Robotica (2016) volume 34, pp. 837–858. © Cambridge University Press 2014
doi:10.1017/S0263574714001908

WiFi SLAM algorithms: an experimental comparison

F. Herranz†*, A. Llamazares†, E. Molinos†, M. Ocaña† and M. A. Sotelo‡

†*Department of Electronics, Polytechnic School, University Campus, University of Alcalá, 28871 Alcalá de Henares, Madrid, Spain*
‡*Computer Engineering Department, Polytechnic School, University Campus, University of Alcalá, 28871 Alcalá de Henares, Madrid, Spain*

(Accepted June 19, 2014. First published online: July 18, 2014)

SUMMARY
Localization and mapping in indoor environments, such as airports and hospitals, are key tasks for almost every robotic platform. Some researchers suggest the use of Range-Only (RO) sensors based on WiFi (Wireless Fidelity) technology with SLAM (Simultaneous Localization And Mapping) techniques to solve both problems. The current state of the art in RO SLAM is mainly focused on the filtering approach, while the study of smoothing approaches with RO sensors is quite incomplete. This paper presents a comparison between filtering algorithms, such as EKF and FastSLAM, and a smoothing algorithm, the SAM (Smoothing And Mapping). Experimental results are obtained in indoor environments using WiFi sensors. The results demonstrate the feasibility of the smoothing approach using WiFi sensors in an indoor environment.

KEYWORDS: WiFi; SLAM; SAM; Indoor.

1. Introduction

Wireless networks, specifically WiFi (Wireless Fidelity), have become widespread during the last years, especially in indoor environments. This makes WiFi a good choice to develop SLAM (Simultaneous Localization And Mapping) systems in indoor environments where GPS (Global Positioning System) does not provide good results. In this work the SLAM systems are developed using only range measurements. Range-Only (RO) sensors, such as WiFi sensors, provide the range measurements and a unique identifier per beacon, which helps the SLAM avoid the data association problem. However, information about the angle is not obtained. So, with only one range sample it is impossible to estimate a position. The beacon could be anywhere within a ring of radius equal to the range measurement.

Typically, WiFi localization systems use different map representations of the environment. Most of them[1,2] use fingerprint maps based on the radiological patterns of the signal. In contrast, other authors[3] use maps containing the coordinates of all the beacons. In both cases, building the maps is an expensive task; therefore, recent studies focus their efforts on SLAM techniques.

SLAM is considered to be a complex problem because a robot simultaneously needs a consistent map to localize itself, and an accurate estimate of its location to build the map. During the last years a number of solutions have been presented to solve the SLAM problem such as EKF,[4] FastSLAM,[5] GraphSLAM,[6] and Smoothing approaches.[7] Although, SLAM is a well-studied problem with other sensors, the state of the art in RO SLAM is not fully explored. One of the most relevant works in RO SLAM is the work of the authors in ref. [8] which combines EKF with a Relative Over Parameterized (ROP) approach to solve the SLAM problem in outdoor environments. The use of that parameterization derives from the polar coordinate system where annuli, crescents, and other ring-like

* Corresponding author. E-mail: fernando.herranz@depeca.uah.es

shapes can be easily modeled. Furthermore, the authors have made public the dataset described in ref. [9].

This paper proposes the use of the SAM (Smoothing And Mapping) approach to solve the RO SLAM problem. In order to improve the state of the art, it compares the SAM with two RO SLAM algorithms, the EKF, and the FastSLAM. This paper describes the theoretical background of all the methods and their application with RO sensors. The theoretical justification is supported by an experimental comparison of their performance based on the path and map error, complexity, and consistency analysis. The algorithms are tested on two real databases: a publicly available database that consists of ranging radios in various outdoor environments using two robot platforms, and the author's own database. The latter database represents one of the contributions of this paper, consisting of two different indoor scenarios that use WiFi and the Seekur Jr. robotic platform. Along with the database, a generic WiFi observation model is proposed to obtain more accurate results.

The following sections of this paper are organized as follows: Section 2 presents the related work; Section 3 briefly describes the WiFi observation model; Section 4 studies the algorithms that are compared in this work; Section 5 shows the datasets that have been used and the experimental results; and finally, in Section 6 some conclusions and future work are presented.

2. Related Work

Studies in RO SLAM have identified two main problems to deal with. The first one is overcoming the lack of angle information while the second one relates to the way in which the signal propagation is modeled. Both problems are particularly significant in indoor environments, where the environment can change dynamically and the signal can be affected by several interferences like multipath effect.

In order to overcome the lack of angle information, some works[10] use a two-dimensional (2D) probability grid and a voting system to provide estimates of the beacon location. Other solutions[11] propose the use of FastSLAM to estimate the initial position of the beacon. They also replace the conventional EKFs with particle filters to avoid the need of angle measurements. This idea is extended in ref. [12] maintaining independent Sum of Gaussians (SOGs) for each beacon.

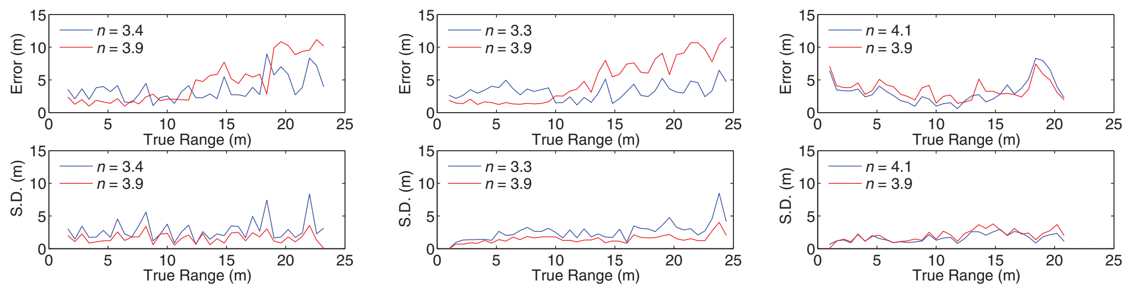
Other works are focused on modeling the signal propagation, which is specially complex in the presence of obstacles such as buildings, walls, or people. In ref. [13] Ferris uses GPLVM (Gaussian Process Latent Variable Model) to generate a likelihood model for signal strength measurements. This model in combination with an appropriate motion dynamics model can be used to reconstruct a topological connectivity graph, performing efficient localization. However, it requires a large number of beacons to obtain good results. In ref. [14] the authors show how a wireless signal strength SLAM can be formulated as a GraphSLAM problem, modeling the signal propagation by means of Gaussian interpolation weights to interpolate WiFi signal strengths. GraphSLAM is a commonly used technique in the robotics community for simultaneously estimating a trajectory and building a map. It shares many benefits of the Gaussian processes but can be applied to a broader range of environments, therefore improving the runtime complexity from $O(N^3)$ to $O(N^2)$, where N is the dimensionality of the state space.

3. Observation Model

In order to achieve a safe and efficient navigation, autonomous systems have to acquire knowledge about their environment. This task is accomplished by sensors, which provide a wide range of measurements during the perception stage. Observation models manage the sensor's perception and extract meaningful information from it. If every parameter that affects the sensor measurements is known, these models can be precisely adjusted. However, the adjustment of the model is not an easy task because the more information the model manages the more complex the design will be.

Generally, signal propagation models attempt to calculate the signal path loss due to propagation. Theoretical and empirical models are used to translate the difference between the transmitted and the received signal strength into a range estimate. Propagation models can be precisely adjusted if all the parameters of the beacons and the receiver that influence the signal are known.

However, the number of different types of beacons in indoor environments is quite large which makes it impracticable to adjust a model per beacon. Kotanen in ref. [15] proposed the Hata-Okumura model (Eq. (1)) which has become popular in the last decade for being suitable in indoor environments



(a) Hall with LOS (b) Corridor with LOS (c) Corridor with NLOS

Fig. 1. Observation model results.

with WiFi technology. This model computes distance d as follows:

$$\log d = \frac{1}{10n}(P_{TX} - P_{RX} + G_{TX} + G_{RX} - X_\alpha + 20\log\lambda - 20\log(4\pi)), \tag{1}$$

where $d(m)$ is the distance between a transmitter and a receiver. P_{TX} (dBm) is the transmitted power and P_{RX} (dBm) is the power level measured by a receiver. G_{TX} (dBi) and G_{RX} (dBi) are the antennae gains of a transmitter and a receiver respectively. Wavelength is λ (m), and n denotes the influence of walls and other obstacles. Since X_α is a normal random variable with standard deviation α, error is also included in the equation.

The Hata-Okumura parameters are conditioned by the physical characteristics of a beacon and a receiver. In ref. [16], we have obtained a general configuration of the model that works in both LOS (Line of Sight) and NLOS (Non Line of Sight) conditions. The transmitted power is $P_{TX} = 17$ dBm, which is half the maximum allowed by the regulations. The antenna gain is fixed to $G_{TX} = 2$ dBi, which is also an allowed value according to P_{TX}. The receiver determines the antenna gain and it is set to $G_{RX} = 5$ dBi which is a common value for laptops. The standard 802.11 establishes the wavelength of WiFi which is $\lambda = 12.5$ cm since WiFi frequency is 2.4 GHz. The standard deviation of X_α is set to $\alpha = 3.22$ as demonstrated in ref. [16]. The influence of walls is modeled with the n parameter which also determines the quantification step. It is set to $n = 3.9$, which is a reliable value to work in LOS as well as in NLOS conditions.

Figure 1 presents a comparison of the different configurations of the n parameter. It shows the results of the propagation model using the best possible configuration of n (blue line) for each testbed and a generic configuration (red line) for all of them ($n = 3.9$). The results are compared by means of the error and the standard deviation of the samples. Figure 1(a) shows the results of a free space hall with LOS ($n = 3.4$). Figure 1(b) shows the results of the corridor with LOS ($n = 3.3$). Figures 1(c) shows the results that were obtained in a corridor with NLOS ($n = 4.1$).

It can be seen from Fig. 1 that the propagation model fits the signal propagation well when working with the optimum values of the n parameter. The error and the standard deviation are less than 5 m in almost every case for the optimum configuration. The generic configuration is better suited for short distances than the optimum configuration proposed but it has a worse performance for distances above 10 meters. Hence, the error increases when applying the generic configuration which is especially noticeable for the LOS cases (Figs. 1(a) and 1(b)). This effect is a result of the n parameter increasing its value from the optimum value to the generic one. The standard deviation of the generic configuration decreases for the LOS cases in comparison to the optimum configurations. In addition, it is worth noting that the standard deviation of the NLOS case does not suffer any important variation when comparing both n configurations. It is important that the standard deviation of the generic configuration remains constant because that error can be managed by the SLAM algorithms. In the presence of occlusions, Fig. 1(c) shows a significant portion of the range data (from 15 to 20 m) subject to additional "unmodeled" noise. This additional noise causes a peak in the error due to an unknown source of noise which suggests the following premise: the longer the signal travels through the environment the higher the probability to be affected by any source of noise.

4. Simultaneous Localization And Mapping with RO Sensors

SLAM is defined as the problem of building a map of the environment while simultaneously localizing a user relative to the map. SLAM binds both problems in a loop and therefore supports the continuity of both aspects in separated processes. In addition, it exploits the iterative feedback from one process to the other and enhances the results of both consecutive steps. Hence, SLAM is an extension of localization and mapping problems although it is significantly more difficult than either of them.

In the last decade, SLAM methods have been a important research area for the robotics community so there is a big diversity of SLAM alternatives. This work aims to compare some of the most popular approaches to SLAM in order to decide which one adjusts better to RO sensor characteristics. It presents SLAM techniques based on filtering and smoothing methods and compares their performance.

4.1. EKF SLAM

The SLAM problem with RO sensors can be approached using the extended Kalman filter. The EKF algorithm solves the online SLAM problem using maximum likelihood.[17] To do so, it represents beliefs $bel(x_t)$ by their first and second moments, the mean μ_t and covariance Σ_t. The map representation is feature-based, which is composed of point landmarks. In RO sensor they are also known as beacons. EKF SLAM assumes *Gaussian noise* for robot motion and perception, and linearizes the motion and measurement models to approximate them. The amount of uncertainty then must be relatively small, since otherwise the linearization in EKF tends to introduce intolerable errors.

SLAM with RO sensors is approached like a known correspondence problem since RO sensors provide a unique identifier per beacon. The EKF SLAM algorithm estimates the robot's pose and the coordinates of all the beacons. Both elements compose the state vector the size of which is mainly defined by the number of beacons. EKF SLAM with other sensors such as cameras or lasers usually presents some computational problems due to the large number of features, as is studied in Section 5.5. However, the number of RO beacons is relatively small which makes EKF suitable for RO SLAM. The state vector is then denoted

$$y_t = \begin{pmatrix} x_t \\ m \end{pmatrix} = (x_t, y_t, \theta_t, m_{1,x}, m_{1,y}, \ldots, m_{N,x}, m_{N,y},)^T, \tag{2}$$

where $x_t = (x, y, \theta)^T$ denotes the robot's coordinates at time t and $m = (m_{b,x}, m_{b,y})^T$ is the coordinates of the bth beacon, for $b = 1, \ldots, B$. The dimension of the state vector is $2B + 3$, where B denotes the number of beacons in the map. EKF SLAM calculates the online posterior $p(x_t, m|z_{1:t}, u_{1:t})$.

The EKF algorithm constructs the belief $bel(x_t)$ recursively from the belief $bel(x_{t-1})$ one time earlier. After each control input, the state vector is modified according to the motion model during the *prediction step*. The control input is defined as $u_t = (\Delta d_t, \Delta\theta)^T$, where Δd_t is the distance traveled between time intervals and $\Delta\theta_t$ is the incremental orientation change. The motion model is denoted by

$$y_{k+1} = \begin{bmatrix} x_k + \Delta d_k \cos\theta_k \\ y_k + \Delta d_k \sin\theta_k \\ \theta_k + \Delta\theta_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{3}$$

The Jacobians that are needed to linearize the motion model are the system matrix F_k and the input gain matrix G_k.

$$F_k = \begin{bmatrix} 1 & 0 & -\Delta d_k \sin\theta_k & 0 & 0 \\ 0 & 1 & \Delta d_k \cos\theta_k & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

$$G_k = \begin{bmatrix} \cos\theta_k & 0 \\ \sin\theta_k & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}. \tag{5}$$

After obtaining information from the sensors, EKF updates the estimation performing the *update step*. The EKF SLAM algorithm requires a linearized *measurement model* with additive Gaussian noise (η_k). The measurement model of RO sensors corresponds to the Euclidean distance between the robot and the beacon and it is denoted by

$$h(x_k, j, m) = \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} + \eta_k. \tag{6}$$

The Jacobian of h, H_k, with respect to the robot's location (the predicted mean $\hat{\mu}_k^-$) and the map (m_j) needs to be computed in order to linearize the measurement model:

$$H_k = \begin{bmatrix} \frac{\partial h}{\partial x_k} & \frac{\partial h}{\partial y_k} & \frac{\partial h}{\partial \theta_k} & \frac{\partial h}{\partial m_{1,x}} & \frac{\partial h}{\partial m_{1,y}} & \cdots & \frac{\partial h}{\partial m_{N,x}} & \frac{\partial h}{\partial m_{N,y}} \end{bmatrix}, \tag{7}$$

where

$$\frac{\partial h}{\partial x_k} = \frac{(x_k - m_{j,x})}{r_j} \tag{8}$$

$$\frac{\partial h}{\partial y_k} = \frac{(y_k - m_{j,y})}{r_j} \tag{9}$$

$$\frac{\partial h}{\partial \theta_k} = 0 \tag{10}$$

$$\frac{\partial h}{\partial m_{1,x}} = \frac{\partial h}{\partial m_{1,y}} = \frac{\partial h}{\partial m_{N,x}} = \frac{\partial h}{\partial m_{N,y}} = 0 \tag{11}$$

$$\frac{\partial h}{\partial m_{j,x}} = \frac{-(x_k - m_{j,x})}{r_j} \tag{12}$$

$$\frac{\partial h}{\partial m_{j,y}} = \frac{-(y_k - m_{j,y})}{r_j} \tag{13}$$

$$r_j = \sqrt{(x_k - m_{j,x})^2 + (y_k - m_{j,y})^2}. \tag{14}$$

It can be noticed from Eq. (10) that the partial derivative is zero because the distance between the robot and the beacon does not depend on the robot's orientation. The partial derivatives described in Eq. (11) are also zero because the distance between the robot and a beacon (Eq. (6)) only depends on the robots coordinates and the coordinates of the beacon itself (m_j).

As mentioned before, the map that EKF uses is a feature-based map which means that it requires an estimate of the beacon's coordinates. RO sensors only provide information of the distance between the beacon and the robot which makes it impossible to provide an $(m_{j,x}, m_{j,y})$ estimation with only one observation. An approach to overcome this problem is the one presented in ref. [18] which uses a voting scheme over a 2D grid to estimate the beacon's position. However, this work aims to use particle filters to estimate a first initial position of the beacon since they use a continuous space representation. The particle filter is denoted as pf_b where b denotes the beacon.

Algorithm 1 describes the EKF algorithm. It shows the modifications made to the standard EKF to work with RO sensors.

Algorithm 1 EKF SLAM with RO sensors

```
Require: Sequence of measurements $z_{1:k}$, command controls $u_{1:k}$
Ensure: Posterior $P(x_k, m|z_{1:k}, u_{1:k})$ represented by the mean $\mu_k$ and covariance $\Sigma_k$.
 1: for $k \leftarrow 1$ to steps do
                                                                      ▷ prediction step
 2:     $u_k, Q_k = get\_odometry$
 3:     $\hat{\mu}_k^-, F_k = prediction(\hat{\mu}_{k-1}, u_k, Q_k)$
 4:     $\Sigma_k^- = F_k \Sigma_{k-1} F_k^T + G_k Q_k G_k^T$

 5:     $z_k, R_k = get\_observations$
                                                                      ▷ Beacon initialization
 6:     if new $b$ then
 7:         Initialize $pf_b$ beacon with $\hat{\mu}_k^-, z_k$
 8:     else
 9:         if $pf_b$ has converged then
10:             $\hat{\mu}_k, \Sigma_k^- = add\_beacon(\hat{\mu}_k, \Sigma_k^-, pf_b)$
11:         else
12:             Update $pf_b$ beacon with $\hat{\mu}_k^-, z_k$
13:         end if
14:     end if
                                                                      ▷ update step
15:     $S_k = H_k \Sigma_k^- H_k^T + R_k$
16:     $K_k = \Sigma_k^- H_k^T S_k^{-1}$
17:     $\Sigma_k = (I - K_k H_k)\Sigma_k^-$
18:     $v_k = z_k - h(\hat{\mu}_k^-)$
19:     $\hat{\mu}_k = \hat{\mu}_k^- + K_k v_k$
20: end for
```

In detail, the EKF algorithm works as follows:

1. Prediction step (lines 2–4). The EKF algorithm uses the motion model defined in Eq. (3) to propagate the state of the robot $(\hat{\mu}_k)$ and the covariance matrix (Σ_k) while maintaining the uncertainty about the current state.
2. Observation step (line 5). The robot receives measurements from the beacons that are observable. In addition, it provides information about the covariance R_k of the measurement noise.
3. Beacon initialization (lines 6–13). The EKF algorithm needs to know an initial beacon hypothesis. Here, a mapping algorithm based on particle filter is used to provide an initial estimation of the beacon's position. The particle filter is described in our previous work.[19] The beacon's location is added to the EKF when the particle filter has converged. The convergence is determined by a combination of two analyses: first, it is verified that the particle filter is only following one hypothesis by means of clustering; second, the Mardia multivariate normality test using skewness and kurtosis[20] is used to determine if the null hypothesis of multivariate normality is a reasonable assumption regarding the particle population. Once that convergence has been determined, the state vector $(\hat{\mu}_k)$ and the covariance matrix (Σ_k^-) are extended to allocate the beacon information. In case the particle filter does not converge, the beacon is not added to the EKF because its location is not accurate enough to help the EKF.
4. Update step (lines 14–18). This step executes the standard equations of EKF to correct the estimation with the sensor information. Line 14 computes the innovation covariance matrix S_k. Line 15 computes the Kalman gain K_k. Line 16 updates the covariance matrix Σ_k based on the

Kalman gain, the linearization of the measurement model, and the estimated covariance matrix. Line 17 computes the innovation (i.e. the difference between the measurement and its expected value according to the model). Finally, Line 18 updates the state estimate using the Kalman gain and the innovation.

4.2. FastSLAM

The FastSLAM[21] is an alternative approach to SLAM that is based on particle filters. It is an efficient SLAM algorithm which decomposes the SLAM problem into a robot localization problem, and a collection of beacon estimation problems that are conditioned by the robot pose estimate.

The FastSLAM presents some advantages with respect to the EKF to solve the SLAM problem. In SLAM, each control or observation collected by the robot only constrains a small number of state variables. Controls constrain the pose of the robot relative to its previous pose, while observations constrain the position of the beacons relative to the robot. It is only after a large number of these probabilistic constraints are incorporated that the map becomes fully correlated. The EKF, which makes no assumptions about structure in the state variables, fails to take advantage of this sparsity over time. The FastSLAM exploits conditional independences that are a consequence of the sparse structure of the SLAM problem to factor the posterior into a product of low dimensional estimation problems. The FastSLAM samples over potential robot paths, instead of maintaining a parameterized distribution of solutions like the EKF. Hence, the FastSLAM maintains multiple tracking hypotheses making it possible to solve the SLAM problem even with consistently high measurement ambiguity. Finally, since the FastSLAM uses particle filters, it is able to solve the *full SLAM problem* and the *online SLAM problem.*

Particles in the set of particles are used for estimating the robot path. The FastSLAM makes every particle estimate and individual map which errors are conditionally independent. Hence, the mapping problem is factored into many separated problems, one for each beacon in the map. The standard FastSLAM algorithm estimates these beacon locations by means of EKFs, but using a separate low-dimensional EKF for each individual beacon. This approach is completely opposite to the EKF one which uses a single filter to estimate the location of all beacons jointly. FastSLAM recursively estimates the full posterior distribution over robot pose and beacon locations (Eq. (15)).

$$p(x_t, m|u_t, z_t) = p(x_t|z_t, u_t)\prod^{N} p(m|x_t, z_k). \tag{15}$$

The FastSLAM maintains a set S of N particles. Particles in the standard FastSLAM algorithm contain an estimated robot pose (x_t^j) and a Kalman filter with mean $\mu_{b,t}^j$ and covariance $\Sigma_{b,t}^j$ per beacon in the map. Here index j is the index of the particle and b is the index of the beacon. The number of filters is "$NB + 1$" where B is the number of beacons in the map. The standard FastSLAM algorithm works based on the following steps:

- Repeat N times (one per particle):
  1. Retrieval. Retrieve pose x_{t-1}^j from particle set S_{t-1}.
  2. Prediction. Sample a new pose $x_t^j \sim P(x_t|u_t, x_{t-1}^j)$.
  3. Update. For each observed beacon b, incorporate the measurement $z_{b,t}$ into the corresponding EKF by updating $\mu_{b,t}^j$ and covariance $\Sigma_{b,t}^j$.
  4. Importance weight. Calculate the importance weight for the particle.
- Resampling. Sample, with replacement, N particles.

The initial step involves the sampling of a robot pose for time t using the motion model. The update step updates the EKFs for the beacons that have been observed, using the standard EKF update equations (Section 4.1). The FastSLAM takes advantage of this step to learn the map. Finally, the importance weight step computes the weight of the particles, which are used in the resampling step.

The standard FastSLAM algorithm is not suitable to directly work with RO sensors due to the fact that the EKFs have to be initialized with a precise estimate of the beacons. Since RO sensors do not provide any angle information, it is not possible to obtain that precise estimate. This work aims for replacing the EKFs with particle filters as suggested in ref. [11]. The particle filter is denoted as pf_b^j where j denotes the ith particle and b the beacon associated with the filter. It seems counter-intuitive at first due to the fact that the complexity of the algorithm increases. However, this decision is based on the assumption that the number of beacons is not very large (about 20 beacons in the worst case scenario) and computers are able to manage "$NB + 1$" particles filters at the same time. The choice of using particle filters instead of EKFs presents another important advantage, FastSLAM can add the beacons to the SLAM problem the first time they are observed. This fact is independent of whether the beacon locations are sufficiently precise or the map already contains well-localized beacon.

Blanco in ref. [11] suggested to replace the particle filter pf_b^i with a standard EKF once it has converged. However, this work contemplates the use of WiFi technology, which justifies to maintain the particle filters and not to replace them with EKFs since particle filters manage better the uncertainty of WiFi technology.

Algorithm 2 describes the FastSLAM algorithm. Suppose S is a set of particles that represents the posterior potential paths of the robot and S' is the set of particles that represents the posterior potential positions of each beacon. For simplicity, the algorithm assumes that only a single beacon is measured at each point in time.

Algorithm 2 FastSLAM algorithm with RO sensors

```
Require: Sequence of measurements of the beacons $z_{1:k}$ and movements $u_{1:k}$ and set of $N$ particles $S$ for the
    main filter and and a set of $N'$ particles $S'$ for the particle filter of each beacon.
Ensure: Posterior $P(x_{1:k}, m|u_{1:k}, z_{1:k})$ represented by M about the path of the robot at time $k$ and the map.
 1: for $j \leftarrow 1$ to $N$ do
 2:     $x_i^j \leftarrow (0, 0, 0)$
 3: end for
 4: for $k \leftarrow 1$ to steps do
                                                        ▷ prediction step
 5:     for $j \leftarrow 1$ to $N$ do
 6:         compute a new state $x$ by sampling according to $P(x|u_k, x_{k-1}^j)$.
 7:         $x_k^j \leftarrow x$
 8:     end for
 9:     $\eta \leftarrow 0$
10:     $z_k, R_k = get\_observations$
                                                        ▷ update step
11:     for $j \leftarrow 1$ to $N$ do                  ▷ beacon estimation
12:         if beacon $b$ never seen before then
13:             Initialize $pf_b^j$ beacon with $x_k^j, z_k$ (based on[19])
14:         else[updating the beacon]
15:             Update $pf_b^j$ beacon with $x_k^j, z_k$ (based on[19])
16:         end if                                      ▷ FastSLAM update
17:         $w_k^j = P(z_k|x_k^j, m_{1:k-1}^j)$
18:         $\eta = \eta + w_k^j$
19:     end for
20:     for $j \leftarrow 1$ to $N$ do
21:         $w_k^j = \eta^{-1} \cdot w_k^j$
22:     end for
23:     $S = resample(S)$
24: end for
```

In detail, the FastSLAM algorithm works as follows:

1. Initialization (lines 1–3). The algorithm initializes all the particles at $(0, 0, 0)^T$ because the algorithm references the map and the robot to the origin of coordinates.
2. Prediction step (lines 5–8). Whenever an control input u_k is obtained, the FastSLAM computes a new state x_k for each particle in S_k by sampling according to $P(x|u_k, x_{k-1}^j)$.
3. Line 9 assign the value zero to η. This parameter is later used as a normalization factor of the weight.

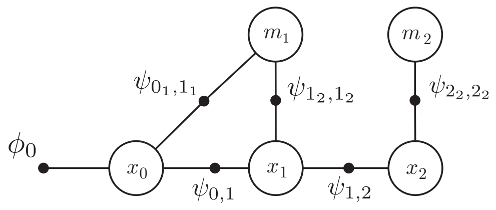


Fig. 2. Factor graph for beacon-based SLAM.

4. Update step (lines 11–22). The FastSLAM updates the posterior over the beacon feature estimates which are represented by the particle filter estimation. This process is subdivided in the following steps:
   (a) Beacon initialization (line 13). When a new beacon is observed, the FastSLAM adds the beacon in the SLAM problem. This step is performed the first time that the beacon is observed without waiting for a precise beacon location. To do so, every particle sets up a particle filter pf_b^j and associates it to beacon b. The initialization process of pf_b^j is the same as the one described in our previous work.[19]
   (b) Beacon update (line 15). The FastSLAM updates pf_b^j using pose x_k^j and observation z_k as the authors of ref. [19] describe. The FastSLAM takes advantage of conditional independence and only updates the particle filters related with beacon b.
   (c) FastSLAM update (lines 17 and 18). The FastSLAM applies the measurement step and calculates the *importance factor* w_k^j based on $P(z_i|x_k^j, m_{1:k-1}^j)$ for each particle x_k^j. Line 17 adds up weight w_k^j to factor η.
   (d) Normalization (line 21). The weight of the particle is normalized by factor η which contains the sum of all the weights.
5. Resampling (line 23). The FastSLAM resamples, with replacement, the set S of particles.

4.3. SAM

SAM[7] is presented as a viable alternative to filtering approaches, specifically EKF-based solutions, to the SLAM problem. Filtering models the problem as an online state estimation where the state of the system consists in the current robot position and the map. The estimate is augmented and refined by incorporating the new measurements as they become available. In contrast, smoothing estimates the full trajectory of the robot and the map from the full set of measurements. SAM proposes a factor graphs representation[22] and relies on least-square error minimization techniques to solve the full SLAM problem. It adds the entire trajectory of the robot and the map in the estimation problem which seems counter-intuitive at first because more variables are added to the problem. However, the simplification arises from the fact that the smoothing information matrix "$\mathcal{I}$" is naturally sparse, which allows SAM to take advantage of the sparsity of $\mathcal{I}$.[23] SAM formulates the corresponding mean as the solution to a set of linear approximations for which $\mathcal{I}$ is the matrix of coefficients. To do so, it decomposes the information matrix into either its Cholesky or QR factorization, paying close attention to variable ordering. Finally, it jointly solves for the mean over the pose and map via back-substitution.

As mentioned before, the SAM represents the SLAM problem using factor graphs. Factor graphs are graphical models that are well suited to modeling complex estimation problems. A factor graph is a bipartite graph containing two types of nodes: variable nodes and factor nodes. The variables represent the unknown random variables in the estimation problem. The factors represent the probabilistic information on those variables, derived from measurements or prior knowledge. Edges are only allowed between different types of nodes, i.e. they always connect a node with a factor. Figure 2 shows an example of a factor graph representing a simple SLAM problem.

It shows the typical connectivity of a factor graph representation: white circles represent variable nodes (poses and beacons) while small solid black circles represent factor nodes (measurements). A beacon-based SLAM problem contains robot poses x_i and beacon locations m_b. Consecutive robot poses (x_i and x_j) are connected by an odometry measurement which is denoted as factor $\psi_{i-1,j}$. The beacons can be observed from different poses, and their observations are also represented by factors

$\psi_{(k_n,b_n)}$. The example shows three pose variables x_1, x_2, and x_3 and two beacon variables m_1 and m_2. It also has a unary factor $\phi_0(x_0)$ on the first pose x_0 that encodes the prior knowledge about x_0 and two binary factors ($\psi_{(0,1)}$, $\psi_{(1,2)}$) that relate successive poses. In addition, the example also contains three binary factors ($\psi_{(0_1,1_1)}$, $\psi_{(1_2,1_2)}$, $\psi_{(2_2,2_2)}$) that encode the measurements between poses and beacons.

The probability model corresponding to factor graphs for beacon-based SLAM is presented in Eq. (16).

$$P(x_{1:k}, m|z_{1:n}, u_{1:k}) = \phi_0(x_0)\prod_{k=1}^{K}\psi_{(k-1,k)}(x_{k-1}, x_k)\prod_{n=1}^{N}\psi_{(k_n,b_n)}(x_{k_n}, m_{b_n}), \tag{16}$$

where the state of the robot at the ith time step is denoted by x_i, with $k = 1\ldots K$, a beacon by m_b with $b = 1\ldots B$, and a measurement by z_n, with $n = 1\ldots N$.

The SAM exploits the factor graph model of full SLAM problem and formulates the state estimation as an optimization problem. The full SLAM problem is derived from the standard probabilistic notation to an optimization problem notation. The optimization problem recovers the *maximum a posteriori* (MAP) estimate for the entire trajectory and the map. It proceeds as follows: solving the full SLAM problem consists of estimating the posterior probability (Eq. (17)) of the robot's trajectory $x_{1:k}$, with $k = 1\ldots K$, and map m given a set of measurements $z_{1:n}$, with $n = 1\ldots N$, and control inputs $u_{1:k}$.

$$\begin{aligned} P(x_{1:k}, m|z_{1:n}, u_{1:k}) &= \prod_{k=1}^{K}\underbrace{P(x_k|x_{k-1}, u_k)}_{\text{motion model}}\ \underbrace{P(z_n|x_k, m)}_{\text{measurement model}} \\ &= \prod_{k=1}^{K}P(x_k|x_{k-1}, u_k)\prod_{n=1}^{N}P(z_n|x_{k_n}, m_{b_n}). \end{aligned} \tag{17}$$

From the point of view of smoothing, the full SLAM problem is solved by maximizing the posterior probability of full SLAM (Eq. (18)). Maximizing $P(x_{1:k}, m|z_{1:n}, u_{1:k})$ is equal to obtaining the MAP estimate $\Theta^* = (x^*_{1:k}, m^*)$ for the trajectory and map by minimizing the negative log of the posterior probability of full SLAM.

$$\begin{aligned} \Theta^* &= \arg\max_{\Theta} P(x_{1:k}, m|z_{1:n}, u_{1:k}) \\ &= \arg\min_{\Theta} -\log P(x_{1:k}, m|z_{1:n}, u_{1:k}) \\ &= \arg\min_{\Theta} -\log\prod_{k=1}^{K}P(x_k|x_{k-1}, u_k)\prod_{n=1}^{N}P(z_n|x_{k_n}, m_{b_n}). \end{aligned} \tag{18}$$

Then, the full SLAM problem can be presented as a non-linear least-squares problem (Eq. (19)) in which the motion model (f_k) and measurement model (h_n) equations are taking into account. Matrices Λ_k and Σ_n denote the covariance matrices associated to the motion and measurement models respectively. If the motion model or the measurement model are non-linear, non-linear optimization techniques such as Gauss–Newton optimization or Levenberg–Marquardt algorithm can solve the linear approximations to Eq. (19) in order to approach the minimum.

$$\Theta^* = \arg\min_{\Theta}\left\{\sum_{k=1}^{K}\|\underbrace{f_k(x_{k-1}, u_k)}_{\text{motion model}} - x_k\|^2_{\Lambda_k} + \sum_{n=1}^{N}\|\underbrace{h_n(x_{k_n}, m_{b_n})}_{\text{measurement model}} - z_n\|^2_{\Sigma_n}\right\}. \tag{19}$$

Since the SLAM problem is represented using factor graphs it is possible to simplify the notation of the optimization problem. From now on, it is used as a generic error term $e_{ij}(x) = e(x_i, x_j, z_{ij})$ to represent odometry and measurement errors. It represents the residual of the constraint connecting variables i and j. The potential of this constraint is defined by the Mahalanobis distance $\|e_{ij}(x)\|^2_{\Omega_{ij}}$

with respect to covariance Ω_{ij}.

$$\Theta^* = \arg\min_{\Theta} \sum_{i,j} \|e_{ij}(x)\|^2_{\Omega_{ij}}. \tag{20}$$

Given the current state estimate x_0, each error term is linearized at x_0 to obtain $e_{ij,0} + J_{ij}\delta$, where $e_{ij,0} = e_{ij}(x_0)$, J_{ij} is the Jacobian of $e_{ij}(x)$ and $\delta = x - x_0$. Equation (20) can be expressed as follows:

$$\begin{aligned}\Theta^* &= \sum_{i,j} \|J_{ij}\delta + e_{ij,0}\|^2_{\Omega_{ij}} \\ &= \sum_{i,j} \|\Omega_{ij}^{\frac{1}{2}} J_{ij}\delta + \Omega_{ij}^{\frac{1}{2}} e_{ij,0}\|^2_2 \\ &= \sum_{i,j} \|A_{ij}\delta + q_{ij}\|^2_2 \\ &= \|A\delta + q\|^2_2,\end{aligned} \tag{21}$$

where $A_{ij} = \Omega_{ij}^{\frac{1}{2}} J_{ij}$, $q_{ij} = \Omega_{ij}^{\frac{1}{2}} e_{ij,0}$. $A \in \mathbb{R}^{m\times n}$ is the sparse measurement Jacobian of constraints and is formed by all A_{ij}. Each row of A corresponds to a constraint, and its only two non-zero entries are at columns corresponding to the two variables involved in this constraint. Similarly, $q \in \mathbb{R}^m$ is formed by all q_{ij}.

Finally, after collecting the Jacobian matrices into matrix A and elements q_{ij} into the right-hand-side (RHS) vector q, the following standard least-squares problem is obtained:

$$\delta^* = \arg\min_{\delta} \|A\delta - q\|^2_2. \tag{22}$$

Such sparse least-squares systems are typically converted into an ordinary linear equation system by setting the derivative of Eq. (22), resulting in equation $A^T A\delta = A^T q$, where the information matrix is denoted as $\mathcal{I} = A^T A$, which is sparse. This equation system can be solved by Cholesky or QR factorization of $A^T A$ which provide the square root information matrix $R \in \mathbb{R}^{n\times n}$.

From a graphical model point of view, factorization is equivalent to variable elimination, in which nodes are eliminated one by one. Eliminating one node introduces edges among all its neighbors. These extra edges correspond to fill-ins in one row of the square root matrix R. Too many fill-ins results in slow factorization, and the order in which variables are eliminated may drastically affect the number of fill-ins. Since finding an optimal ordering is NP-complete, SAM uses COLAMD (Column Approximate Minimum Degree Ordering)[24] to approximate the solution.

Algorithm 3 describes the SAM algorithm. It shows the factor graph generation, the initialization of the first estimation, and the optimization process. For simplicity, the algorithm assumes that only a single beacon is measured at each point in time.

In detail, the SAM algorithm works as follows:

1. Factor graph generation (lines 1–8). The algorithm builds the factor graph based on the odometry and range measurements. To do so, the algorithm proceeds as follows:
   (a) Prior (line 1). The prior knowledge of the robot position is added to the graph. In this case, no prior knowledge is assumed and the prior is set up at the origin of coordinates.
   (b) Odometry factors (lines 3–5). Factors regarding the odometry measurements are added to the graph by identifying which variables relate (key_j and key_k). The odometry measurements are added taking into account their noise covariance Λ_k.
   (c) Range factors (lines 6–7). Range factors are added to the graph when measurements are available. Range measurements are obtained using the observation model described in Section 3. The factor is added relating the odometry variable, in which the range measurement was obtained, and the beacon. The beacon identifier (key_b) is provided by the RO sensors. In addition it is also used the noise covariance Σ_k of the measurement.

Algorithm 3 SAM algorithm with RO sensors

Require: Sequence of measurements of the beacons $z_{1:k}$ and movements $u_{1:k}$.
Ensure: Posterior $P(x_{1:k}, m|u_{1:k}, z_{1:k})$ about the path of the robot and the map.

```
                                                            ▷ create factor graph
                                                            ▷ prior odometry factor
 1: $graph \leftarrow prior\_factor(key_0, origin)_{\Lambda_0}$
 2: for $k \leftarrow 1$ to steps do
                                                            ▷ add odometry factor
 3:     $u_k, \Lambda_k = get\_odometry$
 4:     $key_j = k - 1; \quad key_k = k$
 5:     $graph \leftarrow odom\_factor(key_j, key_k, u_k)_{\Lambda_k}$
                                                            ▷ add range factor
 6:     $key_b, z_k, \Sigma_k = get\_observations$ (Section 3)
 7:     $graph \leftarrow range\_factor(key_k, key_b, z_k)_{\Sigma_k}$
 8: end for
                                                            ▷ set initial variable values
 9: for $k \leftarrow 1$ to steps do
10:     $key_k = k$
11:     $init\_est \leftarrow odom\_pose(key_k, robot\_pose_k)$
12: end for
13: for $b \leftarrow 1$ to $B$ do
14:     $key_b = b$
15:     $init\_est \leftarrow beacon\_point(key_b, robot\_pose_b)$
16: end for
                                                            ▷ optimize
17: Build Jacobian $A$ and RHS $q$
18: Find a good column ordering $p$, and reorder $A_p \xleftarrow{p} A$
19: Solve $\delta_p^* = \arg\min_\delta \|A_p\delta_p - q\|_2^2$
20: Recover the optimal solution $\delta \xleftarrow{p^{-1}} \delta_p$
```

2. Initial estimation (lines 9–16). After building the factor graph, it is needed to set up the nodes.
   (a) Odometry variables (lines 10–11). For every robot pose measurement it is given an initial estimation which relates the variable key_k with the robot pose at time k.
   (b) Beacon variables (lines 14 and 15). This step is one of the key "tricks" of the algorithm. Since an initial estimation of the beacon is needed and RO sensors do not provide it, the algorithm uses the robot position where the beacon was first seen as the estimation. At first glance, it seems to be a mistake but it takes advantage of the optimization process which is executed afterward. The optimization process is able to handle the error in the estimation and find a minimum.
3. Optimization (lines 17–20). Once the factor graph has been built and the initial estimation has been set up, the SLAM problem can be solved as shown in Eq. (19). To do so, Jacobian A and vector q must be built as described in Eq. (21). A good variable ordering p of the Jacobian A must also be defined in order to improve the performance of the optimization problem. Finally the non-linear problem is solved using a QR factorization method and the optimal solution is recovered by undoing the variable ordering.

5. Test-Bed and Results

This section presents the test bed, the metrics that have been used to compute the errors, and the experimental results.

Methods for SLAM can be evaluated by means of a visual inspection when information about the ground truth is not available or is not accurate enough. The visual inspection compares maps or overlays with blueprints of buildings. When the true locations are known methods are usually evaluated by measuring the Euclidean or Mahalanobis distance between the estimated locations and the true ones. Most researchers often validate the algorithms using an absolute reference frame in which the true locations and the estimated ones are attached to the frame and can be compared. A generic metric that is used for measuring the performance of the algorithm in the absolute frame

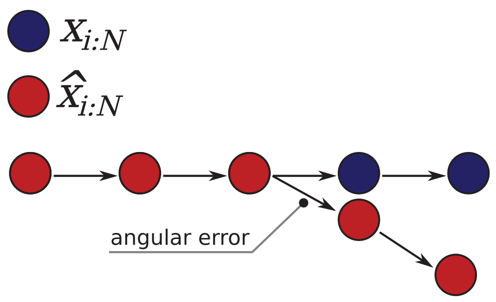


Fig. 3. Absolute reference frame.

(Eq. (23)) is shown below.

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{x}}_i \ominus \mathbf{x}_i)^2, \tag{23}$$

where ε is the mean squared error, $\hat{\mathbf{x}}_i$ is the estimated location, $\mathbf{x}_i$ is the true location, and $\ominus$ is the inverse of the standard motion composition operator $\oplus$ as defined by Lu and Milios.[25] This metric is widely used because it is easy to compute and provides comprehensible information. However, it has some shortcomings when it is applied in an absolute reference frame.[26] Figure 3 illustrates a simple example where the metric in Eq. (23) is suboptimal. It shows the true trajectory of a robot (blue circles) and the estimated trajectory (red circles). The robot moves along a straight line and after some poses, the estimated trajectory makes a small angular error but then continues without any further error. The trajectory is well estimated and only one of the nodes contains a small angular error. According to Eq. (23), the error in these estimates increases with every node after an angular error has occurred although the relative geometric relation between the nodes is perfectly estimated. Thus, the error depends on the point in time where the robot made an estimation error without considering that it might not introduce any ("further") error. The reason for this is that the metric operates on global coordinates and considers the trajectory, and thus the map is a rigid body that has to be aligned with the ground truth.

Burgard in ref. [26] uses a metric that is only relative to the relation between poses and does not rely on a global reference frame (Eq. (24)). This metric considers the energy that is "virtually" needed to deform the trajectory estimated into the ground truth as a quality measure. This can be done by considering the poses as masses and connections between them as springs. Thus, the metric is based on the relative displacement between poses.

$$\varepsilon(\delta) = \frac{1}{N}\sum_{i,j}^{N}(\hat{\delta}_{i,j} \ominus \delta_{i,j})^2 = \frac{1}{N}\sum_{i,j}^{N} trans(\hat{\delta}_{i,j} \ominus \delta_{i,j})^2 + rot(\hat{\delta}_{i,j} \ominus \delta_{i,j})^2, \tag{24}$$

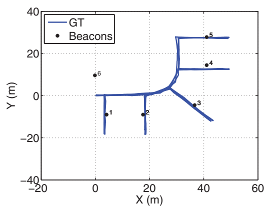
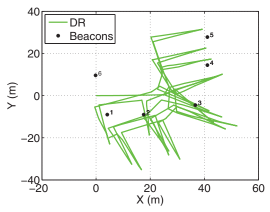
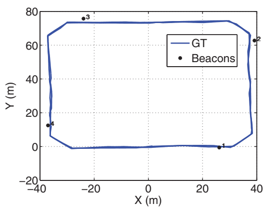
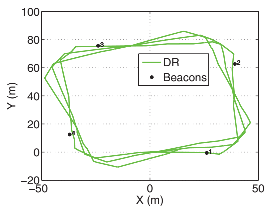
where N is the number of relative relations, $\delta_{i,j}$ is the relative transformation that moves node $\mathbf{x}_i$ onto $\mathbf{x}_j$, $\hat{\delta}_{i,j}$ is the relative transformation based on $\hat{\mathbf{x}}_\mathbf{i}$ and $\hat{\mathbf{x}}_\mathbf{j}$, and $trans(\cdot)$ and $rot(\cdot)$ are used to separate the translational and rotational components. In this case, the error (or "transformation energy") in the above-mentioned example (Fig. 3) will be consistently estimated as a single rotational error no matter where the error occurs in the space or in which order the data are processed.

5.1. Database and ground truth values

Selecting the database that will be used to test an algorithm is a critical decision in Mobile Robotics due to the large number of sensors and methods involved. Because of the characteristics of RO sensors, a quality database must be of sufficient size and be able to represent the variability of different parameters of interest such as signal strength, robot pose, ground truth. Collecting the vast amounts of data required to create such a database is a resource-intensive task.

Given that RO sensors are very specific sensors, there are not very many publicly available databases. Most of the researchers build their own databases and do not publish them. However, they

Table I. Datasets description.

Ground truth path	Dead reckoned path	RO technology	Description
		WiFi-based ranging	UAH1 dataset. Distance = 0.95 km, 4604 RO measurements
		WiFi-based ranging	UAH2 dataset. Distance = 0.9 km, 5081 RO measurements

usually provide the type of information that the databases store. The databases commonly store the robot geometry, raw odometry and ground truth, RO technology, number of beacons, observation model, and raw signal strength measurements. One relevant public database is the one supplied by the *Robotics Institute* of *Carnegie Mellon University*.[9] This work provides an extension to that database by collecting the indoor WiFi database described below.

5.1.1. UAH Database. The Alcalá WiFi database (*http://www.robesafe.com/repository/UAHWiFiDataset/*) has been created at the Polytechnic School of the University of Alcalá by the authors.

The robotic platform Seekur Jr. from MobileRobot has been used to build the database and perform the test. The Seekur Jr. was equipped with a laser (to obtain the ground truth) and an on-board laptop to obtain the RO measurements and process the data. The laptop configuration is the following: Ubuntu 12.04, ROS Fuerte, and Wireless Tools v29. The ground truth path of the robot has been obtained by means of a Gmapping algorithm from.[27] Gmapping solution precision is within a few centimeters which makes Gmapping solution a reliable ground truth for WiFi SLAM methods.

WiFi technology has been used in this database because most of the buildings already provide this network. Since it is pre-installed there is no need to modify the environment and the use of WiFi frequency (2.4 GHz) is free. The database has been built through teleoperation of the robot to explore the environments. The scenarios were defined under real conditions, which means people wandering around, small changes in the environment, small interferences due to portable devices, etc., in indoor environments.

The datasets are separated according to the environment in which they were placed. The UAH1 dataset was collected in the west area of the second floor; the area of this place is 60×60 m. There are four small corridors which are 18-m long and a main corridor of 35-m that finishes in a hall. The hall is a semi-empty space only occupied by two elevators. The UAH2 dataset was collected in the third floor of the building which has an area of 120×120 m. This environment consists of four large corridors forming a square in which a landmark is placed in every corner.

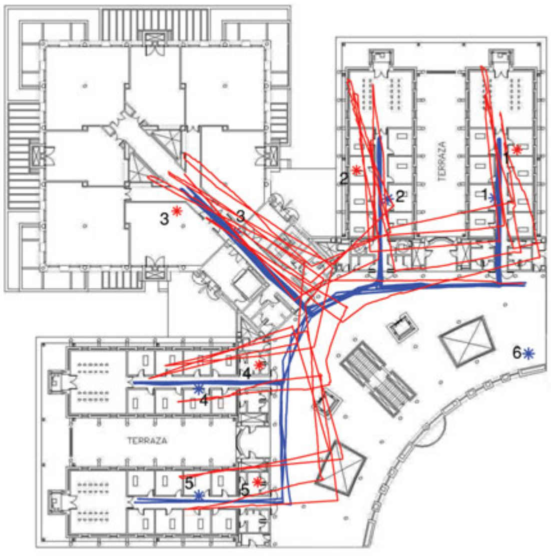
Table I presents the different datasets of the UAH database. It shows the ground truth and the dead reckoned path. It also shows the RO technology that was used and a brief description of the dataset.

5.2. EKF-SLAM results

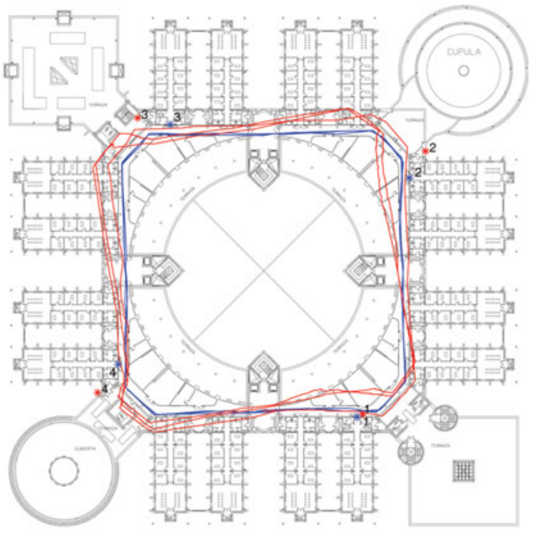
Figure 4 shows the estimated path and map for the UAH database. The ground truth trajectory of the robot is shown in blue; the beacons are presented as blue stars. The estimation of the robot is shown in red while the estimation of the beacons is shown as red stars. Both datasets demonstrate

Table II. EKF SLAM. UAH results.

Error		UAH1	UAH2
Global error	Mean (m)	5.329	5.192
	90th percentile (m)	8.052	8.469
Local translational error	Mean (m)	0.066	0.061
	90th percentile (m)	0.197	0.164
Local rotational error	Mean (rad)	0.076	0.027
	90th percentile (rad)	0.007	0.005
Map	Mean (m)	4.242	5.278



(a) UAH1



(b) UAH2

Fig. 4. EKF SLAM. UAH results.

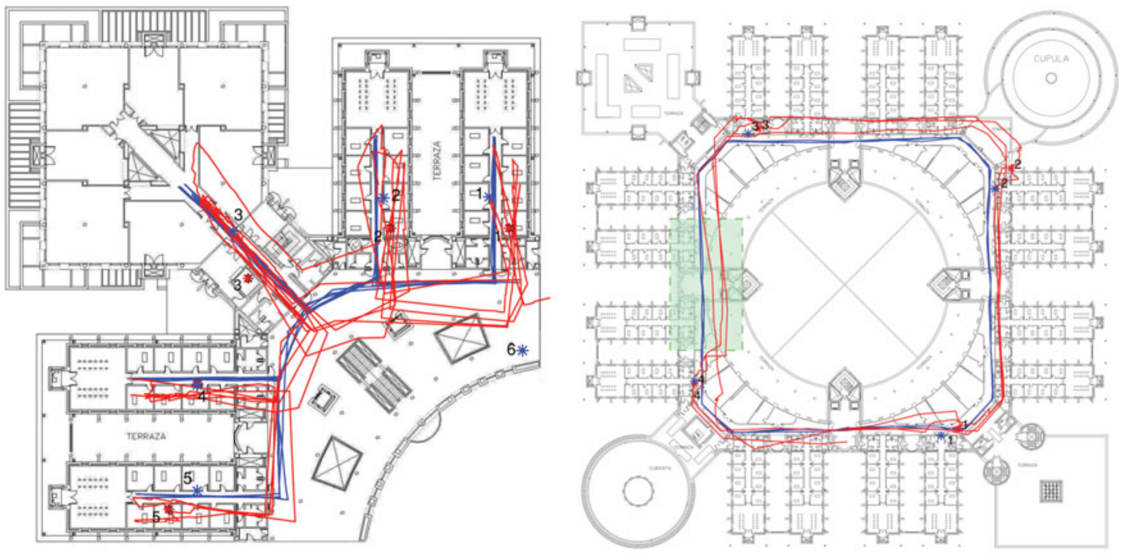
that the EKF is affected by the noise of the WiFi signal. The EKF filters the odometry and corrects it but the EKF does not fit the ground truth because of the high noise in the odometry and range measurements. Figure 4(a) shows that the estimation of the robot pose fails because of the high uncertainty in the odometry measurements. In addition, the noise from WiFi technology does not help the robot estimation because the noise makes it difficult to obtain an accurate map which could help the EKF. Results shown in Fig. 4(b) are similar to those in Fig. 4(a). However, the first beacon is accurately estimated in this experiment because the odometry has not yet accumulated too much rotational error before estimating the beacon location.

Table II shows a metrical evaluation of the results for the UAH database. It presents global, local, and map errors. The EKF obtains poor errors although it is noteworthy that this database is using WiFi technology which is a challenge. The global error is about 5 m and the 90th percentile is about 8 m. 90th percentile means that 90% percent of the samples are bounded in the interval from 0 to 8 m, which is a large interval especially for robotics applications.

The performance analysis of the EKF shows that the EKF presents some problems with high noise measurements which cannot be linearized. Hence, the linearization that the EKF performs does not properly adjust the non-linearities in the motion and measurement model. This is especially important when using WiFi technology because the update stage is not able to correct the errors in the prediction stage due to the uncertainty of WiFi measurements. EKF, then, does not accurately solve the SLAM problem with RO sensors. Hence, it would be useful to use techniques more robust to non-linearities and high uncertainties such as FastSLAM.

Table III. FastSLAM. UAH results.

Error		UAH1	UAH2
Global error	Mean (m)	3.028	4.501
	90th percentile (m)	4.893	8.624
Local translational error	Mean (m)	0.079	0.067
	90th percentile (m)	0.212	0.163
Local rotational error	Mean (rad)	0.086	0.036
	90th percentile(rad)	0.033	0.050
Map	Mean (m)	2.544	3.087



(a) UAH1 (b) UAH2

Fig. 5. FastSLAM. UAH results.

5.3. FastSLAM results

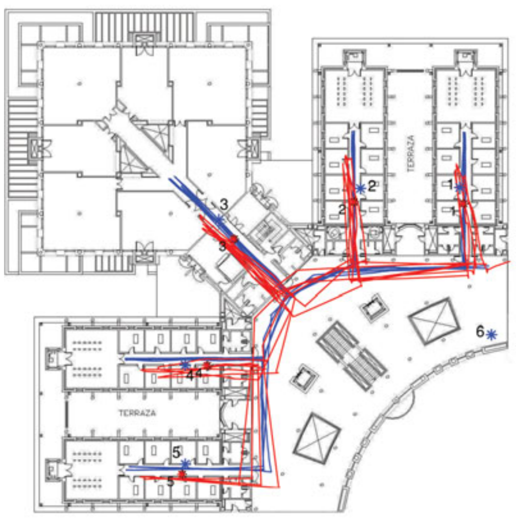
Figure 5 shows the FastSLAM results for the UAH database using the same color representation as Fig. 4. Figure 5 demonstrates that FastSLAM almost aligns with the ground truth trajectory although it fails due to the uncertainty in the WiFi measurements. Hence, it finds it difficult to eliminate hypotheses during the resample stage which makes FastSLAM not converge into the optimal solution. It can also be seen that the beacons are well placed with respect to the robot trajectory.

Table III shows the evaluation of FastSLAM errors. It presents global, local, and map errors. The FastSLAM performs better than the EKF in terms of global error. However, the global error is still significant which is especially notable in the UAH2 dataset (mean error of 4.5 m and 90th percentile of 8.6 m). One of the reasons for the global error in the UAH2 dataset is because the beacons are not observable from some areas in the environment and the FastSLAM algorithm gets lost and diverges. This occurs, for example, near beacon number 4 (green area in Fig. 5(b) in which the FastSLAM is lost but when it observes the beacon it corrects its estimation. Local errors are quite similar to those observed for the EKF. The map error is about 3 m, which is not accurate enough. However, these errors can be considered good for some applications that do not need high accuracy such as human localization.

The performance analysis of the FastSLAM shows that this algorithm solves some of the problems of EKF, especially the small rotation errors that affect the global error. However, the FastSLAM does not estimate the robot trajectory and the map accurately when working with WiFi because it finds it difficult to converge into an accurate solution. In addition, the FastSLAM diverges in areas where there are not observations from the beacons. These issues suggest the use of optimization techniques

Table IV. SAM. UAH results.

Error		UAH1	UAH2
Global error	Mean (m)	2.672	1.505
	90th percentile (m)	4.985	2.593
Local translational error	Mean (m)	0.075	0.057
	90th percentile (m)	0.178	0.155
Local rotational error	Mean (rad)	0.070	0.027
	90th percentile (rad)	0.008	0.005
Map	Mean (m)	2.436	1.815


(a) UAH1

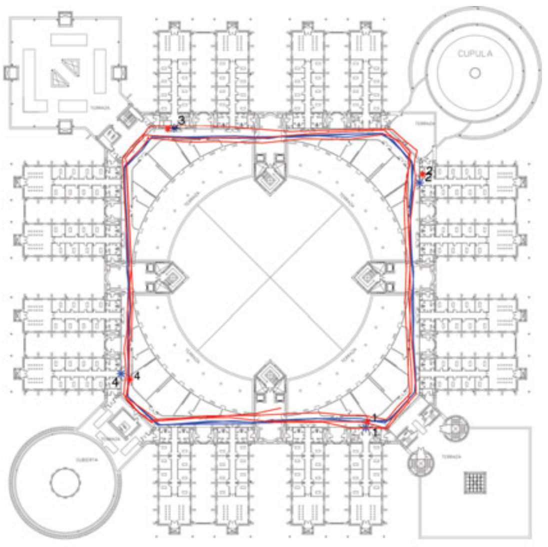

(b) UAH2

Fig. 6. SAM. UAH results.

in order to take advantage of the complete trajectory of the robot and the whole set of observation to find an optimum minimum to the SLAM problem.

5.4. SAM results

Figure 6 shows the SAM results for the UAH database using the same color representation as in Fig. 4. Figure 6 demonstrates that the SAM estimations are very similar to the ground truth. Adding the complete trajectory of the robot and all the WiFi measurements into the optimization problem allows the algorithm to correct the odometry and range errors. The estimated trajectory of the robot is smoothed by SAM and the error is minimized along the trajectory. The map is also well built taking into account the uncertainty of WiFi measurements.

Table IV presents the study of the errors that have been obtained by the SAM algorithm. It shows the global, local, and map errors. In terms of global error, SAM obtains a maximum mean error of 2.6 m and a maximum 90th percentile of 4.98 m. It is noteworthy the results of UAH2 dataset where the SAM obtains a mean error of 1.5 m and a 90th percentile of 2.6 m which are very good results for indoor SLAM with WiFi technology. The local errors are also small and similar to those obtained with other methods. The SAM also performs accurately in terms of map error since it obtains a mean error of 2.4 m for UAH1 dataset and 1.8 m for UAH2 dataset.

The performance analysis of the SAM demonstrates that this algorithm accurately solves the full SLAM problem with RO sensors. The fact that SAM uses the complete trajectory of the robot improves the trajectory and map estimations. In addition, the SAM is able to correct errors that happened in the past by taking advantage of the factor graph representation and optimization techniques.

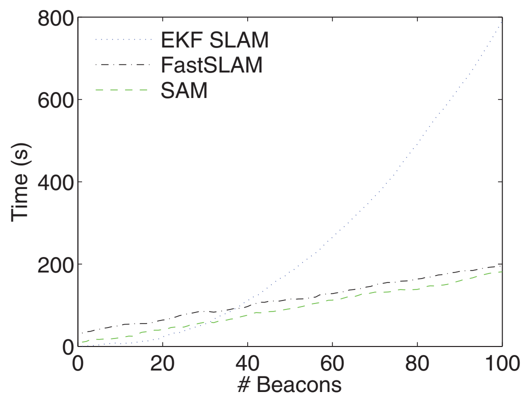


Fig. 7. Computational complexity analysis. Mean of the step time per number of beacons.

5.5. Computational complexity analysis

Computational complexity is an important aspect of SLAM algorithms which defines the computational requirements of the algorithms. Since computational complexity of SLAM algorithms deeply depends on the number of beacons B, this section presents a comparison of the effect on the SLAM methods of the number of beacons. Due to the fact that in real environments the number of beacons is usually small, here the computational complexity is studied by means of simulated data. Figure 7 shows the mean of the step time of the three methods over the same dataset against the number of beacons.

It can be seen from Fig. 7 that computational complexity of the EKF SLAM is quadratic $O(B^2)$ due to the covariance update step. The covariance step requires the multiplication of the Kalman gain matrix the dimension of which is $(2B + 3)$. The computational time of the FastSLAM presents a linear computational complexity $O(NB + 1)$ which depends on the number of beacons B and the number of particles N. In this case the number of particles was 20. The computational complexity of the SAM is linear $O(K + B)$ which depends on the number of poses K and the number of beacons. Figure 7 shows that up to 40 beacons the execution time of all the methods is similar but when the number of beacons are bigger than 40 the FastSLAM and the SAM are better solutions in terms of computational complexity.

5.6. Consistency analysis

Apart from a metrical measurement of the error, it is also important to study the consistency of every method. Consistency is defined as the ability of the filter to accurately estimate uncertainty.[28] Ideally, it is measured by comparing the filter's estimate with the probability density function obtained from the ideal method. In practice, when the ground truth solution x for the state variables is available, a statistical test[29] for filter consistency can be carried out on the estimation using the Normalized Estimation Error Squared (NEES), defined as

$$D^2 = (x - \hat{x})^T P^{-1}(x - \hat{x}), \tag{25}$$

where $\hat{x}$ is the estimated mean and P is the covariance matrix for the EKF and FastSLAM or the inverse of the Hessian for the SAM.

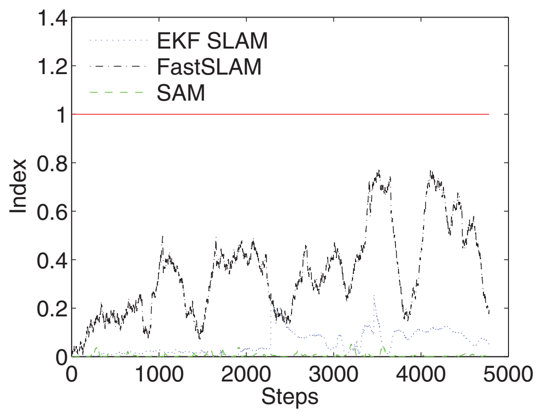
Consistency is checked using a chi-square test:

$$D^2 \leq \chi^2_{r,1-\alpha}, \tag{26}$$

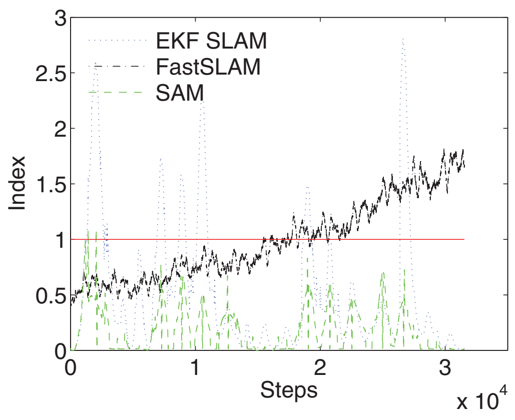
where $r = dim(x)$ and α is the desired significance level (usually $\alpha = 0.05$).

In order to measure the precision of the estimation with respect to the real error this work has used the consistency index (CI):[29]

$$CI = \frac{D^2}{\chi^2_{r,1-\alpha}}, \tag{27}$$



(a) Gesling



(b) UAH

Fig. 8. Consistency analysis.

when $CI < 1$, the estimation is consistent with the ground truth, and when $CI > 1$, the estimation is inconsistent (optimistic) with respect to the ground truth.

Figure 8 shows the study of the consistency for the three SLAM methods. In order to test the effect of radio frequency and WiFi on consistency two different scenarios have been used, one from the Carnegie Mellon database and another one from the UAH database. Figure 8(a) presents the evolution of the consistency in the Carnegie Mellon database. It can be seen that the estimations of all the methods are consistent because the RO sensor that was used in this dataset can be modeled by the algorithms. Figure 8(b) shows the consistency test for the UAH database. Algorithms are more inconsistent in this case which is caused by the uncertainly of WiFi. It is remarkable that EKF SLAM and FASTSLAM become optimistic while the SAM remains consistent along the experiment. The EKF SLAM inconsistency is due to the fact that non-linearities of WiFi introduce errors in the update of the beacon estimations due to variations in linearization. In addition, the true probability distribution of WiFi sensors is non-Gaussian, so that even the true mean and variance may not be an adequate description. These factors affect the convergence of the filter. The consistency of the SAM is due to the ability of this method to work with non-linearities.

5.7. Results comparison

This section closes with a comparison between our proposals to solve the SLAM problem with RO sensors (EKF, FastSLAM, and SAM) and ROP-EKF solution proposed by Djugash in ref. [9]. The comparison is made using the Carnegie Mellon database and the UAH one.

Table V presents a summary of the results of SLAM methods in order to compare them. It only summarizes the global and map errors since they provide more understandable information to the comparison. It shows also the errors that have been obtained by the ROP-EKF. The analysis of computational complexity and consistency, that have been presented before, demonstrates that the SAM is a fast, efficient, and robust alternative to traditional filtering techniques. This fact is especially remarkable in terms of consistency when working with WiFi technology. In terms of global error and improvement error rate, the SAM obtains an average improvement of 66% with respect to the EKF, 52% with respect to the FastSLAM, and 45% with respect to the ROP-EKF. In terms of map error and improvement error rate, the SAM obtains an average improvement error rate of 55% with respect to the EKF, 41% with respect to the FastSLAM, and 5% with respect to the ROP-EKF. The improvement error rate of the SAM with respect to the ROP-EKF in terms of map error is small as the ROP-EKF performs better for two of the datasets (Gesling2 and Plaza1), however, the general performance of SAM is better specially with the UAH database. The FastSLAM performs better than the ROP-EKF when using WiFi technology but the ROP-EKF is able to obtain better results with the Carnegie Mellon database due to the small amount of noise in the range observations.

Table V. Results comparison.

Method	Error	Gesling1	Gesling2	Gesling3	Plaza1	Plaza2	UAH1	UAH2
EKF	Global error (m)	2.08	1.26	1.65	1.53	1.52	5.33	5.19
	map (m)	1.89	1.24	1.22	1.56	2.02	4.24	5.28
FastSLAM	Global error (m)	0.97	1.06	1.06	1.09	1.36	3.03	4.50
	map (m)	0.79	1.42	0.93	1.44	1.62	2.54	3.09
SAM	Global error (m)	**0.32**	**0.48**	**0.52**	**0.51**	**0.56**	**2.67**	**1.51**
	map (m)	**0.32**	1.14	**0.51**	0.78	**0.45**	**2.44**	**1.82**
ROP-EKF	Global error (m)	1.01	0.75	0.88	0.97	0.59	5.41	4.99
	map (m)	0.55	**0.50**	0.61	**0.53**	0.59	5.16	6.52

6. Conclusions

In this work, three SLAM techniques have been presented to solve the SLAM problem with RO sensors. The methods for adapting these techniques to work with RO sensors have been shown. In addition, their advantages and drawbacks have been studied.

In order to test the algorithms in indoor environments with WiFi technology, a new database, the UAH database, has been collected. This database is an extension of the one presented in ref. [9]. The database has been made public to the research community.

The EKF algorithm is able to solve the online problem by means of an easy implementation which converts the EKF into a simple solution to the SLAM problem. A particle filter stage has been proposed to account for an initial estimation of the beacon which is required by the EKF and otherwise not provided by the RO sensors. However, the EKF has presented some drawbacks when working with large amounts of non-Gaussian noise which makes EKF optimistic when working with WiFi technology. In addition, it has been shown that the computational complexity of EKF SLAM increases with the number of beacons. A comparison with the other two shows that the complexity of the EKF SLAM is the greatest for more than 40 beacons. These findings suggest that the EKF SLAM is a good solution when working with ans small number of reliable RO sensors in outdoor environments.

The FastSLAM algorithm has solved online and full SLAM problems. In order to adapt the FastSLAM to RO sensors, the typical EKFs, that FastSLAM uses to estimate the beacons, have been proposed to be replace by particle filters. In addition, it has been proposed that the particle filters are maintained when they converge to a solution. This decision is based on the fact that the number of beacons is not usually large and particle filters can be used without increasing the computational complexity of FastSLAM significantly. Computational complexity of FastSLAM is "$NB + 1$", where N is the number of particles and B is the number of beacons. Hence, the particle representation allows FastSLAM to be a robust solution in terms of typical non-Gaussian noise and multimodal distributions of RO sensors. However, the FastSLAM has presented some problems when no observations are available, which makes FastSLAM diverge.

The SAM algorithm has been studied as a viable alternative to the filtering approach of the problem. It has solved the full SLAM problem by means of a factor graph representation and optimization problems. The SAM has been almost directly applied to RO sensors because the pose of the robot, when a beacon is observed for the first time, was used as the beacon initial estimate. This decision is based on the fact that the optimization problem is able to find a minimum to the problem in spite of the error in the initial estimation. The SAM works as a batch optimization problem which is useful to solve the problem presented in this work. The results of SAM have surpassed the results of the other experiments in almost all of the case studies. It has been demonstrated that the SAM is a better alternative in terms of global error and consistency which is remarkable when working with WiFi in indoor environments. The fact that the computational complexity of the SAM is $O(n)$ makes it an efficient solution when working with a large number of beacons. However, it would be desirable to use online alternatives to SAM such as those mentioned in ref. [30].

Acknowledgments
This work has been funded by the "Ministerio de Economía y Competitividad" through TIN2011-29824-C02-01 and TIN2011-29824-C02-02 (ABSYNTHE project).

References

1. P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," *Proceedings of the IEEE Infocom 2000*, vol. 2 (IEEE Computer and Communications Societies, Tel-Aviv, Israel, 2000) pp. 775–784.
2. M. Ocaña, L. M. Bergasa, M. Á. Sotelo, R. F. Flores, D. F. Llorca and D. Schleicher, "Automatic training method applied to a WiFi+ultrasound POMDP navigation system," *Robotica* **27**(07), 1049–1061 (Dec. 2009).
3. V. Matellán, J. M. Cañas and O. Serrano, "WiFi localization methods for autonomous robots," *Robotica* **24**(4), 455–461 (2006).
4. S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)* (MIT Press, Cambridge, Massachusetts, 2005).
5. M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *Proceedings of the AAAI National 2002* (Conference on Artificial Intelligence, Edmonton, Canada, 2002) pp. 593–598.
6. S. Thrun, W. Burgard, D. Fox *et al.*, *Probabilistic Robotics,* vol. 1 (MIT Press, Cambridge, Massachusetts, 2005).
7. F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.* **25**(12), 1181–1203 (2006).
8. J. Djugash and S. Singh, "A Robust Method of Localization and Mapping Using Only Range," *Proceedings of the International Symposium on Experimental Robotics 2008* (Springer Tracts in Advanced Robotics, Athens, Greece, 2008) pp. 341–351.
9. J. Djugash, B. Hamner and S. Roth, "Navigating with ranging radios: Five data sets with ground truth," *J. Field Robot.* **26**(1), 689–695 (Sep. 2009).
10. J. Djugash, S. Singh and P. Corke, "Further Results with Localization and Mapping Using Range from Radio," *Proceedings of the International Conference on Field & Service Robotics 2005* (Springer Tracts in Advanced Robotics, Port Douglas, Australia, 2005) pp. 231–242.
11. J.-L. Blanco, J. González and J.-A. Fernández-Madrigal, "A Pure Probabilistic Approach to Range-only SLAM," *Proceedings of the Internaitonal Conference on Robotics and Automation, Pasadena, USA* (May 19–23, 2008) pp. 1436–1441.
12. J. L. Blanco, J. A. Fernandez-Madrigal and J. Gonzalez, "Efficient Probabilistic Range-only SLAM," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Robotics and Automation Society, Nice, France, 2008) pp. 1017–1022.
13. B. Ferris, D. Fox and N. Lawrence, "Wifi-slam Using Gaussian Process Latent Variable Models," *Proceedings of the 20th International Joint Conference on Artificial Intelligence 2007*, vol. 7 (Hyderabad, India, 2007) pp. 2480–2485.
14. J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun and A. Aggarwal, "Efficient, Generalized Indoor WiFi Graphslam," *Proceedings of the IEEE International Conference on Robotics and Automation 2011* (IEEE Robotics and Automation Society, Shanghai, China, 2011) pp. 1038–1043.
15. A. Kotanen, M. Hannikainen, H. Leppakoski and T. Hamalainen, "Positioning with Ieee 802.11b Wireless Lan," *Proceedings of the 14th IEEE Conference on Personal, Indoor and Mobile Radio Communications 2003*, vol. 3 (IEEE, Beijing, China, 2003) pp. 2218–2222.
16. F. Herranz, Simultaneous Localization and Mapping Using Range Only Sensors, *Ph.D. Dissertation* (Alcalá de Henares, Spain: University of Alcalá, Sep. 2013).
17. J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous Map Building and Localization for an Autonomous Mobile Robot," *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems 1991* (IEEE Robotics and Automation Society, Osaka, Japan, 1991) pp. 1442–1447.
18. E. Olson, J. Leonard and S. Teller, "Robust Range-only Beacon Localization," *Proceedings of the IEEE/OES Conference on Autonomous Underwater Vehicles 2004* (IEEE Oceanic Engineering Society, 2004) pp. 66–75.
19. F. Herranz, M. Ocaña, L. Bergasa, N. Hernández, A. Llamazares and C. Fernández, "Mapping Based on a Noisy Range-Only Sensor," **In**: *Computer Aided Systems Theory–EUROCAST 2011* (R. Moreno-Diaz, F. Pichler and A. Quesada-Arencibia) (Springer Berlin Heidelberg, 2012) pp. 420–425.
20. K. V. Mardia, "Measures of multivariate skewness and kurtosis with applications," *Biometrika* **57**(3), 519–530 (1970).
21. M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *Proceedings of the National Conference on Artificial Intelligence 2002* (Conference on Artificial Intelligence, Edmonton, Canada, 2002) pp. 593–598.
22. F. R. Kschischang, B. J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory* **47**, 498–519 (1998).

23. R. M. Eustice, H. Singh and J. J. Leonard, "Exactly Sparse Delayed-state Filters," *Proceedings of the IEEE International Conference on Robotics and Automation 2005* (IEEE Robotics and Automation Society, Barcelona, Spain, 2005) pp. 2417–2424.
24. T. A. Davis, J. R. Gilbert, S. I. Larimore and E. G. Ng, "A column approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.* **30**(3), 353–376 (Sep. 2004).
25. F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robots* **4**(4), 333–349 (1997).
26. W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kummerle, C. Dornhege, M. Ruhnke, A. Kleiner and J. Tardós, "A Comparison of SLAM Algorithms Based on a Graph of Relations," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2009* (IEEE Robotics and Automation Society, St. Louis, USA, 2008) pp. 2089–2095.
27. G. Grisetti, C. Stachniss and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. Robot.* **23**(1), 34–46 (2007).
28. T. Bailey, J. Nieto and E. Nebot, "Consistency of the Fastslam Algorithm," *Proceedings of the IEEE International Conference on Robotics and Automation 2006* (IEEE Robotics and Automation Society, Orlando, USA, 2005) pp. 424–429.
29. L. M. Paz, J. Tardos and J. Neira, "Divide and conquer: EKF SLAM in O(n)," *IEEE Trans. Robot.* **24**(5), 1107–1120 (Oct. 2008).
30. M. Kaess, A. Ranganathan and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.* **24**(6), 1365–1378 (2008).