A SLAM algorithm for indoor mobile robot localization using an Extended Kalman Filter and a segment based environment mapping

Luigi D'Alfonso, Andrea Griffo, Pietro Muraca, Paolo Pugliese

***Abstract*—The present paper faces the Simultaneous Localization And Mapping (SLAM) problem for a mobile robot in an unknown indoor environment. A set of segments is used to model the robot surrounding environment and segments' starting and ending points are used as SLAM landmarks. A segment based mapping algorithm is proposed and used along with an Extended Kalman filter driven by measurements taken by ultrasonic sensors located on the robot. The proposed SLAM algorithm has been tested in both simulated and real experiments yielding to encouraging estimation and mapping results.**

I. Introduction

Smith and Cheeseman [1] introduced the Simultaneous Localization And Mapping (SLAM) problem in the 80's and since then this problem has received very considerable attention. The SLAM problem involves the parallel estimation of a mobile robot pose (position and orientation) and of the environment where the robot moves.

To solve the SLAM problem, the robot needs to build an environment map and to localize itself within this map. In mobile robotics the SLAM problem is considered as *chicken and egg* problem due to the strong correlation between the localization task and the mapping task. To know where the robot is, a good environment mapping is needed but to have a reliable environment mapping, the robot's position and orientation should be estimated as best as possible.

The literature shows various solutions to the SLAM problem [2], [3], [4], [5], [6] using various sensors types, e.g. wheel encoders, laser range sensors, sonar range sensors and cameras. Each SLAM algorithm is really influenced by the used sensors types and the same algorithm can be very efficient using some sensors but it can yield to poor performance using different ones. Thanks to its sensors, the robot can observe (sometimes only partially and inexactly) itself and the world where it moves. Once the sensors' measurements have been acquired, depending on the measurements' typology, various algorithms can be used to obtain the robot pose and to reconstruct the robot environment. For instance, laser sensors measurements can be fused through an Extended Kalman filter (EKF) as shown in [2], while camera measurements can be used along with a Rao-Blackwellised particle filter (RBPF) to solve the SLAM problem, see [4]. Laser sensors and cameras probably represent the best way to face the SLAM problem but they have some drawbacks. Laser sensors can be very expensive while cameras provide a large amount of information and it may be difficult and computationally onerous to extract this information starting from the camera image. The SLAM problem can be solved also using ultrasonic sensors. These sensors are less expensive than laser scanners and range cameras, their use is computationally cheap and they work well also in dark or transparent environments, where cameras usually fail. Many works can be found in the literature on the use of the ultrasonic sensors in SLAM applications. In [6] Tardos et al. describe a technique to face the SLAM problem using standard sonar sensors and the Hough transform [9] to detect corners, edges and walls into the environment starting from acquired sonar data. In [7] a pose-based algorithm to solve the SLAM problem for an autonomous underwater vehicle is proposed. A probabilistic scan matching technique using range scans gathered from a mechanical scanning imaging sonar is used along with two Extended Kalman filters, one for the estimation of the local path traveled by the robot and the second one that estimates and keeps the registered SLAM landmarks.

Very often the SLAM algorithms assume to have at least some information about the environment or they assume to model the environment in a very approximated way. For example in [8] the authors assume the robot placed in an environment modeled as a set of orthogonal-parallel lines. On the contrary, in the present paper, we propose a solution to the SLAM problem in an unknown environment using noisy sonar measurements and a very simple but effective structure for the environment map. The proposed solution uses a segment based model for the robot surrounding environment and it solves the SLAM problem thanks to an Extended Kalman filter suitably adapted to the above environment model.

In the SLAM context, it is mandatory to satisfy time constraints since information provided by SLAM algorithms is typically used to compute the output of a control law designed, for example, to make the robot follows a given trajectory or completes a given task. The proposed SLAM algorithm has been developed looking for a very simple but efficient strategy in terms of the algorithm computational requirements. It will be shown, through simulations and experimental tests, that the resulting algorithm can be suitably adapted to satisfy time constraints ensuring good estimation and mapping results.

The paper is organized as follows: in section II the problem statement is described; in section III the proposed SLAM algorithm is discussed; in section IV experimental and numerical results are shown and in section V conclusions are drawn and possible extensions for future investigations are proposed.

II. Problem Statement

Assume to have a mobile robot placed in an unknown environment, the robot is equipped with on board ultrasonic sensors able to provide the distance of the robot from the environment bounds. Since there is no *a-priori* knowledge on the environment, to yield the output equation related to the on board sonar sensors, a model for the environment boundaries is required. In the following subsections the robot and the environment models will be described.

A. Robot model

Consider a battery-powered mobile robot with two independent driving wheels and a castor wheel: a concrete example is the

978-1-4799-2722-7/13/$31.00 2013 IEEE

Authors are with DIMES, Università della Calabria, Rende, Italy. E-mail: {ldalfonso,muraca,pugliese@dimes.unical.it}, and.griffo@gmail.com

Khepera III robot [10], which has been used in the experiments described in Section IV. For such robot, an approximated, discrete-time model is [11]:

$$\begin{cases} x^1_{k+1} = x^1_k + V_k\, T\cos(\theta_{k+1}) + w^1_k \\ x^2_{k+1} = x^2_k + V_k\, T\sin(\theta_{k+1}) + w^2_k \\ \theta_{k+1} = \theta_k + \Delta_k + w^\theta_k, \end{cases} \tag{1}$$

where:

- (x^1_k, x^2_k) is the robot position and θ_k is the angle between the robot axle and the x^1-axis, at time t_k;
- $V_k = r\,(\omega^l_k + \omega^r_k)/2$ is the robot linear velocity and ω^l_k, ω^r_k are the wheels angular velocities;
- w^1_k, w^2_k, w^θ_k are zero-mean uncorrelated Gaussian noises;
- r is the wheels radius and d is the distance between the active two wheels;
- $\Delta_k = r\,(\omega^l_k - \omega^r_k)T/d$ is the rotation within the sampling period $T = t_k - t_{k-1}$.

The model input variables are the wheels angular velocities ω^l_k and ω^r_k, and they have been precomputed so that the robot follows the desired trajectories. The model's state variables are $x_k = [x^1_k \quad x^2_k \quad \theta_k]^T$. The Gaussian disturbances take into account unmodeled dynamics, friction, wheels slipping and also, if the case, external disturbances such as wind.

B. Environment model

The robot is assumed to be equipped with $n_S = 5$ ultrasonic sensors, placed as shown in Figure 1. The environment will be modeled as a set of segments such that each of them intersects at least at one point of the environment boundaries (Figure 1).

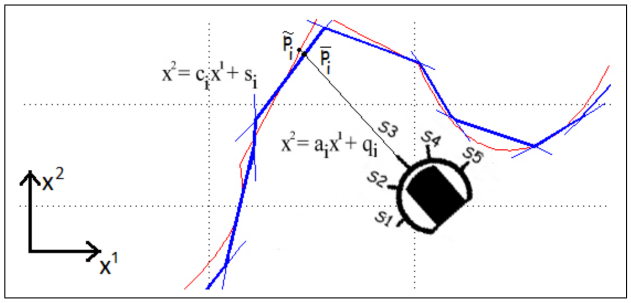

Fig. 1. real environment (red line), segment based modeled environment (blue line), robot on board sensors positions

Each sensor S_i provides, at each step k, the distance from the robot center, denoted by $p = (x^1_k, x^2_k)$, to one point on the environment boundaries, denoted by $\tilde{p}_i = (\tilde{x}^1, \tilde{x}^2)$.
Considering the model used for the environment, the measurement provided by the sensor S_i is approximated, as shown in Figure 1 for the case $S_i = S_3$, by the distance from p to the intersection point, $\overline{p}_i = (\overline{x}^1, \overline{x}^2)$, between the axis of the sensor S_i and one of the environment modeling segments.
Marking the axis of the sensor S_i as $x^2 = a_i x^1 + q_i$ and denoting the axis of the modeling segment in front of the sensor as $x^2 = c_i x^1 + s_i$, the coordinates of the intersection point $\overline{p}_i$ are given by

$$\overline{x}^1 = \frac{s_i - q_i}{a_i - c_i}, \qquad \overline{x}^2 = \frac{a_i s_i - c_i q_i}{a_i - c_i}\ , \tag{2}$$

and the measurement, y_i, from sensor S_i, is modeled as $y_i = \sqrt{(x^1_k - \tilde{x}^1)^2 + (x^2_k - \tilde{x}^2)^2}$ and it is approximated by

$$y_i \approx \sqrt{(x^1_k - \overline{x}^1)^2 + (x^2_k - \overline{x}^2)^2}\ . \tag{3}$$

Since the robot structure is given, the orientation, α_i, of each sensor S_i, with respect to robot axis (placed on third sensor axis, orthogonal to the wheel axes), is known and then the axis of the sensor S_i is given by:

$$a_i = \tan(\theta_k + \alpha_i), \quad q_i = x^2_k - a_i x^1_k\ . \tag{4}$$

By replacing (3) with (4), an observation function depending on robot state and segment parameters (s_i, c_i), is obtained as

$$y_i \approx h((x^1_k, x^2_k, \theta_k), (s_i, c_i)), \quad i = 1, \dots, n_S$$

These relationships define the output equation of the robot model and they will be written in the more compact form

$$y_k \approx h(x_k, (\overline{s}_k, \overline{c}_k)) + v_k, \tag{5}$$

where v_k is a Gaussian noise and it is assumed uncorrelated with $w_k = [w^1_k \quad w^2_k \quad w^\theta_k]^T$. The vectors $\overline{s}_k$ and $\overline{c}_k$ contain the parameters (s_i, c_i), of the segments intercepted by the sensors $S_i, i = 1, \dots, n_S$ at step k.
The goal of the present work is to estimate the position and orientation of the robot, $x_k = [x^1_k \quad x^2_k \quad \theta_k]^T$, and to simultaneously find a segment based approximation of the robot surrounding environment. To achieve this goal, a segment based SLAM algorithm will be described in the next Section.

III. Segment based SLAM

Bailey et Al [12],[13] divide a SLAM algorithm into 5 main parts: landmark extraction, data association, state estimation, state update and landmark update. Using the sonar measurements and the actual robot pose estimation, the data association and landmark extraction processes are performed and yield to the actual environment mapping. Starting from this mapping and from the model inputs, the state estimation, state update and landmark update processes provide the robot pose and update the environment map.

A. Landmark extraction and Data association

Starting from the segment based model for the robot surrounding environment, the proposed Segment based SLAM algorithm uses the starting and ending points of each modeling segment as landmarks. The main idea behind the landmark extraction and data association algorithm is to use the currently acquired measurements to update the actual environment mapping by finding new modeling segments of the surrounding environment or by improving the previously obtained ones.
The following notation will be used:

- $E_k = [p_{k,1}, p_{k,2}, \dots, p_{k,n_k}]$ is an array of n_k *sorted* points. In the following, an array of points will be considered sorted if each couple $(p_{k,i}, p_{k,i+1})$, of consecutive points, represents the starting and ending points of one of the environment modeling segments.
- $\mathcal{E}_k$ is a set containing all the points in E_k.
- Π_k is the set of currently acquired environment points. Using the state estimation $\hat{x}_k$ along with the measurements y_k provided by the on board ultrasonic sensors, an approximation of the environment points detected by the sensors can be obtained as $b^i_k, i = 1, \dots, n_S$:
$$b^i_k = [\hat{x}^1_k + y_{i,k}\cos(\hat{\theta}_k + \alpha_i),\ \hat{x}^2_k + y_{i,k}\sin(\hat{\theta}_k + \alpha_i)] \tag{6}$$
where $y_{i,k}$ is the measurement provided by sensor S_i.

The goal of the landmark extraction and data association process is to find E_k starting from the previous environment mapping E_{k-1}, the predicted robot state $\hat{x}_{k|k-1}$ and the acquired measurements y_k. To accomplish this task the main idea is to compute a new landmark related to each point in Π_k. The obtained n_S landmarks are then suitably added to E_{k-1} yielding to E_k.

More precisely, at each step k, given the prediction $\hat{x}_{k|k-1}$ and the acquired measurements y_k, the set Π_k is computed. For each point $b_k^i \in \Pi_k$ the set of points close, in a neighborhood δ, to b_k^i is obtained as

$$\Omega_i = \{p \in \{\mathcal{E}_{k-1} \bigcup \Pi_k\} : ||b_k^i - p|| \leq \delta\}$$

and $\delta > 0$ is one of the algorithm parameters. For each set Ω_i a virtual point q_i is obtained computing the weighted mean of the points contained in the set (see Figure 2(a)). More precisely, the set Ω_i can contain currently acquired points or virtual points obtained in the previous steps. The new virtual point q_i is obtained as

$$q_i = g_x(\hat{x}_{k|k-1}, y_k) = \frac{1}{M} \sum_{p_i \in \Omega_i} m_i p_i, \quad M = \sum_{i=1}^{|\Omega_i|} m_i \qquad (7)$$

where $|\Omega_i|$ is the number of elements contained in Ω_i, $m_i = 1$ if p_i is one of the currently acquired points ($p_i \in \Pi_i$), while if p_i is one of the previously obtained virtual points, computed using n_i points, then $m_i = n_i$.

Please note that the proposed weighted mean is a function of the robot predicted pose $\hat{x}_{k|k-1}$ and of the currently acquired measurements y_k since each point into Π_k is computed using (6) and thus the function $g_x(\hat{x}_{k|k-1}, y_k)$ can be obtained by replacing (6) with (7).

The proposed weighted mean is such that the more the number of points represented by a previously computed virtual point is big, the bigger this point influence will be on the new virtual point q_i. At the end, the number of points described by the new virtual point q_i is M and it will be related to the new virtual point for future algorithm executions. For each set Ω_i there will be a new virtual point q_i and each of these points will be related to a number of mapped points M. Once all the new virtual points have been obtained, each set of points Ω_i can be removed from E_{k-1}. Therefore, the new array E_{k-1} can be computed as the sorted array (following the previously described sorting definition) containing the points in $\mathcal{E}_{k-1} = \tilde{\mathcal{E}}_{k-1}$, where $\tilde{\mathcal{E}}_{k-1} = \mathcal{E}_{k-1} \setminus (\bigcup_{i=1}^{n_S} \Omega_i)$. Figure 2(a) shows E_{k-1} before the Ω_i deleting process while Figure 2(b) shows the resulting E_{k-1} after the Ω_i deleting process. As shown in these Figures, after the Ω_i deleting process all the segments related to starting and ending points in Ω_i are removed.

At this point, the new virtual points $q_i, i = 1, \ldots, n_S$ have to be inserted into the vector E_{k-1}. As previously remarked, this vector contains the sorted sequence of starting points and ending points of each of the environment modeling segments. To insert a new virtual point q_i into E_{k-1} without infringing this ordering three main steps are performed. First of all, the line $\overline{r}$ between the robot position and the virtual point is computed. As a second step, the subvector e_{k-1} of E_{k-1} containing all the segments in E_{k-1} intercepted by $\overline{r}$, and in front of the robot, is obtained. Finally, the segment $\hat{l}_{k-1}$ contained in e_{k-1} and closest to the robot along the line $\overline{r}$ is computed and let $p_{k-1,j}, p_{k-1,j+1}$ be the segment starting and ending points respectively (see Figure 2(b)). To ensure the array E_{k-1} is sorted after the point q_i is added, this point has to be inserted between the starting point and the ending point of $\hat{l}_{k-1}$. Therefore when a new virtual point q_i is inserted into E_{k-1}, one of the previously obtained segments is deleted and substituted by two new segments, the first one ends in q_i while the second one starts from q_i. Figures 2(c) and 2(d) show an example of virtual point insertion into the landmarks array E_{k-1}, in particular Figure 2(d) shows the resulting environment mapping after the new point insertion.

Note that after the Ω_i deleting process, the resulting intermediate mapping (black line in Figure 2(b)) is usually worse than the previous one (black line in Figure 2(a)) but adding the obtained virtual point q_i, the final mapping (black and green line in Figure 2(d)) is better and it contains a lower number of segments than the starting mapping (black line in Figure 2(a)). Moreover comparing the segments related to the points contained into Ω_i (orange points in Figure 2(a)) with the segments related to q_i (Figure 2(d)), the second ones are less influenced by noise than the first ones thanks to the equation (7) which can be seen as a noise filtering operation.

As a drawback, using the described insertion strategy, segments with very short length can be chosen after the new virtual point insertion; moreover starting from a single segment, two new segments are always obtained and, as a consequence, the number of segments can increase step by step with a resulting high computational cost for the algorithm. To face these problems, a minimum acceptable segment length, $\sigma > 0$, has been defined and if both new segments have a length less or equal than σ, they are discarded. Let $E_{k-1} = [p_{k-1,1}, \ldots, \mathbf{p_{k\text{-}1,j}}, \mathbf{p_{k\text{-}1,j+1}}, \ldots, p_{k-1,n_{k-1}}]$ be the array containing the landmarks, and let $\mathbf{q_i}$ be a virtual point to insert into this array. If $||\mathbf{p_{k\text{-}1,j}} - \mathbf{q_i}|| > \sigma$ or $||\mathbf{q_i} - \mathbf{p_{k\text{-}1,j+1}}|| > \sigma$ then the resulting sorted array will be

$$E_{k-1} = \left[p_{k-1,1}, \ldots, \mathbf{p_{k\text{-}1,j}}, \mathbf{q_i}, \mathbf{p_{k\text{-}1,j+1}}, \ldots, p_{k-1,n_{k-1}}\right]$$

where the points $\mathbf{p_{k\text{-}1,j}}, \mathbf{p_{k\text{-}1,j+1}}$ are computed using the previously described strategy. The related set of elements will be $\mathcal{E}_{k-1} = \tilde{\mathcal{E}}_{k-1}$ where $\tilde{\mathcal{E}}_{k-1} = \mathcal{E}_{k-1} \bigcup \{\mathbf{q_i}\}$. Otherwise, if the condition on the minimum segments length is not satisfied, the new virtual point is not added and the array E_{k-1} and the set $\mathcal{E}_{k-1}$ do not change.

Note that the parameter σ has to be accurately chosen depending on the desired performance. If $\sigma \to 0$ then very short segments are accepted resulting in a more precise environment estimation but also in a high computational cost. On the contrary, if a high σ value is chosen then the algorithm computational cost will be low but the resulting estimation and mapping performance can be very poor. After all the new virtual points have been considered, the obtained array E_{k-1} and its related set of points $\mathcal{E}_{k-1}$ can be considered as the updated environment map, thus $E_k = E_{k-1}$ and $\mathcal{E}_k = \mathcal{E}_{k-1}$. The entire landmark extraction and data association process will be denoted in the following Sections as

$$E_k = \mathcal{L}(E_{k-1}, y_k, \hat{x}_{k|k-1})$$

B. State estimation, State and Landmark Update

The state estimation and the state and landmark update problems can be solved using an Extended Kalman filter.

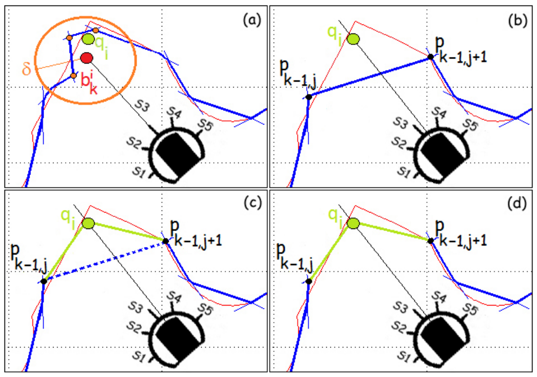

Fig. 2. (a): virtual point q_i computation, the orange points and the red point are in Ω_i; (b) segment $\hat{l}_k$ computation; (c) and (d): virtual point insertion

Given a nonlinear stochastic system from noisy measurements, represented by the equations

$$\begin{aligned} x_{k+1} &= f(x_k,u_k)+w_k \\ y_k &= h(x_k)+v_k, \end{aligned} \tag{8}$$

where x_k is the state to be estimated, y_k is the measured output, w_k and v_k are the process and measurements noises, the Extended Kalman filter (EKF) is based on the linearization of the nonlinear maps (f,h) of (8) around the estimated trajectory. The filter is based on the assumption that the initial state, the measurement noise and the process noise are Gaussian and uncorrelated each other. From the computational point of view the EKF is simply a time-varying Kalman filter where the dynamic and output matrices are given by

$$A_k = \left.\frac{\partial f(x_k,u_k)}{\partial x_k}\right|_{x_k=\hat{x}_{k|k}}, \quad C_k = \left.\frac{\partial h(x_k)}{\partial x_k}\right|_{x_k=\hat{x}_{k|k-1}}, \tag{9}$$

and its output is a sequence of state estimates $\hat{x}_{k|k}$ and estimation error covariance matrices $P_{k|k}$.
Ignoring $\bar{s}_k$ and $\bar{c}_k$, using model (1) as the state update function $f(\cdot)$ and equation (5) as model output equation, the described framework falls within (8) on defining

$$x_k=[x_k^1 \;\; x_k^2 \;\; \theta_k]^T, \quad u_k=[\omega_k^l \;\; \omega_k^r]^T, \quad w_k=[w_k^1 \;\; w_k^2 \;\; w_k^\theta]^T$$

In the following we will indicate with W and V the w_k and v_k covariance matrices respectively and these matrices will be assumed to be known a-priori.
To solve the SLAM problem the Extended Kalman filter has to be suitably adapted to provide an estimation of the robot pose along with an estimation of the environment modeling segments with their starting and ending points. To this end, the following augmented state is defined

$$X_k=[x_k^T, p_{k,1}, p_{k,2}, \ldots, p_{k,n_k}]^T$$

containing the robot state x_k and all the starting and ending points of the environment modeling segments. Please note that each point $p_{k,i}$ is represented by its x^1 and x^2 coordinates and thus the dimension of the augmented state is $3+2\times n_k$ and it is a time varying dimension.

The starting and ending points of each segment are considered as landmarks and their positions are obviously constant whatever is the control input. Moreover the landmarks are assumed to be not influenced by any process noise, therefore the state update function, the process noise covariance matrix and the dynamic matrix related to the augmented state will be

$$X_{k+1}=\begin{bmatrix} x_{k+1} \\ p_{k+1,1} \\ \vdots \\ p_{k+1,n_k}\end{bmatrix} = \mathcal{F}\left(\begin{bmatrix} x_k \\ p_{k,1} \\ \vdots \\ p_{k,n_k}\end{bmatrix}, u_k\right) = \begin{bmatrix} f(x_k,u_k)+w_k \\ p_{k,1} \\ \vdots \\ p_{k,n_k}\end{bmatrix}$$

$$\mathcal{W}=\begin{bmatrix} W & \emptyset \\ \emptyset & \emptyset_{2\times n_k}\end{bmatrix}; \mathcal{A}_k=\begin{bmatrix} A_k & \emptyset \\ \emptyset & I_{2\times n_k}\end{bmatrix}$$

where I_q and $\emptyset_q$ are the identity matrix of order q and the null matrix with $q\times q$ elements respectively.
The output equation (5) is a function of both the robot state and the environment modeling segments. Since the parameters (s,c) of each segment can be related to its starting and ending points, the output function can be seen as a function of the augmented state X_k, $y_k=h(x_k,(\bar{s}_k,\bar{c}_k))+v_k=h(X_k)+v_k$. The output matrix will be

$$\mathcal{C}_k=\begin{bmatrix} C_{1,k} & \frac{\partial h_1}{\partial p_{k,1}} & \cdots & \frac{\partial h_1}{\partial p_{k,n_k}} \\ \vdots & & \vdots & \\ C_{n_S,k} & \frac{\partial h_{n_S}}{\partial p_{k,1}} & \cdots & \frac{\partial h_{n_S}}{\partial p_{k,n_k}} \end{bmatrix}$$

where n_S is the robot number of sensors, $C_{i,k}$ is the i-th row of the C_k matrix, h_i is the function $h(X_k)$ related to the i-th sensor and $\frac{\partial h_i}{\partial p_{k,i}}$ is a two dimensional array containing the partial derivatives of h_i w.r.t the $p_{k,i}$ coordinates.
Indicating with $\hat{x}_k$ the estimation of the state x_k and with $\hat{p}_{k,i}$ the estimation of the landmark $p_{k,i}$, the estimation error covariance matrix related to the augmented state is

$$\mathcal{P}_k=\begin{bmatrix} P_k & P(\hat{x}_k,\hat{p}_{k,1}) & \ldots & P(\hat{x}_k,\hat{p}_{k,n_k}) \\ P(\hat{p}_{k,1},\hat{x}_k) & P(\hat{p}_{k,1},\hat{p}_{k,1}) & \ldots & P(\hat{p}_{k,1},\hat{p}_{k,n_k}) \\ \vdots & & \ddots & \\ P(\hat{p}_{k,n_k},\hat{x}_k) & P(\hat{p}_{k,n_k},\hat{p}_{k,1}) & \ldots & P(\hat{p}_{k,n_k},\hat{p}_{k,n_k}) \end{bmatrix}$$

where $P(a,b)$ is the estimation error covariance matrix between a and b.
Due to the landmark extraction process, new points can be inserted into the current mapping E_k or old points can be removed from it. Therefore the number of points n_k can change during the SLAM process and the resulting sizes of the augmented state X_k and of the matrices $\mathcal{A}_k, \mathcal{C}_k, \mathcal{P}_k, \mathcal{W}$ are time varying. After the landmark extraction process, for each change in the mapping, there is a change in the matrices $\mathcal{A}_k, \mathcal{C}_k, \mathcal{P}_k, \mathcal{W}$ and in the augmented state.
Let $E_{k-1}=[p_{k-1,1},\ldots,p_{k-1,j},p_{k-1,j+1},\ldots,p_{k-1,n_{k-1}}]^T$ and assume that $\mathcal{L}(\cdot)$ extracts the point $\mathbf{p}$ and returns $E_k=[p_{k-1,1},\ldots,p_{k-1,j},\mathbf{p},p_{k-1,j+1},\ldots,p_{k-1,n_{k-1}}]^T$. The augmented state becomes $X_k=[x_k,\;E_k]^T$ and, as shown in [14], the estimation error covariance matrix related to the augmented state after the new landmark insertion is

$$\tilde{\mathcal{P}}_k = G_x\mathcal{P}_kG_x^T + G_yVG_y^T\;,\; \mathcal{P}_k=\tilde{\mathcal{P}}_k. \tag{10}$$

The matrix G_x is

$$G_x=\left[\begin{array}{ccccccc} I_3 & \emptyset & \ldots & \emptyset & \emptyset & \ldots & \emptyset \\ \emptyset & I_{2,1} & \ldots & \emptyset & \emptyset & \ldots & \emptyset \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ \emptyset & \emptyset & \ldots & I_{2,j} & \emptyset & \ldots & \emptyset \\ \hline \frac{\partial \mathbf{g_x}}{\partial \mathbf{x_k}} & \emptyset & \ldots & \emptyset & \emptyset & \ldots & \emptyset \\ \hline \emptyset & \emptyset & \ldots & \emptyset & I_{2,j+1} & \ldots & \emptyset \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ \emptyset & \emptyset & \ldots & \emptyset & \emptyset & \ldots & I_{2,n_k} \end{array}\right]$$

and $I_{2,i}$ is the identity matrix of order two related to the i-th landmark while $\emptyset$ is a null matrix of appropriate size.

The matrix G_y is $G_y = \left[\begin{array}{ccc|c|ccc} \emptyset & \dots & \emptyset & \frac{\partial \mathbf{g_x}}{\partial \mathbf{y_k}} & \emptyset & \dots & \emptyset \end{array} \right]^T$ and $\frac{\partial g_x}{\partial y_k}$ is the $j+1$-th row of this matrix. In the above matrices the function g_x is the landmark extraction function (7). For what regards the landmark elimination, if the landmark extraction process deletes a point p_i from E_k, then the corresponding entries in the augmented state and in the matrix $\mathcal{P}_k$ have to be removed. Following the above update rules, it is convenient to define the update function

$$(X_k^*, \mathcal{P}_k^*) = \mathcal{U}(X_k, \mathcal{P}_k, E_{k-1}, E_k)$$

which properly modifies the augmented state and the estimation error covariance matrix according to the variations in the environment mapping from E_{k-1} to E_k. The function outputs are then used as the new augmented state, $X_k = X_k^*$, and as the new estimation error covariance matrix, $\mathcal{P}_k = \mathcal{P}_k^*$.

Finally, at each time step, using the observation structure (5), the landmark extraction function $\mathcal{L}(\cdot)$ and the update function $\mathcal{U}(\cdot)$, the resulting segment based SLAM algorithm can be summarized as follows:

Segment based SLAM

$$\begin{aligned}
&\hat{X}_{k+1|k} = \mathcal{F}(\hat{X}_{k|k}, u_k)\\
&\mathcal{P}_{k+1|k} = \mathcal{A}_k \mathcal{P}_{k|k} \mathcal{A}_k^T + \mathcal{W}\\
&E_{k+1} = \mathcal{L}(E_k, y_{k+1}, \hat{x}_{k+1|k})\\
&(\hat{X}_{k+1|k}^*, \mathcal{P}_{k+1|k}^*) = \mathcal{U}(\hat{X}_{k+1|k}, \mathcal{P}_{k+1|k}, E_k, E_{k+1})\\
&\hat{X}_{k+1|k} = \hat{X}_{k+1|k}^*\\
&\mathcal{P}_{k+1|k} = \mathcal{P}_{k+1|k}^*\\
&K_{k+1} = \mathcal{P}_{k+1|k} \mathcal{C}_{k+1}^T (\mathcal{C}_{k+1} \mathcal{P}_{k+1|k} \mathcal{C}_{k+1}^T + V)^{-1}\\
&\hat{X}_{k+1|k+1} = \hat{X}_{k+1|k} + K_{k+1}(y_{k+1} - h(\hat{X}_{k+1|k}))\\
&\mathcal{P}_{k+1|k+1} = \mathcal{P}_{k+1|k} - K_{k+1} \mathcal{C}_{k+1} \mathcal{P}_{k+1|k}
\end{aligned}$$

Where the algorithm initial conditions are $\mathcal{E}_0 = \emptyset$, an empty array E_0 $(n_0 = 0)$, $\hat{X}_{0|0} = [\hat{x}_{0|0}]$ and $\mathcal{P}_{0|0} = P_{0|0}$. This algorithm consists in an Extended Kalman filter used to estimate the augmented state formed by the robot pose and the landmarks positions. Please note the landmark extraction function is invoked using only the first three elements of the estimated augmented state representing the robot state estimation.

IV. Numerical and Experimental Results

The proposed SLAM algorithm performance has been tested through numerical simulations and real experiments. In both the experimental setup and the numerical simulations the robot Khepera III ([10]) has been used and modeled. The following parameters have been used:

- $d = 0.09m$, $r = 0.0205m$ for the robot Khepera III;
- Sampling period $T = 1s$;
- $V = 0.0004I_5$: a standard deviation of $0.02m$ for the measurements provided by the ultrasonic sensors;
- $W = diag\{0.01^2, 0.01^2, 0.000002\}$: a standard deviation of $0.01m$ on w_k^1 and w_k^2, a standard deviation of $0.1°$on w_k^θ for the robot model;
- $P_{0|0} = diag\{0.05^2, 0.05^2, 0.000002\}$: a standard deviation of $0.05m$ on robot initial position estimation error and a standard deviation of $0.1°$on robot initial heading estimation error;
- $\delta = 0.1$ for the landmark extraction and data association algorithm.

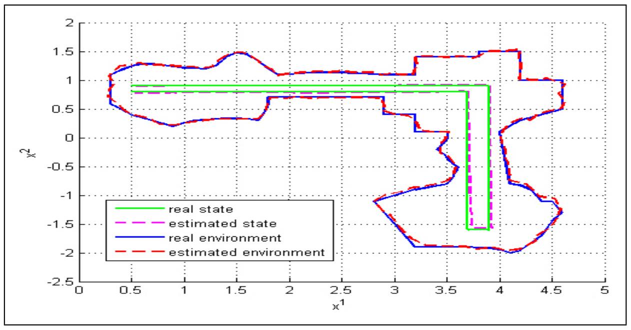

Fig. 3. Segment based SLAM simulation result using $\sigma = 0.08$

To evaluate the proposed algorithm performance, three indexes have been defined. The first index is $\varepsilon = \frac{1}{k_f+1} \sum_{k=0}^{k_f} \frac{||x_k - \hat{x}_{k|k}||}{||x_k||} \times 100$, where k_f is the number of time steps in which each experiment/simulation is performed. The above index represents the percentage relative estimation error between the estimated robot pose and the real one. The index can be used to evaluate the localization performance.

To evaluate the quality of the environment mapping a further index ρ has been used. The index is evaluated as follows: the proposed environment mapping algorithm provides a set of points $p_{k,i}, i = 1, \dots, n_k$ such that each couple $(p_{k,i}, p_{k,i+1})$ represents a segment which maps the robot surrounding environment. Given the i-th couple $(p_{k,i}, p_{k,i+1})$, a set of n_i points, $P^{i,j}, j = 1, \dots, n_i$, equally spaced by an amount equal to 0.01, is computed on the segment between this two points. For each point $P^{i,j}$, the distance $d^{i,j}$ between this point and its nearest real environment boundaries point is computed yielding to the index $\rho = \frac{1}{n_k-1} \sum_{i=1}^{n_k-1} \rho_i$ where $\rho_i = \frac{1}{n_i} \sum_{j=1}^{n_i} d^{i,j}$ and $n_k - 1$ is the number of modeling segments contained into E_k.

The third index is $\tau = \frac{1}{k_f+1} \sum_{k=0}^{k_f} \tau_k$, where τ_k is the execution time of the segment based SLAM algorithm at step k. This index represents the averaged execution time of the SLAM algorithm and it has been used to evaluate the time performance of the proposed algorithm.

In the simulation setup, the Khepera III robot model has been tested in the environment shown in Figure 3. The angular velocities have been precomputed so that the robot follows the trajectory shown in Figure 3, starting from $(0.5, 0.9)m$, with $\theta_0 = 0rad$, passing by $(3.9, 0.9)m$, $(3.9, -1.6)m$, $(3.7, -1.6)$, $(3.7, 0.8)$ and $(0.5, 0.8)$. The path has been performed in $k_f = 500$ steps. 150 numerical experiments have been performed setting $\hat{x}_{0|0} = [0.5\ 0.9\ 0]^T$ as initial condition for the SLAM algorithm. For each simulation, four possible values of the parameter σ have been tested: 0.06, 0.08, 0.1, 0.12. The obtained averaged indexes over the 150 simulations are shown in Table I. The execution times τ_k have been computed using MATLAB R2012b running the proposed algorithm on an Intel Core 2 Duo $T7300$ CPU.

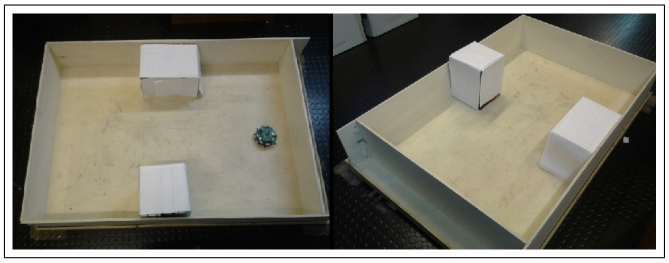

Fig. 4. experimental framework

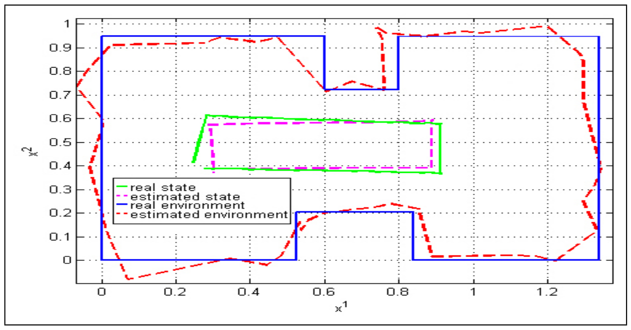

Fig. 5. Segment based SLAM results in a real experiment

Table I shows that the performance of the proposed Segment based SLAM algorithm is satisfying using all the tested σ values since the algorithm yields to appreciable localization and mapping errors. Moreover, choosing $\sigma \geq 0.08$, τ index is always such that the algorithm can be used during the robot evolution with no delays troubles since the algorithm execution time is lower than the chosen sampling period $T = 1s$.

As expected, when σ tends to 0, the estimation and mapping errors decrease and τ index grows; as σ grows, the estimation and mapping performance are deteriorated while the time performance increases. A typical result of the SLAM algorithm, using $\sigma = 0.08$, is depicted in Figure 3.

TABLE I. AVERAGED INDEXES OVER THE 150 PERFORMED SIMULATIONS

index	$\sigma = 0.06$	$\sigma = 0.08$	$\sigma = 0.1$	$\sigma = 0.12$
ε	1.09%	1.21%	1.46%	4.16%
ρ	5.76%	6.15%	7.83%	11.68%
τ	$1.35s$	$0.96s$	$0.85s$	$0.71s$

In the experimental setup, the Khepera III robot has been used in the environment shown in Figure 4. The angular velocities have been precomputed so that the robot follows a rectangular trajectory starting from $(0.4, 0.4)m$, with $\theta_0 = 0rad$, passing by $(1, 0.4)m$, $(1, 0.6)m$, $(0.4, 0.6)$ and coming back to $(0.4, 0.4)$. The path has been performed in $k_f = 100$ steps. A set of 50 experiments have been performed using $\hat{x}_{0|0} = [0.4\ 0.4\ 0]^T$ as SLAM initial condition and $\sigma = 0.05$.

The obtained averaged indexes over the performed 50 experiments are: $\varepsilon = 5.1\%$, $\rho = 4.7\%$, $\tau = 0.41s$ and also in this case they are very encouraging and the time constraints are satisfied. A typical result of the SLAM algorithm in a real experiment is shown in Figure 5. Please note that the computed τ index values are very influenced by the used CPU and by the used operative system. Therefore the shown results can be read only as an indication of the real algorithm time performance. Further experiments using a real time CPU are in progress to better test the proposed SLAM algorithm.

V. Conclusions and Future Research Directions

In this work the SLAM problem for a mobile robot in an unknown indoor environment has been faced. The environment has been modeled using a set of segments and segments' starting and ending points have been used as SLAM landmarks. A Segment based mapping algorithm has been proposed. The SLAM algorithm consists in a modified Extended Kalman filter suitably adapted to face both the mobile robot localization problem and the surrounding environment mapping problem. Measurements taken from a set of on board ultrasonic sensors have been used to update the robot pose estimation and the environment mapping. The performance of the proposed algorithm has been evaluated in both numerical and experimental tests obtaining very good mapping and localization results. Moreover experiments and simulations have shown that the algorithm execution time is low enough to avoid the use of the proposed SLAM in 'real time' in the proposed experimental setup. As future research directions we are working on a new version of the proposed algorithm able to adapt its parameters σ and δ during the algorithm evolution to increase the localization and mapping performance.

References

[1] R. C. Smith, P. Cheeseman, On the representation and estimation of spatial uncertainty, IJRR, vol. 5, no. 4, pp. 5668, 1986

[2] Y.Misono, Y.Goto, Y.Tarutoko, K.Kobayashi, K.Watanabe, Development of Laser Rangefinder-based SLAM Algorithm for Mobile Robot Navigation, SICE Annual Conference 2007, Sept.17-20, Japan

[3] S.Fu,H. Liu,L.Gao,Y.Gai, SLAM for mobile robots using laser range finder and monocular vision, 14th International Conference on Mechatronics and Machine Vision in Practice, 2007. M2VIP 2007, 4-6 Dec.

[4] L.Maohai, H.Bingrong, L.Ronghua, Novel Mobile Robot Simultaneous Loclization and Mapping Using Rao-Blackwellised Particle Filter, International Journal of Advanced Robotic Systems, Vol. 3, No. 3, pp. 231-238, 2006

[5] A.K.Pandey, K.M. Krishna, H.Hexmoor, Feature Chain Based Occupancy Grid SLAM for Robots Equipped with Sonar Sensors, International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS) 2007, April 30-May 3 2007, Waltham Massachusetts

[6] J. D. Tardos, J. Neira, P. M. Newman, and J. J. Leonard, Robust mapping and localization in indoor environments using sonar data, IJRR, vol. 21, no. 4, pp. 311330, Apr. 2002

[7] A.Mallios, P.Ridao, D.Ribas, E.Hern, Probabilistic Sonar Scan Matching SLAM for Underwater Environment, OCEANS 2010, Sydney, 2010

[8] T.N.Yap Jr, C.R.Shelton, SLAM in Large Indoor Environments with Low-Cost, Noisy, and Sparse Sonars, 2009 IEEE International Conference on Robotics and Automation, May 12-17, 2009, Japan

[9] R. O. Duda and P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Comm. ACM, vol. 15, no. 1, pp. 1115, Jan. 1972

[10] K-TEAM Corporation [Online]. Available: http://www.k-team.com

[11] E. Ivanjko, I. Petrović, Extended Kalman Filter based mobile robot pose tracking using occupancy grid maps, presented at the IEEE MELECON, Dubrovnik, Croatia, May 2004, pp. 311–314

[12] T.Bailey, H.Durrant-Whyte, Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms, IEEE Robotics & Automation Magazine, Vol. 13, No. 2. (05 June 2006), pp. 99-110

[13] T.Bailey, H.Durrant-Whyte, Simultaneous Localisation and Mapping (SLAM): Part II State of the Art, IEEE Robotics & Automation Magazine, Vol. 13, No. 3. (21 September 2006), pp. 108-117

[14] L.Pedraza, D. Rodriguez-Losada, F. Matia; G. Dissanayake; J.V. Miro, Extending the Limits of Feature-Based SLAM With B-Splines, IEEE Transactions on Robotics, vol.25, no.2, pp.353,366, April 2009