The Knowledge Engineering Review, Vol. 00:0, 1–24. © 2009, Cambridge University Press
DOI: 10.1017/S000000000000000 Printed in the United Kingdom

Autonomous SLAM driven by Monte Carlo uncertainty maps-based navigation

Fernando A. Auat Cheein, Fernando M. Lobo Pereira[†], Fernando di Sciascio and Ricardo Carelli

Instituto de Automatica, National University of San Juan, San Juan, Argentina
[†]*Faculdade de Engenharia da Universidade do Porto, Porto, Portugal*
E-mail: fauat@inaut.unsj.edu.ar, flp@fe.up.pt, fernando@inaut.unsj.edu.ar, rcarelli@inaut.unsj.edu.ar

Abstract

This paper addresses the problem of implementing a SLAM algorithm combined with a non-reactive controller (such as trajectory following or path following). A general study showing the advantages of using predictors to avoid mapping inconsistences in autonomous SLAM architectures is presented. In addition, this paper presents a priority based uncertainty map construction method of the environment by a mobile robot when executing a SLAM algorithm. The SLAM algorithm is implemented with a Extended Kalman Filter (EKF) and extracts corners (convex and concave) and lines (associated with walls) from the surrounding environment. A navigation approach directs the robot motion to the regions of the environment with the higher uncertainty and the higher priority. The uncertainty of a region is specified by a probability characterization computed at the corresponding representative points. These points are obtained by a Monte Carlo experiment and their probability is estimated by the sum of Gaussians method, avoiding the time-consuming map-gridding procedure. The priority is determined by the frame in which the uncertainty region was detected (either local or global to the vehicle's pose). The mobile robot has a non-reactive trajectory following controller implemented on it to drive the vehicle to the uncertainty points. SLAM real time experiments in real environment, navigation examples, uncertainty maps constructions along with algorithm strategies and architectures are also included in this work.

1 Introduction

This article concerns the problem of the SLAM algorithm which consists in building a map of an unknown environment while the vehicle localizes itself within that map and moves around it. It proposes a scheme enabling the integration of non-reactive control strategies for local or global navigation of a mobile robot in a structured environment while a Simultaneous Localization and Mapping (SLAM) algorithm is continuously executed. The SLAM algorithm consists on a sequential EKF (Extended Kalman filter) feature-based SLAM. This algorithm fuses corners (convex and concave) and lines of the environment in the SLAM system state. The navigation strategy is based on the construction of uncertainty maps based on the Gaussianity of the SLAM system state and the *sum of Gaussians* method. The uncertainty maps provide navigation goals and priorities to the navigation strategy, indicating the regions of the environment of which there is no information available. The algorithms and architectures of the proposal, along with real time experimental results are also shown in this work.

The combination of the SLAM algorithm with a strategy for exploration or navigation within the environment is known as Active SLAM and has been a key problem in the implementation of autonomous mobile robots. The integration of SLAM algorithms with control strategies to govern the motion of a mobile robot and the ability of selecting feasible destinations on its own will endow the vehicle with full autonomy.

In the early development stages of SLAM (Ayache & Faugeras (1989); Chatila & Laumond (1985); Smith et al. (1987); Smith et al. (1990)), the attention of the scientific community focused on solving the issues inherent to the specification of SLAM schemes: optimality, computational cost and consistency. Several algorithms have been proposed as a result. The Extended Kalman Filter (Dissanayake et al. (2001); Castellanos et al. (2004); Bailey et al. (2004); Guivant & Nebot (2001)) (EKF) is one of the first and most used filters to implement a SLAM algorithm, and, among its variants, one may single out the Unscented Kalman Filter (Thrun et al. (2005)), with better performance for non-linear models, and the Information Filter (Thrun et al. (2005)), with better computational performance at the correction stage. More recently, the Particle Filter (Thrun et al. (2005)) and some Bayesian Approaches (Hahnel et al. (2003); Thrun et al. (1998); Dellaert et al. (1999)) to SLAM have enabled significant improvements in the SLAM implementation. Unfortunately, most of this research is confined to simulation or off-line results (Mullane et al. (2008); Huang et al. (2008); Xi et al. (2008)). In what concerns real time implementation, the reduction of the computational cost in the execution of SLAM has been achieved through feature-based SLAM with feature selection (di Marco et al. (2000); Durrant-Whyte & Bailey (2006a); Durrant-Whyte & Bailey (2006b); Auat Cheein et al. (2009b)), as well as topological and hybrid maps (Garulli et al. (2005); Choset & Nagatami (2001); Zunino & Chrinstensen (2001)). However, in order to reduce processing time, most of the feature-based SLAM applications are restricted to single feature environments (Durrant-Whyte & Bailey (2006b); Choset & Nagatami (2001)). Finally, the consistency of the SLAM algorithm has been one of the most studied issues in the last years (Mullane et al. (2008); Zunino & Chrinstensen (2001); Kouzoubov & Austin, (2004); Diosi & Kleeman (2005); Andrade-Cetto & Sanfeliu (2002)). This research effort led to the characterization of the sensitivity of SLAM algorithms with respect to its initial conditions and to the Jacobian matrix associated with the environment feature model (Diosi & Kleeman (2005); Andrade-Cetto & Sanfeliu (2002)). Fusion of the information on the pose of the mobile robot with data from an external sensor, such as GPS (Global Positioning System), is the main approach to maintain the consistency of the SLAM algorithm (Diosi & Kleeman (2004)).

The combination of control strategies with the SLAM algorithm has been addressed from two significantly different points of view. While the first one considers how the control is used to reduce errors during the estimation process (Chatila & Laumond (1985)), the second one concerns exploration techniques providing the best map from the reconstruction perspective. Despite the duality between regulation and estimation, whatever the control strategy is implemented, it will not be guaranteed that, in general, the mobile robot will follow a specific trajectory inside the environment (Bailey et al., 2006). In many applications, the control signal is not considered as an input of the SLAM algorithm, and, instead, odometry measurements of the robot are used (Thrun et al. (2005); Durrant-Whyte & Bailey (2006a); Durrant-Whyte & Bailey (2006b)). Thus, most of the associated implementations focus on the low-level, basic control-reactive behavior (Zunino & Chrinstensen (2001); Diosi & Kleeman (2004)), leaving the motion planning and control as a secondary algorithm. Thus, albeit restricted to a local reference frame attached to the robot, active exploration strategies for indoor environments are proposed in (Xi et al. (2008); Andrade-Cetto & SanFeliu (2006); Liu et al. (2008)). As an example, a boundary exploration problem is proposed in (Xi et al. (2008)). In this case, the robot has to reach the best point determined in the boundary of its local point of view. From a global reference perspective, these implementations have a random behavior inside the environment. To solve the lack of global planning, some implementations have included algorithms for searching optimal path based on the information acquired of the environment (Sim & Roy (2005); Liu et al. (2008)). These algorithms usually require the map to be gridded and, accordingly, they compute a feasible path to a possible destination (closure of the loop or global boundary points) without specifying the control law implemented on the mobile robot. Despite of the advances made so far, the integration of control strategies based on the SLAM system state –map and vehicle– (Durrant-Whyte & Bailey (2006a); Durrant-Whyte & Bailey (2006b)) to guide the robot within an unknown environment from a local and a global reference frame following a pre-established plan is not quite studied or implemented in real time.

The main contribution of this paper is twofold: 1) a general algorithm and uncertainty based navigation strategy for an autonomous SLAM implementation (with non-reactive controllers); and 2) the construction of uncertainty maps based on the Gaussianity of the SLAM system state to determine unvisited regions of the environment -from a probabilistic perspective- guiding the robot to those regions and executing the SLAM algorithm at the same time . The navigation strategy presented in this paper allows the determination of unsatisfactorily mapped or unvisited regions of the environment, which will thus be targeted by the robot and thus permitting the autonomous map improvement. The uncertainty maps construction is based on the *sum of Gaussians* method and the Monte Carlo method. By using the Monte Carlo method instead of gridding the map, we reduce the computational cost of the process and then, by applying the *sum of Gaussians* method to the navigable points of the environment, we are able to construct an uncertainty map of such environment.

This article is completed with an introduction of a general algorithm for a real-time SLAM-Control implementation to avoid inconsistence of the map reconstruction (Andrade-Cetto & Sanfeliu (2006)). This algorithm uses a prediction of the robot's pose up to the time the control law is invoked. Several experimental results are shown along the article. All experiments were carried out in real time. Potential applications of the obtained results are surveillance, metric maps reconstruction, and the construction of probabilistic maps based on the information of the SLAM system state without gridding the environment. During the entire navigation or exploration phase, the SLAM algorithm continues being executed.

This paper is organized as follows. In section 2, the sequential EKF feature-based SLAM is presented; section 3 shows the general algorithm of the SLAM combined with a non-reactive controller; section 4 shows the general method of constructing uncertainty maps, the *sum of Gaussians* method and the navigation strategy based on priority levels of uncertainty regions of the map. Each section shows real time experimental results of their proposal. Finally, section 5 shows the real time experimental result of the navigation strategy combined with the uncertainty maps method.

2 Feature based EKF-SLAM

The SLAM algorithm implemented in this work is solved by an Extended Kalman filter (EKF). The SLAM system state is composed by the vehicle estimated pose –position and orientation– and the features extracted from the environment –which are known as the *map of the environment*–. The features extracted from the environment correspond to corners –concave and convex– and lines –associated with walls–. For visualization and map reconstruction purposes, a secondary map is maintained. This secondary map stores the beginning and ending points of the segments associated with the lines of the environment. Thus, the secondary map allows finite walls' representation. The secondary map is updated and corrected according to the feature correction in the EKF-SLAM system state, and if a new feature is added to that system state, it is also added in the secondary map (Auat Cheein et al. (2009a)). Equations (1) and (2) show the system state structure and its covariance matrix. All elements of the SLAM system state are referenced to a global coordinate system.

$$\hat{\xi}_t = \begin{bmatrix} \hat{\xi}_{v,t} \\ \hat{\xi}_{m,t} \end{bmatrix} \tag{1}$$

$$P_t = \begin{bmatrix} P_{vv,t} & P_{vm,t} \\ P_{vm,t}^T & P_{mm,t} \end{bmatrix} \tag{2}$$

In Eq. (1), $\hat{\xi}_t$ is the SLAM system state; $\hat{\xi}_{v,t} = [\hat{\xi}_{x,t} \ \hat{\xi}_{y,t} \ \hat{\xi}_{\theta,t}]$ is the estimated pose of the vehicle, where $\hat{\xi}_{x,t}$ and $\hat{\xi}_{y,t}$ represent the global position of the agent within the environment and $\hat{\xi}_{\theta,t}$ its orientation; $\hat{\xi}_{m,t}$ represents the map of the environment and it is composed by parameters that define both: lines and corners (corners are defined in the Cartesian space and lines in the polar space as will be shown in section 2.2). The order in which lines and corners appear in $\hat{\xi}_{m,t}$ dependes on the moment they were detected. P_t is the covariance matrix associated with the SLAM system state; $P_{vv,t}$ is covariance of the vehicle's pose and $P_{mm,t}$ is the covariance of the map. $P_{vm,t}$ and $P^T_{mv,t}$ are cross-correlation matrices (between the vehicle and the map).

The covariance matrix initialization techniques and the EKF definition can be found in (Durrant-White & Bailey (2006a)). The EKF is represented in Eq. (3). All variables involved in the estimation process are considered as Gaussian random variables.

$$\begin{cases} \hat{\xi}_t^- = f(\hat{\xi}_t, u_t) \\ P_t^- = A_t P_{t-1} A_t^T + W_t Q_{t-1} W_t^T \\ K_t = P_t^- H_t^T (H_t P_t^- H_t^T + R_t)^{-1} \\ \hat{\xi}_t = \hat{\xi}_t^- + K_t (z_t - h(\hat{\xi}_t^-)) \\ P_t = (I - K_t H_t) P_t^- . \end{cases} \tag{3}$$

In Eq. (3), $\hat{\xi}_t^-$ is the predicted state of the system at time t; u_t is the input control commands and $\hat{\xi}_t$ is the corrected state at time t; f describes the motion of the elements of $\hat{\xi}$. P_t^- and P_t are the predicted and corrected covariance matrices respectively at time t; A_t is the Jacobian of f with respect to the SLAM system state and Q_t is the covariance matrix of the noise associated to the process, whereas W_t is its Jacobian matrix; K_t is the Kalman gain at time t; H_t is the Jacobian matrix of the measurement model (h) and R_t is the covariance matrix of the actual measurement(z_t). The term $(z_t - h(\hat{\xi}_t^-))$ is called the innovation vector (Thrun et al. (2005)) and takes place when the data association procedure has reached an appropriate matching between the observed feature and the predicted one ($h(\hat{\xi}_t^-)$). Both, the process model (f) and the observation model are non-linear expressions. Further information about the EKF-SLAM can be found in (Auat Cheein et al. (2009a)).

In this work, the sequential EKF was implemented in order to reduce computational costs. The sequential EKF-SLAM is based on the iterative calculation of the correction stage (SLAM system state and covariance matrix) for each feature with correct association –see (Thrun et al. (2005))–. The last statement implies that the Jacobian matrix of the measurement model and Kalman gain are sparse matrices, decreasing in that way the processing time during a correction iteration. Nevertheless, the prediction stage remains as stated in Eq. (3).

The general form of the correction stage of the classical sequential EKF-SLAM algorithm (Thrun et al. (2005)) is summarized in the algorithm shown in Fig. 1. Sentences (3) to (9) describe the for–loop of the correction stage of the algorithm. For every feature with correct association –sentence (2)– the for–loop is executed. Sentence (4) shows the Kalman gain calculation; sentence (5) is the correction of the SLAM system state whereas sentence (6) is the correction of the covariance matrix of the SLAM algorithm; in sentence (7), the current feature is deleted from the set of features with correct association (M_t). In the next iteration, the next predicted SLAM system state and covariance matrix are the last corrected SLAM system state and covariance matrix respectively, as noted in sentence (8).

Further information concerning the EKF-SLAM implemented in this work can be found in (Auat Cheein et al. (2010)).

2.1 Mobile Robot

The vehicle used in this work is a non-holonomic unicycle type mobile robot Pioneer 3AT build by **ActivMedia**. Figure 2 shows the kinematic model of the mobile robot. Equation (4) shows the discrete kinematic equation of the robot.

```
1: Let $N_t$ be set of the observed features
2: Let $M_t \subseteq N_t$ be the set of features with correct association
3: for $j = 1$ to $\#M_t$ do
4:     $K_{t,j} = P^-_{t,j} H^T_{t,j} (H_{t,j} P^-_{t,j} H^T_{t,j} + R_{t,j})^{-1}$
5:     $\hat{\xi}_{t,j} = \hat{\xi}^-_{t,j} + K_{t,j}(z_j - h(\hat{\xi}^-_{t,j}))$
6:     $P_{t,j} = (I - K_{t,j} H_{t,j})) P^-_{t,j}$
7:     $M_{t,j} = M_{t,j} - \{z_j\}$
8:     $P^-_{t,j} := P_{t,j};\ \hat{\xi}^-_{t,j} = \hat{\xi}_{t,j}$
9: end for
```

Figure 1: Algorithm of the correction stage of the Sequential EKF-SLAM.

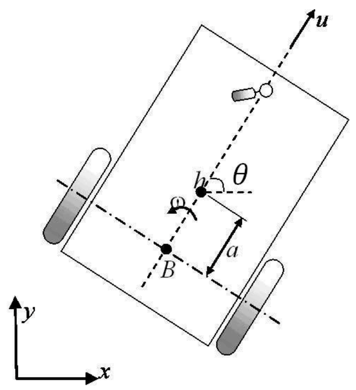


Figure 2: Graphic representation of the kinematic model of the unicycle mobile robot.

$$\begin{bmatrix} \xi_{x,t} \\ \xi_{y,t} \\ \xi_{\theta,t} \end{bmatrix}_G = \begin{bmatrix} \xi_{x,t-1} \\ \xi_{y,t-1} \\ \xi_{\theta,t-1} \end{bmatrix} + \Delta t \begin{bmatrix} \cos(\xi_{\theta,t-1}) & -a\sin(\xi_{\theta,t-1}) \\ \sin(\xi_{\theta,t-1}) & a\cos(\xi_{\theta,t-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_t \\ \omega_t \end{bmatrix} + \Phi_t \tag{4}$$

In Eq. (4), $\xi_{x,t}$, $\xi_{y,t}$ and $\xi_{\theta,t}$ are the coordinates of h –the point of control of the mobile robot– in Fig. 2; Φ_t is the discrete time Gaussian process noise associated with the robot's model; u_t is the linear velocity command applied to the vehicle and ω_t is the angular velocity command; Δt is the sampling time of the system. The suffix G in Eq. (4) refers that the coordinates of the mobile robot are expressed in a global reference frame of the environment.

2.2 Features of the Environment

The models of the features of the environment –corners and lines– are shown in Eqs. 5 and 6. Figure 3 shows the graphical interpretation of the variables in Eqs. 5 and 6. Although all features are extracted from a local reference frame attached to the mobile robot, they are added to the SLAM system state once their parameters are converted to the global reference frame of the system (Durrant-Whyte & Bailey (2006a)).

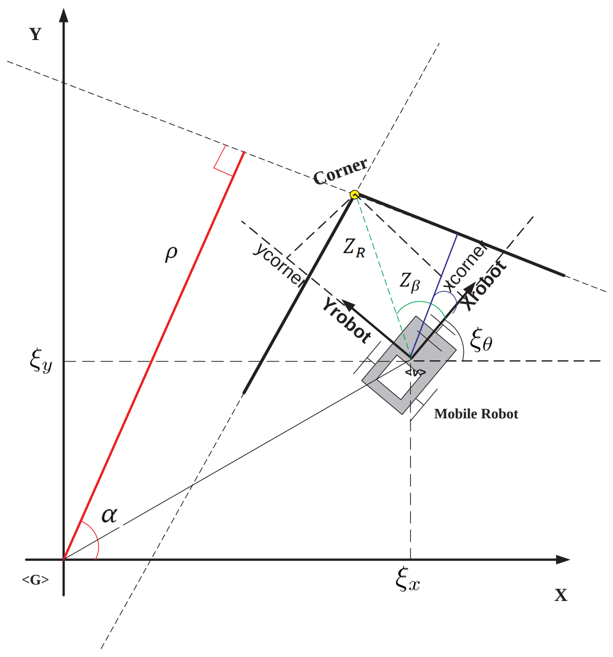


Figure 3: Graphic representation of line and corners extracted from the environment.

$$z_{corner}(k) = h_i[\hat{\xi}_{v,t}, w(k)] = \begin{bmatrix} z_R \\ z_\beta \end{bmatrix} =$$

$$\begin{bmatrix} \sqrt{(\hat{\xi}_{x,t}(k) - x_{corner})^2 + (\hat{\xi}_{x,t}(k) - y_{corner})^2} \\ \arctan \frac{\hat{\xi}_{y,t}(k) - y_{corner}}{\hat{\xi}_{x,t} - x_{corner}} - \hat{\xi}_{\theta,t}(k) \end{bmatrix} + \tag{5}$$

$$+ \begin{bmatrix} w_R \\ w_\beta \end{bmatrix}$$

$$z_{line}(k) = h_i[\hat{\xi}_{v,t}, w(k)] =$$

$$\begin{bmatrix} Z_\rho \\ Z_\alpha \end{bmatrix} = \begin{bmatrix} r - \hat{\xi}_{x,t}(k)\cos(\alpha) - \hat{\xi}_{v,t}(k)\sin(\alpha) \\ \alpha - \hat{\xi}_{\theta,t}(k) \end{bmatrix} \tag{6}$$

$$+ \begin{bmatrix} w_\rho \\ w_\alpha \end{bmatrix}$$

In Eqs. (5) and (6), $w_\rho, w_\alpha, w_R, w_\beta$ are additive Gaussian noise associated with the measurement. Further information concerning the line's modeling can be found in (Garulli et al. (2005)). The corners' extraction method can be seen in (Guivant & Nebot (2001)).

3 SLAM-Control Algorithm: implementation issues

The combination of a control strategy –for navigation purposes– within the SLAM algorithm provides the mobile robot with full autonomy (Thrun et al. (2005)). The control strategy can be divided into two: reactive and non-reactive –or planned– control.

The reactive control strategy is usually implemented on a behavior-based case strategy –or sensor based navigation–. The mobile robot does not need its pose information to plan its movements beyond a local reference frame attached to the vehicle (Arkin (1998)) –which is usually

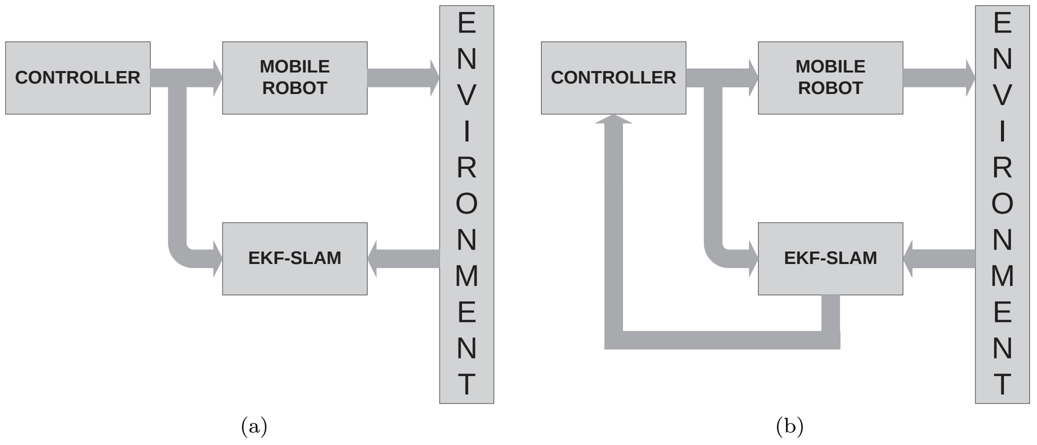


Figure 4: Schematic of the SLAM algorithm combined with control. a) Shows the SLAM with reactive control, the SLAM does not return any information to the controller; b) shows the SLAM with non reactive control, the controller uses the robot's pose estimated by the SLAM algorithm in order to updated the control references.

accomplished with odometry measurements due to the fact that for small distances, the variance of the odometry can be discarded (Thrun et al. (2005)) –. Thus, there is no communication between the SLAM and the control strategy. This kind of SLAM-Control architectures are developed for exploration tasks. Whether the SLAM becomes inconsistent or not, does not affect the exploration. The SLAM-Reactive Control strategy is represented in Fig. 4a.

On the other hand, when the SLAM is combined with a planned-based control strategy –such as trajectory or path following controller–, the control needs the robot's current pose in order to calculate the references and the control actions. Thus, for a stable control law if the SLAM turns into inconsistency, the control will remain stable, although the objective of the navigation will not be fulfilled because of erroneous references –if the SLAM turns into inconsistency, then the estimation is no longer reliable–. Furthermore, due to the non-constant time of the SLAM's executions (Thrun et al. (2005)), the stability of the controller can also be compromised (the controller must be stable for non-constant sampling time). Thus, not any controller can be implemented within a SLAM algorithm. As it can be seen, the SLAM algorithm with non-reactive controllers is more sensitive to the SLAM performance than with reactive controllers. Figure 4b shows the general architecture of an autonomous SLAM with non-reactive control strategy.

Beyond the control stability issues, the fact of the SLAM algorithm being not constant time also compromises the mobile robot integrity. Considering that all inner parameters of the robot are set up every sampling time –i.e., if the control input changes at a time between time t and $t + t_1$, it will be effective only in time $t + t_1$–, let us suppose the following scenario: the SLAM algorithm is executed at time t and it ends at time $t + \Delta$, with $\Delta > 0$. In an implementation of the SLAM-Control algorithm, if during the interval $[t, t + \Delta]$ there was a change in the motion commands, they will not be effective until the next sampling time following $t + \Delta$ (let us call it $t + t_k$, where $t_k \geq \Delta$). The last sentence implies that during the Δ-time, the robot was in an open loop situation. Furthermore, after the $t + \Delta$ time, the estimation of the pose will correspond to the pose that the vehicle had at instant t and that information is then passed to the controller. Thus, the controller at time instant $t + t_k$ will have the references of instant t. The last statement also implies a limitation in the velocity of the robot: if the robot is navigating in an open loop situation, it could collide. In order to avoid this open loop situation, a predictor is used in this work. The following section shows the implementation of a control architecture encompassing a predictor, SLAM, and a reactive controller. By using a predictor during the open loop situation, there will be available information concerning the current pose of the vehicle in the corresponding SLAM calculations.

A last remark is necessary. If the SLAM-Control algorithm is implemented as a sequential algorithm, the control commands can be changed only at the next sampling time after the last SLAM execution.

3.1 Sequential Algorithm Structure

The time charts shown in Fig. 5 shows the sequential implementation of the SLAM algorithm. As recommended by (Durrant-Whyte & Bailey (2006b)), before the mobile robot's initial motion, it extracts the first features from the environment. In this way, once the navigation cycle is closed, the final covariance matrix of the SLAM system state will tend to its initial values.

Figure 5a shows an SLAM-Control implementation without a predictor whereas Fig. 5b shows the sequential implementation of the SLAM-Control with a predictor. As it can be seen in Fig. 5, a cycle of the SLAM-Control algorithm is composed by three stages: Features Extraction–SLAM–Pause –in that order–. Thus, after the robot's initialization, it extracts the features from the environment. After the features extraction, the SLAM is performed. Both, the corrected SLAM system state and covariance matrix are used to calculate the control commands of the mobile robot. Then, in order to update the mobile robot parameters, a pause is needed between the SLAM ending and the following sampling time of the robotic system. During this pause, the robot continues its motion according to its previous command controls.

Regarding the SLAM-Control implementation, let us consider Fig. 5a and let Δ be the sampling time of the system –Δ not necessary constant–. Thus,

(i) After the initialization of the system, the control commands of the mobile robot are set up to zero: $[u\ \omega]^T = [0\ 0]^T$ and they are effective after the first pause, that is, at instant time t_1.
(ii) From instant time t_1 up to t_2, the control command remains unchanged.
(iii) At time $t_1 + t_\alpha$, with $\alpha \leq \Delta$, a new features extraction and SLAM process are completed. Based on the information provided by the SLAM system state and its covariance matrix, the next mobile robot control commands are generated ($[u\ \omega]^T = [v_1\ \omega_1]^T$), where they will be effective only at time t_2 (because of the necessary pause of the system) and maintained up to time t_3. Let us recall that the SLAM system state and covariance matrix estimation at time $t_1 + t_\alpha$ corresponds to the robot's pose at time t_1.
(iv) At time t_2, the mobile robot motion is set up to $[u\ \omega]^T = [u_1\ \omega_1]^T$. A features extraction and a SLAM execution are processed.
(v) At time $t_2 + t_\beta$, with $t_\beta \leq \Delta$, a new estimation of the system state is available from the SLAM algorithm and new control commands are generated based on that information ($[v\ \omega]^T = [u_2\ \omega_2]^T$). The new control commands will only be effective at t_3, meanwhile, the current control commands correspond to $[u\ \omega]^T = [u_1\ \omega_1]^T$.
(vi) The process repeats successively.

Thus, in the chart shown in Fig. 5a, none predictor is used. Due to the fact that the control commands are generated based on a past value of the mobile robot's pose, this situation could lead to inconsistence of the map, as will be shown in section 3.2. This is so because the covariance matrix associated with the control errors –Q_t in Eq. (3)– cannot absorb the errors of the robot's pose as the SLAM algorithm execution time increase.

In the chart shown in Fig. 5b, the sequential SLAM-Control implementation is presented. In this chart, a prediction stage is used in order to compensate the lack of robot's pose information present in the chart of Fig. 5a. The SLAM-Control functionality is similar to the one presented before, with the following differences:

(i) At time $t_2 + t_\beta$, the control commands –$[u_2\ \omega_2]$– are generated based on the information provided by the SLAM system state and its covariance matrix and the predicted pose of the mobile robot with the current control commands ($[u\ \omega]^T = [u_1\ \omega_1]^T$). Thus, the predicted pose concerns the elapsed time between t_2 and $t_2 + t_\beta$. The control commands $[u_2\ \omega_2]$ will be effective at time instant t_3.

(ii) The control commands at time $t_3 + t_\gamma$ are generated based on the information provided by the SLAM system state and its covariance matrix and the prediction of the mobile robot's pose at time $t_3 + t_\gamma$. The prediction is formed by the elapsed time of the pause of the previous SLAM-Control execution (it is the time between $t_2 + t_\beta$ and t_3) and time consumed by the features extraction and the SLAM processes (from t_3 to $t_3 + t_\gamma$). As it can be seen, from $t_2 + t_\beta$ to t_3, the mobile robot motion was governed by $[u\ \omega]^T = [u_1\ \omega_1]^T$; whereas from time t_3 to $t_3 + t_\gamma$, the motion commands were $[u\ \omega]^T = [u_2\ \omega_2]^T$. Thus, a prediction for both cases is needed.

(iii) The process repeats successively.

As it can be seen, the control commands in the sequential SLAM-Control implementation are only refreshed after the features extraction and the SLAM process' endings. The predictor used in this work is presented in Eq. (7) which is the same that the one used by the EKF-SLAM algorithm (Auat Cheein et al. (2009a)).

$$\begin{bmatrix} \hat{\xi}_{x,t} \\ \hat{\xi}_{y,t} \\ \hat{\xi}_{\theta,t} \end{bmatrix}_G = \begin{bmatrix} \hat{\xi}_{x,t-1} \\ \hat{\xi}_{y,t-1} \\ \hat{\xi}_{\theta,t-1} \end{bmatrix} + \Delta t \begin{bmatrix} \cos(\hat{\xi}_{\theta,t-1}) & -a\sin(\hat{\xi}_{\theta,t-1}) \\ \sin(\hat{\xi}_{\theta,t-1}) & a\cos(\hat{\xi}_{\theta,t-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_t \\ \omega_t \end{bmatrix} \tag{7}$$

In Eq. (7), Δt is the elapsed time used for the prediction. Thus, for the case shown in the chart of Fig. 5b, that time could be either t_β, $t_3 - (t_2 + t_\beta)$ or t_γ. For example, for $\Delta t = t_\beta$, the prediction equation would be of the form shown in Eq. (8).

$$\begin{bmatrix} \hat{\xi}_{x,t_2+t_\beta} \\ \hat{\xi}_{y,t_2+t_\beta} \\ \hat{\xi}_{\theta,t_2+t_\beta} \end{bmatrix}_G = \begin{bmatrix} \hat{\xi}_{x,t_2} \\ \hat{\xi}_{y,t_2} \\ \hat{\xi}_{\theta,t_2} \end{bmatrix} + t_\beta \begin{bmatrix} \cos(\hat{\xi}_{\theta,t_2}) & -a\sin(\hat{\xi}_{\theta,t_2}) \\ \sin(\hat{\xi}_{\theta,t_2}) & a\cos(\hat{\xi}_{\theta,t_2}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ \omega_1 \end{bmatrix} \tag{8}$$

3.2 Experimental Comparisons

For purposes of showing the differences of using a predictor in a SLAM-Control implementation, some real time experimentations were carried out at the facilities of the *Instuto de Automatica*, National University of San Juan, Argentina. The mobile robot used was a Pioneer 3AT –introduced in section 2.1–; the features extracted from the environment were lines –associated with walls– and corners –see section 2.2–; the range sensor used was a laser built by **SICK**, which acquires 181 measurements in a range of 32 meters from 0 to 181 degrees; the EKF-SLAM implemented was presented in section 2. No odometry information of the robot was included in the estimation process.

Considering that the objective is to show the advantages of using a predictor within the SLAM-Control algorithm, the navigation strategy was a simple trajectory following based on frontier points determination (Auat Cheein et al. (2009a)). Briefly, the frontier points strategy finds non-occupied points of the navigable space at the limit of the sensor range and directs the robot's movements to that point by generating a kinematically plausible path from the robots position to the frontier point. Also, it is possible to find several frontier points by varying the range of the sensor. Figure 6 shows how successive frontier points are determined. Once the plausible path is found, a kinematic controller drives the robot motion through that path until the robot reaches the frontier point. Once the robot reaches a neighborhood of the frontier point, a new frontier point is determined. This situation repeats until the navigation is interrupted. The kinematic controller implemented in these experiments is the Kanayama's trajectory follower (Kanayama et al. (1990)).

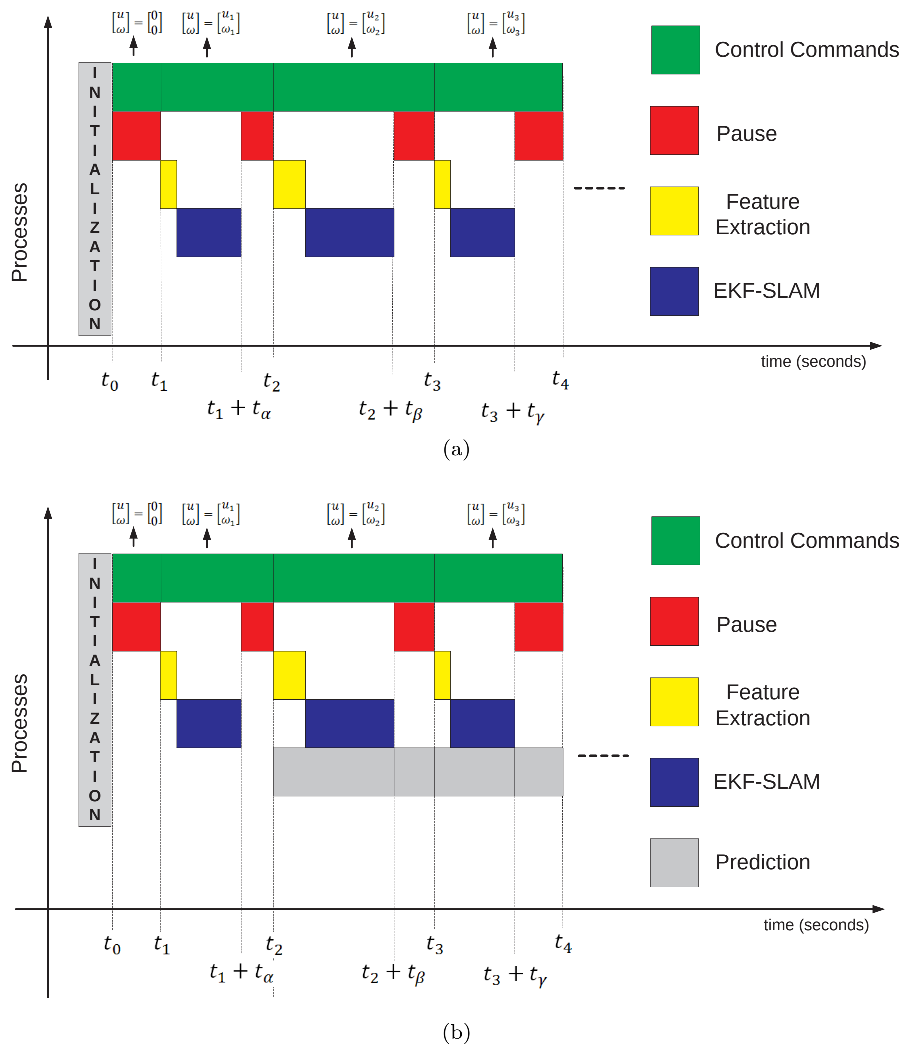


Figure 5: Chart time of the SLAM-Control implementation. a) Shows the SLAM-Control implementation without a predictor; b) shows the chart time of a SLAM-Control implementation with a predictor.

Figure 7 shows the SLAM-Control implementation using the predictor to improve the open loop situation of the navigation during the control planning process. On the other hand, Fig. 8 shows two different snapshots at times t_a and t_b of a same experiment of SLAM-Control algorithm without the predictor. As it can be seen, the map reconstruction starts to become inconsistent at position $[x\ y]^T = [12\ 25]^T$. A third snapshot after t_b is not presented because the estimation is lost due to the inconsistence and no map can be recovered.

4 Autonomous SLAM: an Uncertainty Maps Navigation example

In this section we show an autonomous driven SLAM example. The mobile robot, while performing the SLAM algorithm, will use information contained within the SLAM system state and its covariance matrix to generate uncertainty maps of the environment. These maps are then used to determine uncertainty zones and the robot is driven to these zones by means of a non-reactive controller. Thus, the vehicle navigates the environment based on the SLAM information which is recursively updated during navigation.

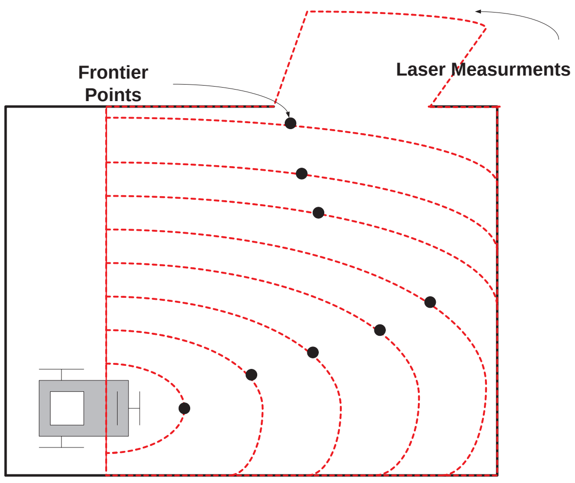


Figure 6: Frontier points determination. The frontier points are determined by localizing middle free space points from the measurement data.

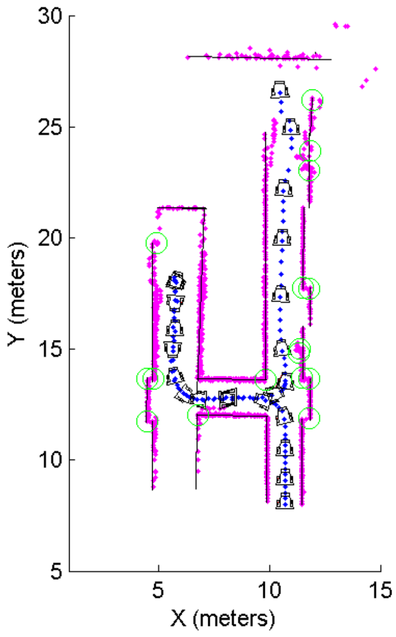
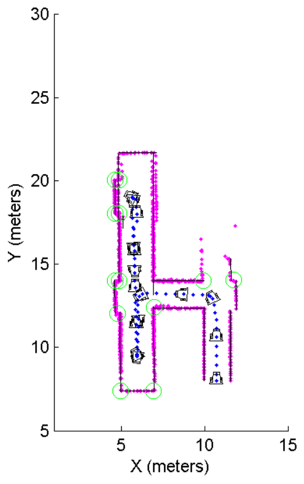
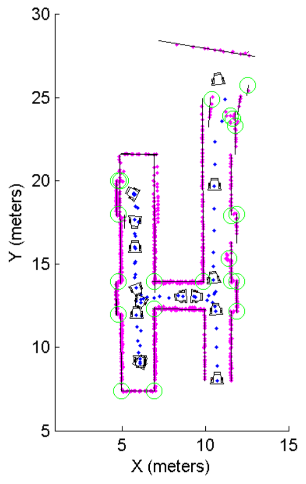


Figure 7: Map reconstruction of the facilities of the *Instituto de Automatica*. The magenta points are raw laser data; the solid black segments represent walls from the environment; green circles are corners and the blue dotted line is the path traveled by the mobile robot.

For the purpose of obtaining uncertainty maps of the environment, a Monte Carlo points generation combined with the *sum of Gaussians* method for the uncertainty maps construction is proposed in this work. The fact of using the Monte Carlo points generation combined with the *sum of Gaussians* method substantially decreases the computational costs when compared with a map gridding procedure (Thrun et al. (2005); Sanchez Miralles & Sanz Bobi (2004)).



(a) snapshot for time t_a



(b) snapshot for time t_b

Figure 8: Two snapshots of a SLAM-Control implementation without a predictor for the planning purpose. Fig. 8a shows the map reconstruction for instant time $t_a < t_b$. On the other hand, Fig. 8b shows that, at position $[x\ y]^T = [25\ 12]^T$, the map starts to differ from the one shown in Fig. 7. The magenta points are raw laser data; dotted blue line is the path traveled by the mobile robot; segments associated with walls from the environment are represented by solid black lines and green circles are corners –convex and concave–.

4.1 Monte Carlo Uncertainty Maps Building Procedure

The construction procedure of the uncertainty maps can be summarized as follows.

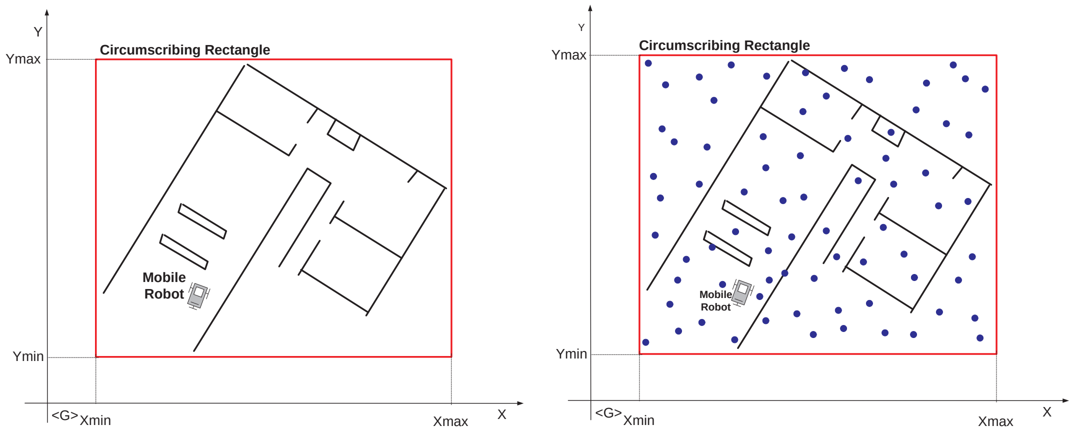
(i) The geometrical representation of the map built by the SLAM and the robot's pose are circumscribed by a rectangle. The rectangle is composed by four edges which are considered as virtual features of the environment.

(ii) The four virtual features are parameterized as it is shown in Eq. (6) and they will be considered as Gaussian distribution functions with probability value of 0.5 at their mean values. By the linear nature of the EKF, the real features of the environment are also Gaussian distributions with mean values given by the SLAM system state and covariance matrix extracted from the SLAM system state covariance matrix. The probability function distribution of the segments of the environment will be explained in detail in section 4.1.4.

(iii) Within the space circumscribed by the rectangle, M points are uniformly generated by the Monte Carlo method, covering the mapped space –see Fig. 9–.

(iv) From the set of M Monte Carlo points, only the navigable points (represented by the set N, where $N \subseteq M$) will be used in the uncertainty points determination.

(v) Each element of N –the set of navigable points– has a probability value associated with it. That probability is the result of several probability distribution functions associated with each feature of the environment contained within the SLAM system state. Those points whose probability values are near zero, will be considered as free space points; those points with probability near one, will be considered as occupied points of the environment (e.g., points near walls) and those points which probability is in a neighborhood of 0.5 will be considered as *uncertainty points*, because no conclusions can be made about the point being occupied or unoccupied. In order to fusion the probability value of a point w.r.t. the features of the environment, the *sum of Gaussians* method is performed.



(a) (b)

Figure 9: Monte Carlo points generation procedure. a) The mapped area and the robot's pose are circumscribed by the rectangle –red solid line–; b) over the circumscribed area, M Monte Carlo points are generated –blue dots–.

(vi) The final map is a representation of the probability values associated with the navigable points of the mapped environment.

The following subsections will show the Monte Carlo uncertainty maps construction in detail.

4.1.1 Monte Carlo Points Generation

In order to present the Monte Carlo points generation, let us suppose the map shown in Fig. 9a with the circumscribing rectangle in solid red. The mobile robot is positioned within the environment. Furthermore, let X_{max} and Y_{max} be the maximum values circumscribed by the rectangle and X_{min} and Y_{min} the corresponding minimum values referenced to the global reference system used by the SLAM algorithm. Then, the Monte Carlo generation points can be expressed as stated in Eqs. (9) and (10).

$$\begin{cases} \mu_i \sim U(\lambda_0, \lambda_1), \;\; i = 1, ..., M \\ m_i = X_{min} + (X_{max} - X_{min}) \times \mu_i \end{cases} \tag{9}$$

$$\begin{cases} \mu_j \sim U(\lambda_0, \lambda_1), \;\; j = 1, ..., M \\ m_j = Y_{min} + (Y_{max} - Y_{min}) \times \mu_j \end{cases} \tag{10}$$

In Eq. (9), μ_i is the i^{th}–outcome of the Monte Carlo experiment ($U(\lambda_0, \lambda_1)$ means an *uniform distribution* with parameters $\lambda_0 = 0$ and $\lambda_1 = 1$) from a set of M possible points; m_i is the x–coordinate (bounded by X_{max} and X_{min}). Equivalent definitions apply for Eq. (10), except that m_j is the y–coordinate (bounded by Y_{max} and Y_{min}). Thus, the Monte Carlo point generated over the circumscribed space shown in Fig. 9a is expressed as: $[x\ y]^T_{<G>} = [m_i\ m_j]^T_{<G>}$. Figure 9b shows the Monte Carlo generated points for the map shown in Fig. 9a.

4.1.2 Navigable Points Determination

A possible way to check if a point of a map is navigable or not, is to verify that there is an obstacle-free path between the robot and that point. Although several works can be found in the scientific literature to check if a point is navigable (Arkin (1998)), in this work a metric map-based path generation was implemented. This algorithm consists on generating an obstacle-free path having into account the metric information available in the stored map of the environment.

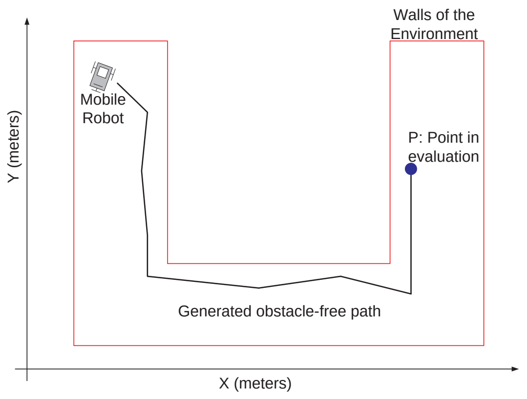


Figure 10: Navigable points determination. If a free-obstacle path can be determined from the mobile robot's pose to the Monte Carlo point P, then P is a navigable point.

Figure 10 shows an example of the path found by the algorithm that proves the point P within the environment can be reached. The navigable points test is performed over the set M defined above. The set $N,\ N \subseteq M$ of navigable points will be the set of points for which a free-obstacle path –starting at the robot's pose and ending at the point's location– can be found.

4.1.3 Probability of a Navigable Point with respect to a Corner

As stated before, the distribution function of any feature extracted from the environment is a Gaussian distribution of the form shown in Eq. (11), due to the Gaussianity nature of the EKF. In Eq. (11) the mean values ($[\hat{\xi}_i\ \hat{\xi}_j]^T$) are extracted directly from the EKF-SLAM system state, where $\hat{\xi}_i$ corresponds to the random variable at the i^{th}– position. The covariance matrix –Ψ_i– associated with $[\hat{\xi}_i\ \hat{\xi}_j]^T$ is extracted from the EKF-SLAM system state covariance matrix. Thus, if $\hat{\xi}_i$ and $\hat{\xi}_j$ are two consecutive variables, their covariance matrix correspond to a 2×2 principal sub-matrix located at the i^{th}–row, i^{th}–column and extended to the j^{th}–row, j^{th}–column.

Let $p_{x,y} = [p_x\ p_y]^T$ be a navigable point of the space and $[\hat{\xi}_{corner,x}\ \hat{\xi}_{corner,y}]^T$ the estimated parameters of a corner from the SLAM system state, then the probability of $p_{x,y}$ with respect to the corner can be expressed as Eq. (11).

$$P(p_{x,y}) = \frac{1}{\sqrt{(2\pi)^2|\Psi_i|}} e^{-\frac{1}{2}\left(\left[\begin{array}{c} p_x \\ p_y \end{array}\right] - \left[\begin{array}{c} \hat{\xi}_{corner,x} \\ \hat{\xi}_{corner,y} \end{array}\right]\right)^T \Psi_i^{-1} \left(\left[\begin{array}{c} p_x \\ p_y \end{array}\right] - \left[\begin{array}{c} \hat{\xi}_{corner,x} \\ \hat{\xi}_{corner,y} \end{array}\right]\right)} \tag{11}$$

Considering that in this work, a corner is modeled in Cartesian coordinates and a line is modeled in the polar coordinate –see section 2.2–, in order to express the probability of a given point w.r.t. to a feature over the same space some transformations are needed, this situation is necessary to apply the *sum of Gaussians* method. Thus, by applying the *Fundamental Probability Theorem* (Theodoris & Koutroumbas (2003)) to Eq. (11), we can obtain the probability of a navigable point $p_{x,y}$ with respect to a corner in the polar space instead of the Cartesian space. Thus, let $f(\sigma, \tau)$ be a density probability function defined in the polar coordinate system. Also, let $g_\sigma(x, y) = \sigma$ and $h_\tau(x, y) = \tau$ be two functions that relate Cartesian coordinates x and y with σ and τ from the polar coordinate system, where σ is related to the distance to the origin of the system and τ is its angle. Then,

$$f(\sigma,\tau) = \frac{f(x,y)}{|J(x,y)|}$$

$$|J(x,y)| = \begin{vmatrix} \frac{\partial\sigma}{\partial p_x} & \frac{\partial\sigma}{\partial p_y} \\ \frac{\partial\tau}{\partial p_x} & \frac{\partial\tau}{\partial p_y} \end{vmatrix} = \begin{vmatrix} \frac{\partial p_x}{\partial\sigma} & \frac{\partial p_y}{\partial\sigma} \\ \frac{\partial p_x}{\partial\tau} & \frac{\partial p_y}{\partial\tau} \end{vmatrix}^{-1}$$

where J is the Jacobbian matrix associated with the transformation. Applying the equations above to transform Eq. (11) into the polar space we have that:

$$\sigma = \sqrt{p_x^2 + p_y^2}$$

$$\tau = atan(p_y, p_x)$$

$$J = \begin{vmatrix} \cos(\tau) & -\sigma\sin(\tau) \\ \sin(\tau) & \sigma\cos(\tau) \end{vmatrix}^{-1} = [\sigma\cos^2(\tau) + \sigma\sin^2(\tau)]^{-1} = \frac{1}{\sigma}$$

$$f(\sigma,\tau) = \sigma f(p_x, p_y) = \sigma f(\sigma\cos(\tau), \sigma\sin(\tau)). \tag{12}$$

Finally, using Eq. (12), the probability of a point w.r.t. a corner of the environment in polar coordinates can be expressed as it is shown in Eq. (13).

$$P(p_{x,y})_{corner,i} = \frac{\sigma}{\sqrt{(2\pi)^2|\Psi_i|}} e^{-\frac{1}{2}\left(\begin{bmatrix} p_x \\ p_y \end{bmatrix} - \begin{bmatrix} \sigma\cos(\tau) \\ \sigma\sin(\tau) \end{bmatrix}\right)^T \Psi_i^{-1} \left(\begin{bmatrix} p_x \\ p_y \end{bmatrix} - \begin{bmatrix} \sigma\cos(\tau) \\ \sigma\sin(\tau) \end{bmatrix}\right)} \tag{13}$$

Although Eq. (13) not longer represents a Gaussian distribution due to the nonlinear transformations, quasi-Gaussianity is preserved: the hyper volume of the distribution is bounded by an elliptical frontier.

4.1.4 Probability of a Navigable Point with respect to a Line

In order to calculate the probability of navigable point w.r.t. a line of the environment some restriction must be taken into account.

The probability of a navigable point could be influenced by either a virtual or a real line. Virtual lines are those that circumscribe the map in section 4.1.1. Though the parameters of a line and its associated covariance determine the Gaussian distribution of the feature, it has no meaning outside the limits of the segment representing that line. Thus, the influence of a line is restricted to those points that belong to the segment's region. Let us recall that the segment associated with a line is available from the secondary map of the SLAM algorithm implemented in this work –see section 2–.

A segment's region is defined as follows. Let S be a segment associated with line L and P_o and P_f be its endpoints in the considered region. let L_o and L_f be two lines normal to L that pass through P_o and P_f respectively. A point P belongs to a region of S if it belongs to the area delimited by L_o and L_f that contains S. Figure 11 shows a representation of the region of a segment.

Thus, those points that belong to the region of a segment are probabilistically influenced by the line that contains that segment in the SLAM system state.

By considering that a line is represented in polar coordinates by a point in the polar plane, any other point represented in the polar space will imply line at the Cartesian space. According to this, to calculate the probability of a point with respect to a line is necessary to determine to which line the point belongs. Having into account that a line-feature has a Gaussian distribution function, its maximum probability value occurs on their mean values. Thus, those values that make null the kernel of the Gaussian distribution represent the points where the probability reaches its

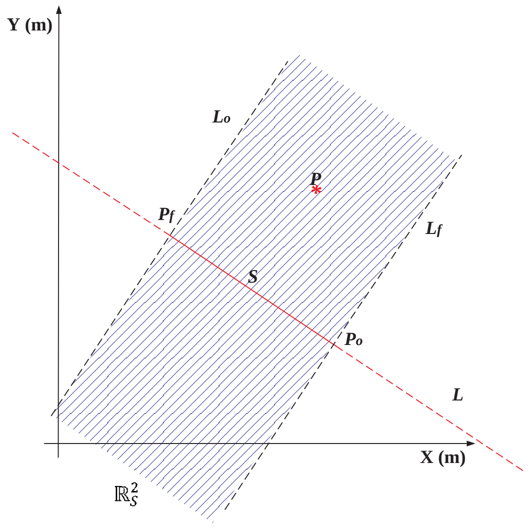


Figure 11: Region of a segment. A navigable point is influenced by the probability distribution associated with a line if that point belongs to the region of the segment of the line.

maximum. In this work it was adopted that the line to which the point of the environment belongs is a parallel of the line-feature of the map. Thus, the angle between them will be null in the kernel of the Gaussian distribution and only the distance will be computed. Equation (14) shows the probability of a point with respect to a line when the point belongs to a parallel of the line-feature.

$$P(p_{x,y})_{line,k} = \frac{1}{\sqrt{(2\pi)^2|\Psi_k|}} e^{-\frac{1}{2}(\Upsilon-\Gamma)^T\Psi_k^{-1}(\Upsilon-\Gamma)} \tag{14}$$

In Eq. (14), Ψ_k is the covariance of the line extracted from the covariance matrix of the SLAM system state, $\Upsilon^T = [\sigma\ \tau]^T$ are the parameters of the line extracted from the SLAM system state. If the line were virtual –e.g., the lines that contain the edges of the circumscribing rectangle in section 4.1–, then $\Upsilon_{virtual}$ contains the parameters that define the line and its covariance could be of the form $\Psi_{virtual} = \begin{bmatrix} 0.32 & 0.0 \\ 0.0 & 0.32 \end{bmatrix}$. Note that $\Psi_{virtual}$ is not unique. $\Gamma^T = [\sigma_p\ \tau_p]^T$ represents the parameters of the parallel line that contains the navigable point $p_{x,y}$. As stated before, $\sigma_p = \sigma$ in Eq. (14) because the lines are parallel.

4.1.5 Weighted sum of Gaussians method

To calculate the probability of a navigational point within the environment, it is necessary to calculate how all the mapped features are influencing it. Thus, considering that all features mapped from the environment have a Gaussian distribution function, in this work the *weighted sum of Gaussians* method was used in order to infer the probability associated with a navigable point. The weighted sum of Gaussians method is faster when compared with other fusion methods (Miralles & Sanz Bobi (2004)) and could be applied for both: real and virtual features. The resulting probability is always smaller than one. Let L be the number of features contained in $\hat{\xi}_{m,t}$, the part of the SLAM system state composed by the extracted features from the environment –see Eq. (1)–. Also, let L_c and L_l the number of mapped corners and lines respectively such that $L = L_c + L_l$. Furthermore, let $L_{l,v}$ be the number of virtual lines. The virtual lines correspond to the segments that circumscribe the environment. Equation (15) shows the sum of Gaussians implementation having into account both: real and virtual features.

$$
\begin{aligned}
&P(p_{x,y}) = \\
&\textstyle\sum_{k=1}^{L_c} \frac{\varepsilon_{c,k}\sigma_{c,k}}{\sqrt{(2\pi)^2|\Psi_{c,k}|}} e^{-\frac{1}{2}\left(\left[\begin{array}{c} p_x \\ p_y \end{array}\right] - \left[\begin{array}{c} \sigma_{c,k}\cos(\tau_{c,k}) \\ \sigma_{c,k}\sin(\tau_{c,k}) \end{array}\right]\right)^T \Psi_{c,k}^{-1} \left(\left[\begin{array}{c} p_x \\ p_y \end{array}\right] - \left[\begin{array}{c} \sigma_{c,k}\cos(\tau_{c,k}) \\ \sigma_{c,k}\sin(\tau_{c,k}) \end{array}\right]\right)} + \\
&+ \textstyle\sum_{k=1}^{L_l} \frac{\varepsilon_{l,k}}{\sqrt{(2\pi)^2|\Psi_{l,k}|}} e^{-\frac{1}{2}(\Upsilon_{l,k}-\Gamma_{l,k})^T \Psi_{l,k}^{-1} (\Upsilon_{l,k}-\Gamma_{l,k})} + \\
&+ \textstyle\sum_{k=1}^{L_{l,v}} \frac{\varepsilon_{l,v,k}}{\sqrt{(2\pi)^2|\Psi_{l,v,k}|}} e^{-\frac{1}{2}(\Upsilon_{l,v,k}-\Gamma_{l,v,k})^T \Psi_{l,v,k}^{-1} (\Upsilon_{l,v,k}-\Gamma_{l,v,k})}
\end{aligned}
\qquad (15)
$$

In Eq. (15), $\Psi_{c,k}$, $\Psi_{l,k}$ and $\Psi_{l,v,k}$ are the covariance matrices associated with the k^{th}–corner, line and virtual line respectively; $\varepsilon_{c,k}$, $\varepsilon_{l,k}$ and $\varepsilon_{l,v,k}$ are the weights associated with each term of the sum of Gaussians method such that the result, $P(p_{x,y})$, is always smaller than one. The weight factors in Eq. (15) were obtained according to (Miralles & Sanz Bobi (2004)). The mechanism shown in sections 4.1.3 to 4.1.5 allows the estimation of an occupational probability value of each navigable point of the map obtained in section 4.1.2.

4.1.6 Monte Carlo Uncertainty Maps examples

Figure 12 shows three examples of the construction of uncertainty maps based on the procedure presented above. Figures 12a and 12d are partial reconstructions of the facilities of the *Instituto de Automatica* of the National University of San Juan; and Fig. 12g is an office environment of the Engineering Department of the Federal University of Espirito Santo, Brazil. Figures 12a, 12d and 12g show the reconstruction of the map based on the SLAM system state and covariance matrix. The solid black segments correspond to walls associated with lines; green circles are corners and the dotted blue line is path traveled by the mobile robot. The magenta points are raw data acquired by the range sensor laser incorporated on the robot.

Figures 12b, 12e and 12h shows the mapped environment –solid black lines– circumscribed by the virtual rectangle –solid red lines–. The small blue squares represent the Monte Carlo points generated within the circumscribed area. The green circles are corners of the environment. The number of generated Monte Carlo points for the three figures was of $M = 1000$.

In addition, Figs. 12c, 12f and 12i show the uncertainty maps after applying the *sum of Gaussians* method to each Monte Carlo point. Although in section 4.1.5 it was stated that the probability value is calculated over the set of navigational points of the environment, in order to see the entire probability map, the probability value in Figs. 12b, 12e and 12h was calculated for all the Monte Carlo points. As it can be seen, the green areas in such figures represent the uncertainty areas that will be used as navigation goals (if the uncertainty point associated with them is a navigational point). In this work, an uncertainty point was considered as such if its probability value was $P(p_{x,y}) = 0.5 \pm 0.2$. Thus the navigational points whose probability value laid in the range of 0.5 ± 0.2 were considered as uncertainty points, because no conclusions can be made about their occupational status.

4.2 Autonomous SLAM Navigation

As stated in section 4.1, a navigable point will be considered as an uncertainty point if the probability value associated with it remains in a neighborhood of 0.5.

Once an uncertainty point is found, a trajectory is planned from the robot's pose to that point, using the information of the environment provided by the SLAM system state. Then, a trajectory follower controller drives the robot to a neighborhood of the uncertainty point goal. Considering that we are using range sensors to map the environment, the fact of reaching the actual position of the uncertainty point is not needed, because the robot is able to map the

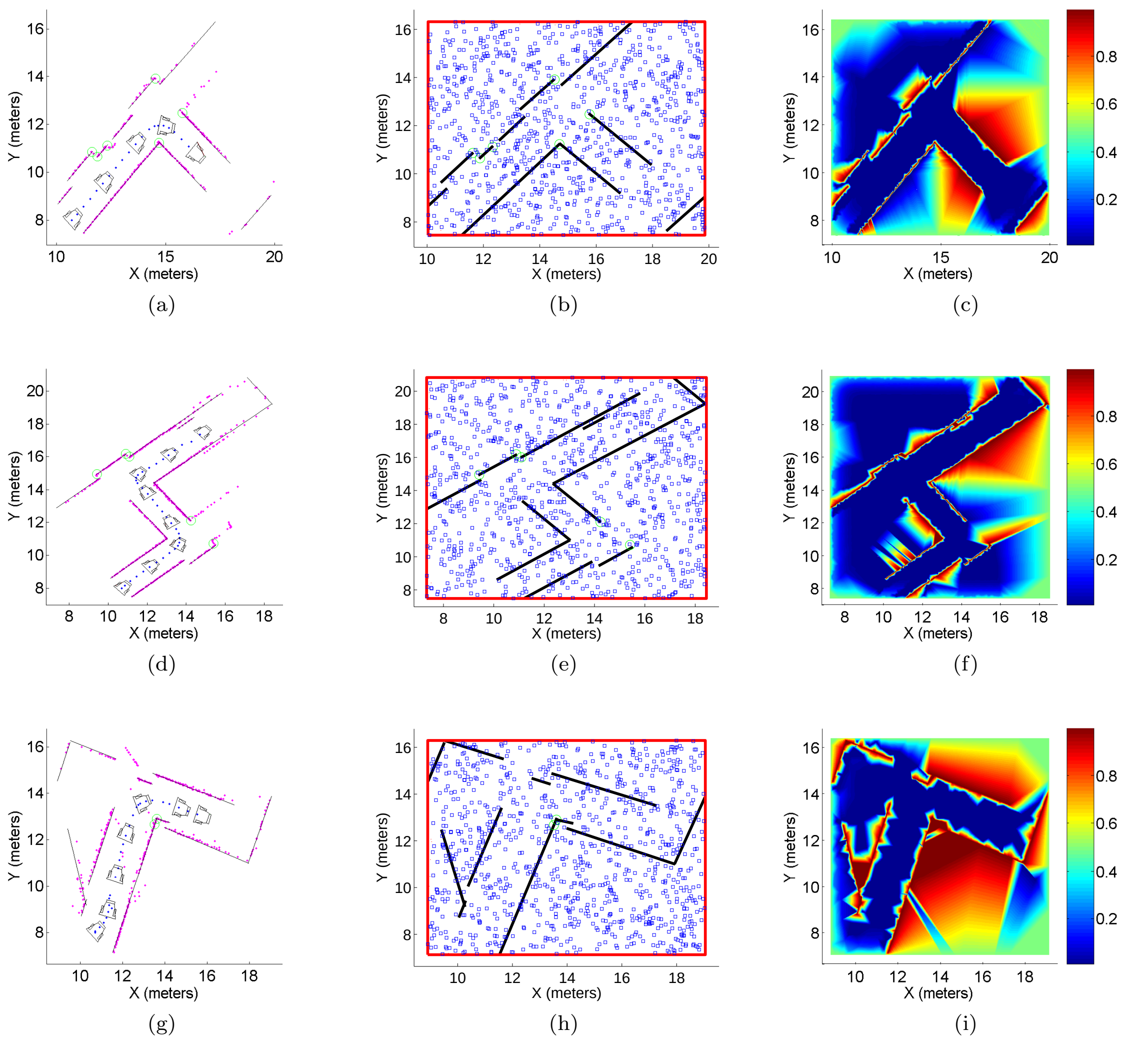


Figure 12: Examples of the construction of uncertainty maps. Figs. 12a, 12d and 12g shows three different environment reconstruction using the EKF-SLAM algorithm presented in this work; Figs. 12b, 12e and 12h show the Monte Carlo points generation after circumscribing the map obtained by the EKF-SLAM by the virtual rectangle –solid red segments–. Figs. 12b, 12e and 12h correspond to Figs. 12a, 12d and 12g respectively. Figs. 12c, 12f and 12i show the uncertainty maps after applying the *sum of Gaussians method* to each Monte Carlo point of Figs. 12b, 12e and 12h.

environment surrounding that point by means of the range sensor. During the navigation, the SLAM algorithm is continuously executed. The trajectory controller implemented in this work is the Kanayama's controller (Kanayama et al. (1990)).

If more than one uncertainty point is found, then a selection criterion is used to determine which point will be the goal of the navigation. Equation (16) shows the goal selection criterion; where $dist(.)$ represents the distance between two points and $P(p_{x,y})$ is the probability associated to point $p_{x,y}$ according to Eq. (15). Thus, for n uncertainty points, only the one which has the maximum ratio between its probability value and its distance from the robot will be chosen as a navigation goal. Further information can be found in a previous work of the authors (Auat Cheein

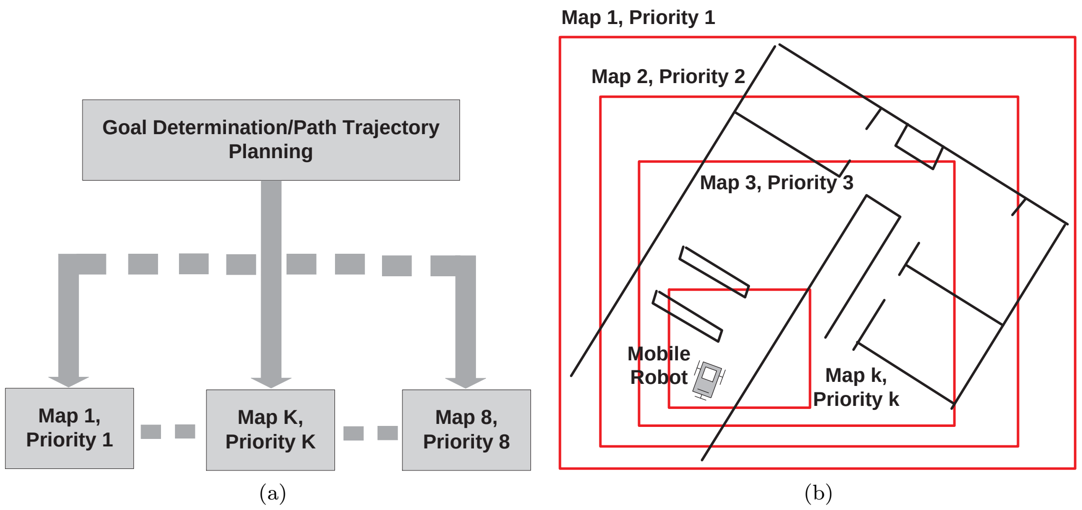


Figure 13: Parallel implementation of the multi-level uncertainty point searching. a) Shows the parallel structure of the implementation; b) shows the multi-level division: each level has a priority associated with it.

et al. (2010)); κ_j is a priority value associated with the uncertainty point, as will be explained below.

$$Goal\ Navigation\ Uncertainty\ Point = \frac{1}{\kappa_j}\ max\left\{\frac{P(p_{x,y|_i})}{dist(robot_{pose}, p_{x,y|_i})}\right\};\ i = 1...n. \qquad (16)$$

Considering that the size of the map increases during the navigation, the uncertainty point determination is made in several levels with different priorities associated with each level. Figure 13 shows the multilevel uncertainty points searching. Each level is implemented in a different thread, therefore, all levels work in parallel. For example, for the map shown in Fig. 13b, the main map is divided into k levels. Each level has a virtual rectangle circumscribing it. All the virtual rectangles have the same orientation with respect to the main map. The priority assignment is as follows: the main map has the higher priority –priority $\kappa = 1$– and the k-level has priority $\kappa = k$, where k is an integer. Thus, when the navigation system is looking for a goal, the uncertainty points within all the different levels are searched. An uncertainty point with priority 1, for example, is chosen when compared with an uncertainty point of priority 3. The uncertainty points searching of level $k-1$ does not include the virtual features of level k, it only works with the real features of the environment and its own virtual circumscribing rectangle. The navigation strategy based on the multi-level uncertainty point searching is shown in the following section.

4.2.1 Mobile Robot Navigation Strategy

The navigation strategy can be summarized as follows.

(i) Let us suppose that the robot initial pose is $[\hat{\xi}_x\ \hat{\xi}_y\ \hat{\xi}_\theta]^T = [\hat{\xi}_{x,0}\ \hat{\xi}_{y,0}\ \hat{\xi}_{\theta,0}]^T$ referenced to a global reference frame previously established during the initial conditions declaration of the SLAM algorithm. The mobile robot performs, without moving, the first features extraction procedure of the navigation.

(ii) A first circumscribing rectangle is determined. This rectangle must circumscribe all extracted features from the environment and the robot's pose. If no features were detected, then a generic size circumscribing rectangle including the robot's pose can be used instead.

(iii) M Monte Carlo points are uniformly generated within the circumscribing rectangle. Then, from the M Monte Carlo points, only the $N \subseteq M$ navigable points are used.
(iv) A probability value is associated with each navigable point. In order to do this, the *sum of Gaussians* method is used. The sum of Gaussians method has into account the probability of the point with respect to any feature –real or virtual– of the environment. A real feature is a feature that is modeled by the SLAM system state and its covariance matrix. As stated before, the edges of the circumscribing rectangle will be considered as Gaussian virtual features with probability value of 0.5 associated to its parameters.
(v) A navigable point will be considered as an uncertainty point if its probability value is in a neighborhood of 0.5. From the set of $L \subseteq N \subseteq M$ uncertainty points, only one will be chosen as navigation goal according to Eq. (16).
(vi) Once an uncertainty point is found, a trajectory is planned from the robot's current pose to the uncertainty point. Then, the Kanayama's trajectory controller drives the robot to a neighborhood of that point.
(vii) When the robot reaches a neighborhood of the uncertainty point, a new uncertainty point should be determined from the information within the SLAM system state and its covariance matrix by means of the procedure mentioned above.
(viii)If any edge of the virtual rectangle that circumscribes the mapped area is bigger than the range of the sensor used, then the map is organized in several nested levels as it is shown in Fig. 13b. Up to 8 different levels are allowed in our work. All the levels are equally placed from the robot's pose to the main map.
(ix) The system searches for uncertainty points within all the levels as it is shown in Fig. 13. Considering that not all levels will found an uncertainty point goal at the same time, the navigation strategy chooses a low priority level uncertainty point according to Eq. (16), until a high priority level uncertainty point is found.
(x) Once a high priority level uncertainty point is found, a trajectory is planned from the robot's current pose to the uncertainty point and the trajectory controller drives the robot motion to it.
(xi) Once the robot reaches an uncertainty point, the searching strategy is repeated only in those levels that are not currently working.
(x) If after a finite number of attempts and despite of the path planned, the robot is not able to reach the navigation goal, then that goal is replaced by a new uncertainty point destination.

The navigation strategy presented above can also be interpreted as follows: *the mobile robot remains navigating based on local uncertainty points until global uncertainty points are found, then the robot is driven to them.* Figure 14 shows the final SLAM-Control architecture implemented in this work. As Fig. 14a shows, the SLAM algorithm provides the robot's pose references to the controller and the map information to the trajectory planner. Figure 14b shows how the priorities of each level of uncertainty maps are managed by the navigation system; the multiplexer, according to the priority level, chooses between the different uncertainty maps.

4.2.2 Non-reactive Controllers: Trajectory follower controller

The non-reactive controller implemented in this work is the Kanayama's trajectory controller for non-holonomic vehicles (Kanayama et al. (1990)). This is an asymptotically stable control law whose stability was proved through Lyapunov theory. The inputs to the vehicle controller are the reference posture $[x_r\ y_r\ \theta_r]^T$ and the reference velocities $[V_r\ W_r]^T$.

The posture error is defined as follows:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos(\hat{\xi}_{\theta,t}) & \sin(\hat{\xi}_{\theta,t}) & 0 \\ -\sin(\hat{\xi}_{\theta,t}) & \cos(\hat{\xi}_{\theta,t}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - \hat{\xi}_{x,t} \\ y_r - \hat{\xi}_{y,t} \\ \theta_r - \hat{\xi}_{\theta,t} \end{bmatrix} \tag{17}$$

where $[\hat{\xi}_{x,t}\ \hat{\xi}_{y,t}\ \hat{\xi}_{\theta,t}]^T$ is the current estimated pose of the vehicle.

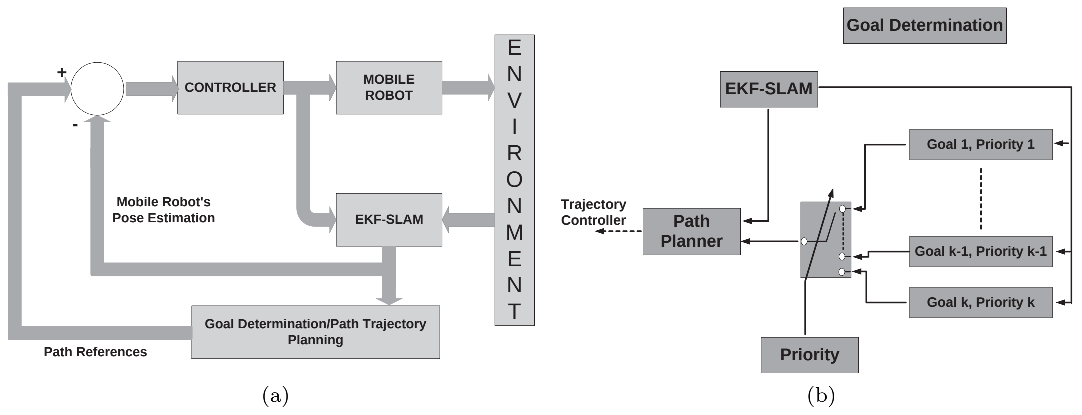


Figure 14: SLAM-Control architecture. a) The SLAM algorithm closes the loop providing the robot's pose information to the controller and the environment information to the trajectory planner; b) the different levels of the uncertainty maps are combined by a multiplexer governed by the priority values of each navigation goals.

The control law is:

$$\begin{bmatrix} V \\ W \end{bmatrix} = \begin{bmatrix} V_r \cos\theta_e + K_x x_e \\ W_r + V_r(K_y y_e + K_\theta \sin\theta_e) \end{bmatrix} \tag{18}$$

where K_x, K_y and K_θ are positive constants.

Kanayama also proposes a parameter selection to obtain a critical damping in the control. The damping of the tracking control can be calculated through

$$\zeta = \frac{K_\theta}{2\sqrt{K_y}}.$$

A critical damping in the control is obtained when $\zeta = 1$.

The path associated with the trajectory was previously obtained by implementing the non-optimum path planning method presented by (Nieto et al. (2010)). Then, with the references positions of the path and the estimated pose of the vehicle by the SLAM system state, the trajectory follower was implemented according to Eq. (18).

5 Experimental Results

Several real time experiments of the SLAM-Control proposal presented in section 4.2 are shown in this section. The maximum SLAM sampling time was of 0.2 seconds. The last implies that the *pause*, the features extraction and the SLAM execution in Fig. 5b were performed within a maximum time of 0.2 seconds during the navigation of the mobile robot. In addition, the features association criterion used in this work corresponds to the Mahalanobis distance (Guivant & Nebot (2001);Thrun et al. (2005)). Figure 15 shows results of the SLAM-based map reconstruction for a simulated office environment whereas Fig. 16 shows different snapshots of the experiments carried out at the facilities of the *Instituto de Automatica* of the National University of San Juan.

The mobile robot used was the non-holonomic unicycle type Pioneer 3AT (built by **ActivMedia**) with a range sensor laser (built by **SICK**) incorporated on it. The laser acquires 181 range measurements between 0 to 180 degrees. The measurement range was set to 7 meters although the laser can be set up to 32 meters. No odometry information was used in the SLAM algorithm.

For the uncertainty maps construction, the maximum number of Monte Carlo points for the level with the smallest priority (which is the closest level to the local reference frame of the mobile robot) was of $M = 1000$ –although M is adjustable–. The maximum number of levels of

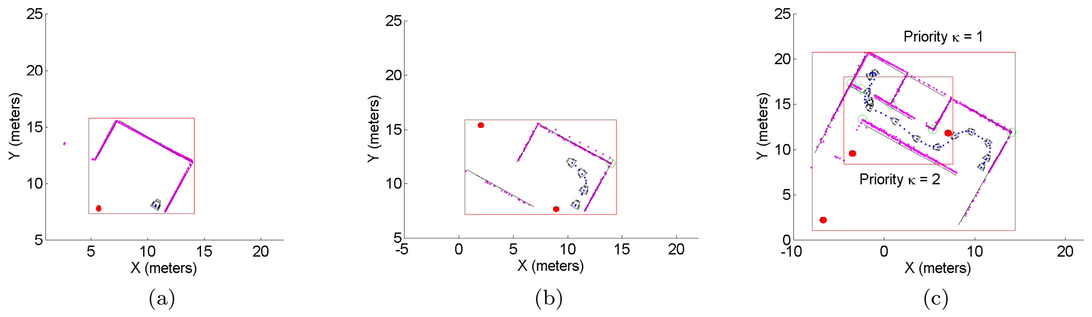


Figure 15: Real time simulation of the navigation strategy. This figure shows the map reconstruction and the navigation of the mobile robot within a simulated office environment. The solid red lines are the rectangles that circumscribe the environment; the magenta points are raw laser data; the solid black lines are segments associated with lines and green circles represent corners; the solid red circles represent the center of mass of a cloud of uncertainty points over that region. a) Shows the first execution of the SLAM algorithm while the robot remains still; b) Shows the second execution of the SLAM; c) after several executions and given the size of the mapped area, the uncertainty points procedure requires two levels of searching.

uncertainty points searching paralleled implemented in this work was of eight levels. The number of Monte Carlo points for each level was a linear function of the priority associated with the corresponding level of the uncertainty map. Thus, for an uncertainty map with priority $\kappa = 8$, $M = 1000$ –as stated before– and for an uncertainty map with priority $\kappa = 1$, $M = 8 * 1000 = 8000$ points. In Figs. 15 and 16, the uncertainty points –that could be used as possible destination goals– are represented by a solid red circles. These solid red circles represent the center of mass of a cloud of uncertainty points located in the corresponding region.

In addition, Fig. 17 shows the behavior of the standard deviation associated with the position of the mobile robot and with the parameters of several features during the SLAM-Control executions shown in the real time experiment of Fig. 16. As it can be seen, the standard deviation remains bounded during the estimation of the SLAM system state.

6 Conclusions

This paper has presented the advantages and methods of introducing a predictor to compensate the no-constant time executions of a sequential EKF-SLAM when implemented with a non-reactive controller, such as a path or a trajectory follower. The EKF-SLAM algorithm used in this work was a feature-based SLAM that extracts corners and lines from the environment. In order to show real time implementations of the combination of the SLAM algorithm with non-reactive controllers, an uncertainty maps construction based on the Monte Carlo method was shown. The construction of uncertainty maps has allowed the determination of unvisited regions –called *uncertainty points*– of the environment. A trajectory controller has driven the vehicle from its current pose to the uncertainty points.

The Monte Carlo method to build uncertainty maps has introduced a new method to generate a probabilistic map of the environment considering all features as Gaussian distributions without the need of gridding the map. The method uses the Gaussianity condition of the features of the environment acquired during the SLAM algorithm execution.

The navigation strategy presented in this work, has searched for uncertainty points within different levels of the mapped environment. Each level had an uncertainty map with a priority associated with it. Levels closer to the local reference frame of the mobile robot had a priority smaller than the levels closer to the global reference frame of the vehicle. Once an uncertainty

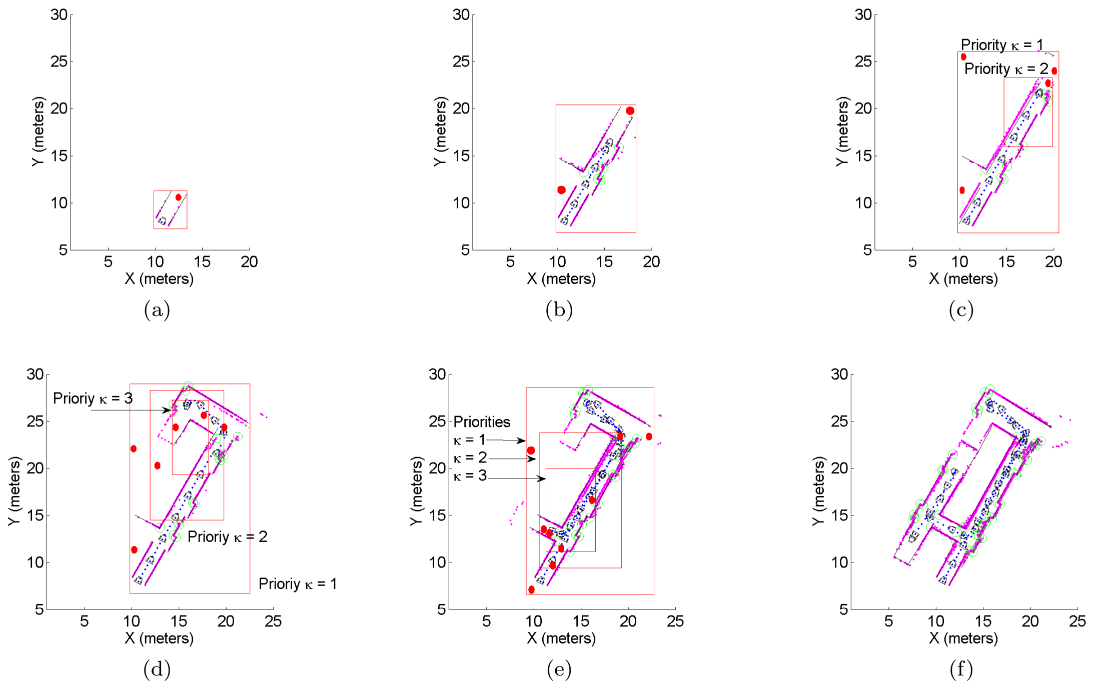


Figure 16: Real time experimentation of the navigation strategy within a real environment. The solid red lines represent the edges of the circumscribing rectangle; solid black lines are walls associated with lines of the environment; dotted blue line is the path traveled by the mobile robot; green circles are corners and solid red circles are the center of mass of a cloud of uncertainty points over that region; magenta points are raw laser data. Figures 16a to 16f show different snapshots of the evolution of the mobile robot navigation within the facilities of the *Instituto de Automatica* of the National University of San Juan. a) The first features from the environment are acquired and added to the SLAM system state while the mobile robot remains still; b) shows another searching of uncertainty points using a single level map; c) given the dimensions of the current map, the uncertainty points are searched within two levels; d) and e) the system requires three priority levels of uncertainty points searching; f) shows the complete map of the environment.

point was found, the trajectory controller had driven the robot to a neighborhood of that point. During the navigation and the construction process, the SLAM algorithm was continuously executed.

Experimental results about the probabilistic map construction were also shown. The entire system was implemented in real time showing its autonomy and performance when mapping and navigating unknown environments. The SLAM algorithm and the navigation strategy were implemented to build a geometric map of the environment, although they were restricted to structured ones. For future works, the SLAM algorithm and the navigation strategy will be implemented to operate in open and semi-structured environments.

Acknowledgement

The authors would like to thank to CONICET-Argentina and FCT-Portugal for partially founding this research.

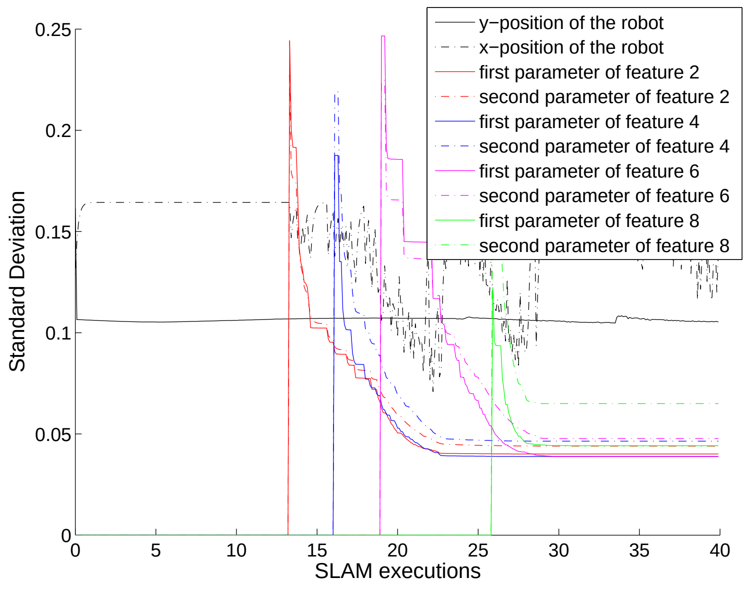


Figure 17: Standard deviation evolution of the robot's position and feature's parameters during the SLAM algorithm executions of the experiment shown in Fig. 16. The standard deviation remains bounded.

References

Andrade-Cetto, J. & Sanfeliu, A. 2006 *Environment Learning for Indoors Mobile Robots*, Springer Tracks in Advanced Robotics, vol. 23, Springer.

Andrade-Cetto, J. & Sanfeliu, A. 2002 *Concurrent Map Building and Localization with Landmark Validation*, in International Journal of Pattern Recognition and Artificial Intelligence, 16(3), pp. 361-374.

Arkin, R. C. 1998 *Behavior-based Robotics*, MIT Press: Cambridge, MA, USA.

Auat Cheein, F., De La Cruz, C., Carelli, R. & Bastos-Filho, T.F. 2009a *Solution to a door crossing problem for an autonomous wheelchair*, in Proc. of the International Conference on Intelligent Robots and Systems, pp.4931-4936, St. Louis, USA.

Auat Cheein, F. A., Scaglia, G., di Sciascio, F. & Carelli, R. 2009b *Feature Selection Algorithm for Real Time EKF-SLAM Algorithm*, in International Journal of Advanced Robotic Systems, vol. 36 (3), pp. 229-238, 2009.

Auat Cheein, F., Toibero, J. M., Lobo Pereira, F., di Sciascio, F. & Carelli, R. 2010 *Monte Carlo Uncertainty Maps-based for Mobile Robot Autonomous SLAM Navigation*, in Proc. of the IEEE International Conference on Industrial Technology (IEEE-ICIT), pp. 1413-1418, Via del Mar, Chile.

Ayache, N & Faugeras, O. 1989 *Maintaining a representation of the environment of a mobile robot*, IEEE Transactions on Robotics and Automation, vol. 5, no. 6, pp. 804-819.

Bailey, T., Nieto, J., Guivant, J., Stevens, M. & Nebot, E. 2006 *Consistency of the EKF-SLAM algorithm*, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3562-3568, Beijing, China.

Castellanos, J. A., Neira, J. & Tardos, J. D. 2004 *Limits to the consistency of EKF-based SLAM*, in 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles.

Chatila, R & Laumond, J.P. 1985 *Position referencing and consistent world modeling for mobile robots*, Proceedings of IEEE International Conference on Robotics and Automation, pp. 138-145, St. Louis, USA.

Choset, H. & Nagatani, K. 2001 *Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization*, in IEEE Transactions on Robotics and Automation, vol 17, n. 2, pp. 125-137.

Dellaert, F., Fox, D. & Burgard, W. 1999 *Monte Carlo localization for mobile robots*, Proceedings of IEEE International Conference on Robotics and Automation, pp. 1322-1328, Detroit, USA.

di Marco, M., Garulli, A., Lacroix, S. & Vicino, A. 2000 *A Set Theoretic Approach to the Simultaneous Localization and Map Building Problem*, in Proc. of the IEEE Conference on Decision and Control, Sydney, Australia.

Diosi, A. & Kleeman, L. 2005 *Laser scan matching in polar coordinates with application to SLAM*, in Proc. of the International Conference on Intelligent Robots and Systems, (IROS 2005), pp. 3317-3322.

Dissanayake, G., Newman, P., Clark, S., Durrant-Whyte, H. F. & Csorba, M. 2001 *A solution to the simultaneous localisation and map building (SLAM) problem*, IEEE Trans. Robotics and Automation, vol. 17, pp. 229-241.

Durrant-Whyte, H. & Bailey, T. 2006a *Simultaneous localization and mapping (SLAM): part I essential algorithms*, IEEE Robotics and Automation Mag., vol. 13, No. 2, pp. 99-108.

Durrant-Whyte, H. & Bailey, T. 2006b *Simultaneous localization and mapping (SLAM): part II state of the art*, IEEE Robotics and Automation Magazine, vol. 13, No. 3, pp. 108-117.

Garulli, A., Giannitrapani, A., Rosi, A. & Vicino, A. 2005 *Mobile robot SLAM for line-based environment representation*, in Proceedings of the 44th IEEE Conference on Decision and Control, pp. 2041-2046, Espain.

Guivant, J. E. & Nebot, E. M. 2001 *Optimization of the simultaneous localization and map-building algorithm for real-time implementation* IEEE Transactions on Robotics and Automation, vol. 17, pp. 242-257.

Hhnel, D., Fox, D., Burgard, W. & Thrun, S. 2003 *A highly efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements*, in Proc. of the IEEE Conference on Intelligent Robots and Systems (IROS).

Huang, G. P., Mourikis, A. I. & Roumeliotis, S. I. 2008 *Analysis and Improvement of the Consistency of Extended Kalman Filter based SLAM*, in Proc. of the IEEE International Conference on Robotics and Automation (ICRA).

Kanayama, Y., Kimura, Y., Miyazaki, F. & Noguchi, T. 1990 *A stable tracking control method for an autonomous mobile robot*, in Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pp. 384389, Cincinnati, OH, USA.

Kouzoubov, K. & Austin, D. 2004 *Hybrid Topological/Metric Approach to SLAM*, in Proc. of the IEEE International Conference on Robotics and Automation, vol.1, no. 1, pp. 872- 877.

Liu, Y., Dong, J. & Sun, F. 2008 *An Efficient Navigation Strategy for Mobile Robots with Uncertainty Estimation*, in Proc. of the World Congress on Intelligent Control and Automation, Chongqing, China.

Mullane, J., Ba-ngu, V., Adams, M. D. & Wijesona, W. S. 2008 *A random set formulation for Bayesian SLAM*, in Proc. of the IEEE International Conference on Intelligent Robots and Systems (IROS).

Nieto, J., Slawinski, E., Mut, V. & Wagner, B. 2010 *Online Path Planning Based on Rapidly-Exploring Random Trees*, in in Proc. of the IEEE International Conference on Industrial Technology (IEEE-ICIT), pp. 1431-1436, Via del Mar, Chile.

Sanchez Miralles, A. & Sanz Bobi, M. A. 2004 *Global Path Planning in Gaussian Probabilistic Maps*, Journal of Intelligent and Robotic Systems, vol. 40, pp. 89-112.

Sim, R. & Roy, N. 2005 *Global a-optimal robot exploration in slam*, in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain.

Smith, R., Self, M. & Cheeseman, P. 1987 *A stochastic map for uncertain spatial relationships*, Autonomous Mobile Robots: Perception, Mapping and Navigation, pp. 323-330.

Smith, R., Self, M. & Cheeseman, P. 1990 *Estimating uncertain spatial relationships in robotics*, Autonomous Robot Vehicles, pp. 167-193.

Theodoridis, S. & Koutroumbas, K. 2003 *Pattern Recognition*, Elsevier.

Thrun, S., Burgard, W. & Fox, D. 1998 *A probabilistic approach to concurrent mapping and localization for mobile robots*, Machine Learning, vol. 31, no. 1-3, pp. 29-53.

Thrun, S., Burgard, W. & Fox, D. 2005 *Probabilistic Robotics*, MIT Press, Cambridge.

Xi, B., Guo, R., Sun, F. & Huang, Y. 2008 *Simulation Research for Active Simultaneous Localization and Mapping Based on Extended Kalman Filter*, in Proc. of the IEEE International Conference on Automation and Logistics, Quindao, China.

Zunino, G. & Chrinstensen, H. I. 2001 *Simultaneous Localization and Mapping in Domestic Environments*, in Proc. of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems.