i. Notice that the orientation θ_k^i does not appear explicitly in the expression of the weight w_k^i.

4.1 Effective sample size

Resampling can avoid the degenerate situation where all but one of the weights are close to zero. However, especially if hard constraint is used, the rejection of particles corresponding to invalid transitions results in a loss of the diversity of particle population, which in turn can induce a tracking loss, given that state estimation provided by PNS measurements alone has a growing error with time. One strategy to benefit from the positive effect of resampling but to keep diversity is to reduce the resampling frequency, or even better to resample only when the degeneracy problem is detected.

One suitable measure of the degeneracy problem is the effective sample size [4, Section III], defined as

$$N_{\text{eff}} = \frac{1}{\sum_{i=0}^{N'} (w_k^i)^2} \ ,$$

where w_k^i is the normalized weight of the i–th particle. The effective sample size is always smaller or equal to the number N' of particles alive, i.e. with positive non–zero weights. Equality means that all the particles have the same weight. A too small effective sample size value indicates a severe degeneracy problem, and an adaptive startegy is to resample when the effective sample size value is smaller than a threshold

$$N_{\text{eff}} < \beta' \, N' \quad \text{with} \quad 0 < \beta' \leq 1 \ .$$

With the hard constraint model, the number N' of particles alive can only decrease. Before N_{eff} attains the prescribed threshold, it may happen that there is already not enough particles alive. As the reduction of the effective sample size is induced by the degeneracy of the weights distribution and by the decrease in the number of particles alive, the definition of criterion should be modified. Instead of comparing N_{eff} with N', it is advisable to compare it with a fixed population size N, which is the desired number of particles alive, i.e. the number of particles after the last resampling step, or in other words

$$N_{\text{eff}} < \beta \, N \quad \text{with} \quad 0 < \beta \leq 1 \ , \tag{7}$$

to take the decrease in the number of particles alive into account. In view of the identity

$$\frac{N_{\text{eff}}}{N} = \frac{N_{\text{eff}}}{N'} \ \frac{N'}{N} \ ,$$

the novel form (7) monitors two phenomena: the degeneracy problem and the loss of particles. Since the rejected particles corresponding to invalid transitions have indeed zero weights, then obviously

$$1/N_{\text{eff}} = \sum_{i=0}^{N'} (w_k^i)^2 = \sum_{i=0}^{N} (w_k^i)^2 \ ,$$

which means that after all the new form is completely consistent with the usual form. Moreover, a minimum value is guaranteed for the number of particles alive, which makes the filtering algorithms more robust.

4.2 Loss detection

In situations where the user is walking in open space and has not met any ranging beacon for a long time, the particle cloud begins to scatter because state estimation provided by PNS measurements alone has a growing error with time. When the particle cloud is too far from the true state, the filtering algorithm is in trouble. Different strategies can be proposed to recover from this situation, but there is also a need to detect this loss automatically.

An approach has been proposed in [9], which compares a short–term average of the weights with a long–term average. A novel idea is to compare the measured distance z_k between the user and the ranging beacon, and the computed distances $(d_k^1, \cdots, d_k^N)$ between each particle and the same beacon, assuming that the user meets a ranging beacon. If none of the computed distances matches the measured distance, then it is reasonable to consider that the particle cloud cannot represent the unknown user position, and as a result all the particles will have too small weights, which means that a loss has occurred. This results in the following criterion

$$\max_{i=1,\cdots,N} \exp\{-\frac{(z_k - d_k^i)^2}{2\,\sigma^2_{\text{range}}}\} < \alpha \ , \qquad (8)$$

where z_k is the measured distance provided by the ranging beacon, $d_k^i = |r_k^i - a|$ is the distance between the i–th particle and the active ranging beacon located at position a, and $0 < \alpha < 1$ is the threshold. This is equivalent to controlling the minimum difference between the measured distance and any computed distance, and the new criterion is

$$\min_{i=1,\cdots,N} |z_k - d_k^i| > \rho\ \sigma_{\text{range}} \ , \qquad (9)$$

where $\rho = \sqrt{-2\,\log\alpha}$ is a given threshold.

4.3 Loss recovery (particles injection)

Once the loss happens, the filter should be reinitialized, by generating particles randomly over the state space. However, the cost of the reinitialization depends a lot on the environment size and the initial probability distribution of particles. One more precise and less–costing approach is to add particles randomly around the ranging beacons that has just been met by the user [9]. This reinitialization procedure is called *particles injection*.

The particles injection can take place automatically once the loss is detected or be carried out manually through the tracking server. The particles are injected randomly, according to the ranging sensor measurement z_k and to the probability distribution of the additive noise v_k^{range}.

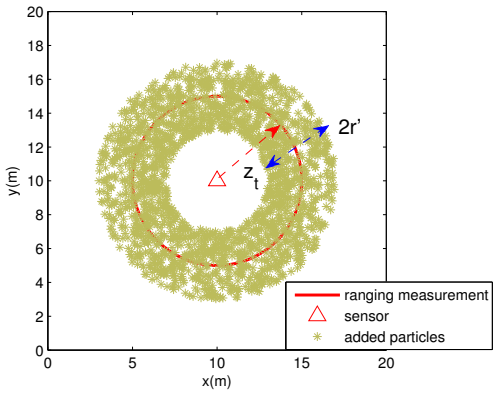


Figure 3: Posterior distribution of the pedestrian's pose given by ranging beacon

5 Simulation

Some implementation issues are discussed here, and a short presentation of the simulator is given.

5.1 Acceptance/rejection with retry

To avoid the roughness of the *acceptance/rejection* model, instead of deleting all invalid transition particles immediately, they are given several chances to try to advance according to the same PNS measurements but with different noise realization. If all the chances fail, then the particle is thrown out. Otherwise, the particle will be added to the set of living particles.

Figure 4 shows an example of how an invalid transition particle survives. The green spot is the particle at time t_{k-1} and the red one is the proposed particle at time t_k, and the corresponding transition crosses a wall. The particle at instant t_{k-1} was given five chances to try with different noise realization each time. The transition is valid at the fourth try. Then the blue spot representing the new particle at time t_k is kept for the next iteration.

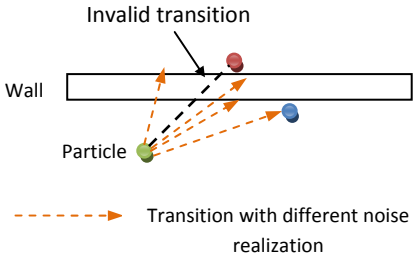


Figure 4: Invalid transition survived with chance

There is a different way to exploit the surviving with chance model, which reduces to *prior boosting*. Instead of just giving another chance to particles with invalid

transitions, all the particles at time t_{k-1} are given a certain number of chances. The terminal states of all the valid transitions are accepted for the next filtering iteration.

The surviving with chance model ensures that particles near an obstacle have more chance to survive during the prediction step. It works especially well in the case that there are many corridors, which are usually narrow, because it can avoid the situation that particles are rejected too easily.

5.2 Computation of shortest path

One efficient way to compute the shortest path on the map between two state hypotheses x_{k-1} and x_k is to use the map *visibility graph*.

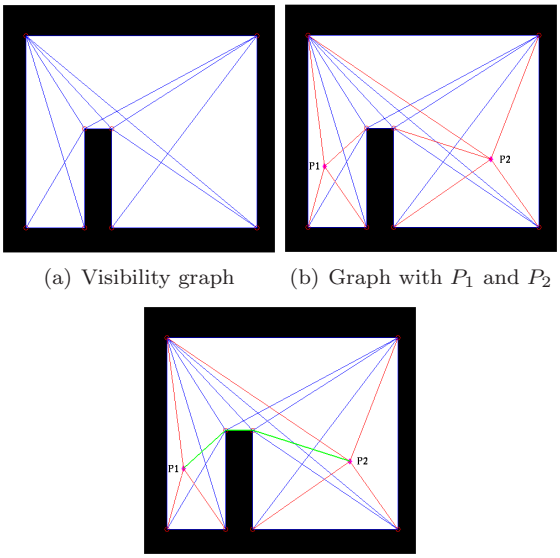
A map can be modelled as a set S of disjoints simple polygons in the plane which are called obstacles. The visibility graph is the graph whose nodes are the vertices of S, as shown in Figure 5(a). There is an edge between vertices v and w if they can see each other, that is, if the segment joining v to w does not intersect the interior of any obstacle in S. Two vertices that can see each other are called (mutually) visible and the segment connecting them is called a visibility edge.

When we want to compute the shortest path between two points P_1 and P_2, as shown in Figure 5(b), we just add these two points to the adjacency matrix of the visibility graph, considering that P_1 or P_2 are adjacent to an edge of the visibility graph when there exists a direct linking between P_1 or P_2 and that edge without crossing any obstacles. Finally applying the Dijkstra algorithm with Fibonacci heaps, we can get the shortest path between P_1 and P_2 from the obtained graph 5(b), as the green one in the Figure 5(c).

The following Figure 6 shows an execution of the algorithm in the case that we use only inertial sensors measurements with known initial (position and orientation) state. The khaki dash–dot line is the trajectory obtained from inertial sensors measurements only. The violet dashed line is the estimated trajectory obtained from inertial sensors measurements and incorporating also cartographic information: it is very close to the blue solid line, which represents the true trajectory. No ranging beacons measurements are used in this example.

5.3 Tracking simulator

A complete tracking simulator has been developped which allows to show the evolution of particles in relation to the true path. The simulation is a fusion of PNS measurements, cartographic constraints and ranging beacon measurements. A map of the building, the coordinates of the ranging beacons and a Voronoi graph automatically generated from the map are loaded in the database before the simulation begins. The Voronoi graph is used to provide a road map, and to generate the true trajectories automatically. The additive noises



(a) Visibility graph (b) Graph with P_1 and P_2

(c) Shortest path on graph between P_1 and P_2

Figure 5: Impermeable surface treatments

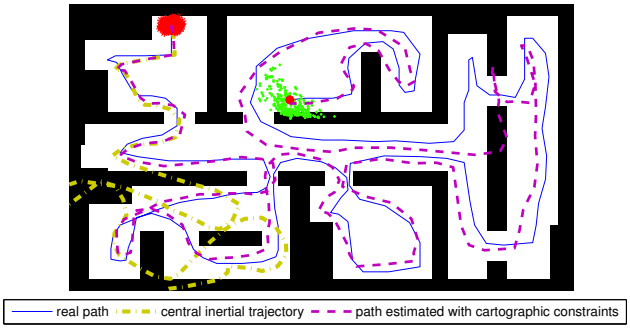


Figure 6: Simulation of trajectory using inertial sensors measurements and cartographic information

w_k^{walk} and w_k^{turn} in the PNS measurements model are independent Gaussian random vectors, with zero mean and standard deviation $0.13\,m$ and $15°$, respectively. A more realistic heading model would incorporate a constant drift b_k of $10°$ per minute and an additive Gaussian noise w_k^{turn}, with zero mean and standard deviation $0.1°$, using the notations of Section 2. The correspondance between lengths in the physical world and the map pixel is $1\,m$ for 3.6 pixel. The map is 940 pixels by 1220 pixels, which corresponds in the physical world to a size of about $250\,m$ by $339\,m$. Once the true trajectory is build, the user begins to walk along the given path with a constant time step $\Delta = 1\,s$ which is also the time interval between two filtering iterations. At the beginning of each iteration, new PNS measurements are generated and the ranging beacon measurements are generated if a user is detected within the beacon

range, which is set as $15\,m$. The additive noise v_k^{range} in the range measurement model is a Gaussian random variable, with zero mean and standard deviation $2.5\,m$.

The main cycle of the filtering algorithm is shown in Figure 7:

1. **Initialization**: particles are initialized randomly around the true position and orientation.
2. **Prediction**: new particles are computed, combining PNS measurements and cartographic constraints.
3. **Weigths update**: only if the ranging measurements are available, weights are updated and the loss detection is performed.
4. **Effective sample size**: weights are normalized and the effective sample size is computed.
5. **Resampling**: only if the effective sample size is smaller than the threshold, the particle population is resampled and in particular the population size is reset to its desired value.
6. **Particles injection**: only if loss is detected, particles are injected around the active ranging beacons, assuming there is one at least.

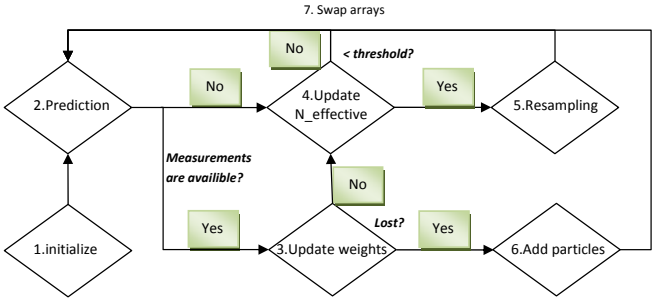


Figure 7: Diagram of simulation program

Figure 8 gives a simulation example. The solid line is the true trajectory and the dashed line is the estimated trajectory. When the user meets a ranging beacon (green triangle), it becomes red and the measured distance between the user and the beacon is drawn as a circle with the appropriate radius. Note that the estimated trajectory in the space where there are many obstacles or walls and more ranging beacons is very close to the true trajectory, whereas the situation is the opposite in the open space. However, with the active ranging beacon located near the exit of the open space, the estimated trajectory returns close to the true trajectory, since only the particles that are close to the true state are kept after the resampling step and for the next filtering iteration.

Figure 8: Tracking simulator using inertial sensors measurements, cartographic information and ranging beacons measurements

6 Future work

This paper describes a data fusion algorithm to localize one or several independent pedestrian users, using PNS measurements, noisy measurements of the distance to beacon, and cartographic information. An additional source of information, that will be considered in future work, is noisy measurements of the distance to another user, considered as a moving beacon with unprecise location [6]. It is also planned to consider a more realistic error model for the PNS estimate of the direction change, such as

$$\alpha_k = \widehat{\alpha}_k + b_k + w_k^{\text{turn}} \ ,$$

with additive error w_k^{turn}, and with a bias b_k that could be modelled either as a constant or as a Gauss–Markov random sequence, in which case it should be incorporated into the state vector. Indeed, if

$$b_k = c_k \, b_{k-1} + w_k^{\text{bias}} \ ,$$

with random input w_k^{bias}, then it is possible to consider an extended state vector $x_k = (r_k, \theta_k, b_k)$, with the following model

$$\begin{aligned} r_k &= r_{k-1} + (\widehat{d}_k + w_k^{\text{walk}}) \, u(\theta_k) \ , \\ \theta_k &= \theta_{k-1} + \widehat{\alpha}_k + b_k + w_k^{\text{turn}} \ , \\ b_k &= c_k \, b_{k-1} + w_k^{\text{bias}} \ , \end{aligned}$$

which generalizes (1).

Acknowledgement

The authors gratefully acknowledge Damien Kubrak and Yann Oster for their careful reading of an earlier version of the paper, and for providing useful comments and input. In addition, they warmly thank Serge Guelton for providing help and advice on software and programming issues.

References

[1] M. Sanjeev Arulampalam, Simon Maskell, Neil J. Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear / non–Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, SP–50(2 (Special issue on Monte Carlo Methods for Statistical Signal Processing)):174–188, February 2002.

[2] Olivier Cappé, Simon J. Godsill, and Éric Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, May 2007.

[3] Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer–Verlag, New York, 2001.

[4] Arnaud Doucet, Simon J. Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, July 2000.

[5] Frédéric Evennou, François Marx, and Emil Novakov. Map–aided indoor mobile positioning system using particle filter. In *Proceedings of the Wireless Communications and Networking Conference, New Orleans 2005*, volume 4, pages 2490–2494. IEEE–CS, March 2005.

[6] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi–robot localization. *Autonomous Robots*, 8(3 (Special issue on Heterogeneous Multi–Robot Systems)):325–344, June 2000.

[7] Dieter Fox, Jeffery Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, July/September 2003.

[8] Lin Liao, Dieter Fox, Jeffrey Hightower, Henry Kautz, and Dirk Schulz. Voronoi tracking : Location estimation using sparse and noisy sensor data. In *Proceedings of the IEEE / RSJ International Conference on Intelligent Robots and Systems, Las Vegas 2003*, pages 723–728. IEEE / RSJ, October 2003.

[9] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. The MIT Press, Cambridge, MA, 2005.