

Integrating Segments and Edges in Feature-based SLAM

D. Rodriguez-Losada

F. Matia

UPM-DISAM

José Gutierrez Abascal 2. E-28006, Madrid (SPAIN)

drodri@etsii.upm.es matia@etsii.upm.es

Abstract

In this paper a new framework for integrating edge information into segments in feature based SLAM, with minimum system overloading, is presented. All operations required for building and maintaining this map are described and formulated. A whole implementation of this approach has been programmed, optimized and successfully tested in several indoor environments with different mobile robots.

1. Introduction

Stochastic mapping with an Extended Kalman Filter (EKF) is a common approach to the Simultaneous Localization And Mapping problem SLAM [6]. The seminal paper by Smith *et al*[1] defined the way to handle uncertain location vectors when using geometric features of the environment as map elements. Their operators are used in this paper. Many implementations use the segment [7] or the line as the main kind of feature, and some of them use corners or edges modeled as points.

While a lot of recent research focus on improving the computational complexity [5] well known to be $O(n^2)$, few approaches deal with geometric uncertainty partiality. The SPmap [2] approach, used in [7] too, models the features location uncertainty with error vectors expressed in the feature coordinate frame, so this partiality is easily handled. The same idea is used in this paper too (fig.1, eqs 1, 2).

An efficient integration between segments and edges has been developed using only 1 extra dimension for each edge instead of the common 2D point representation, so computational complexity is reduced down to 44%.

The main goal of the global project is the development of an interactive robot for trade fairs [3], [4]. Extracting and efficiently handling the maximum available information at each scan is very important when operating in highly dynamic environments.

2. Segments with edges map

Every feature is defined by the estimation of its state vector \hat{x}_F and an error vector \vec{e}_F attached to its relative coordinate frame. A robot is estimated just by its location vector \hat{r}_R . The composition \odot of the robot estimation with its error to obtain the actual robot state (eq.1, fig 1) is done with the composition operator \oplus described in [1].

$$\begin{aligned} \bar{x}_R = [\bar{r}_R] &= [r_R^x \ r_R^y \ r_R^\theta]^T ; \vec{e}_R = [e_R^x \ e_R^y \ e_R^\theta]^T \\ \bar{x}_R = \hat{x}_R \odot \vec{e}_R &\Leftrightarrow \bar{r}_R = \hat{r}_R \oplus \vec{e}_R \end{aligned} \quad (1)$$

An oriented segment 'S' is estimated by the reference frame of its midpoint \hat{r}_S , its length \hat{L}_S , and two flags α_S^l , α_S^r that indicate if the respective edge (l=left, r=right) has been detected, i.e. the segment extreme point is an actual limit instead of an incomplete observation of it. The error vector \vec{e}_S of a segment is decoupled (eq.2) in vector \vec{e}_S^t for translational (transverse and angular) corrections and variable size vector \vec{e}_S^e for edges corrections. The key idea is that only one extra dimension per edge is needed in the error vector, and only if that actual edge is detected. The auxiliary variable β (eq.3) models the left (negative x) and right (positive x) edge direction and δ models the variable size vector \vec{e}_S^e (where the void \emptyset element has the same mathematical properties than the 0). With this δ variable all the equations are formulated, so covariance matrices, Jacobians, and measurement vectors are kept at minimum size with minimum computational complexity.

$$\begin{aligned} \bar{x}_S &= [\bar{r}_S \ L_S \ \alpha_S^l \ \alpha_S^r]^T \\ \vec{e}_S &= \begin{cases} \vec{e}_S^e = [\delta_S^l e_S^l \ \delta_S^r e_S^r]^T \\ \vec{e}_S^t = [e_S^y \ e_S^\theta]^T \end{cases} \\ \bar{x}_S = \hat{x}_S \odot \vec{e}_S &\Leftrightarrow \begin{cases} \bar{r}_S = \hat{r}_S \oplus \begin{bmatrix} 0 \\ \vec{e}_S^t \end{bmatrix} \oplus \begin{bmatrix} \frac{1}{2} \sum_{d=l,r} \alpha_S^d e_S^d \\ 0 \\ 0 \end{bmatrix} \\ L_S = \hat{L}_S + \sum_{d=l,r} \beta_S^d \alpha_S^d e_S^d \end{cases} \end{aligned} \quad (2)$$

$$\forall d = l \text{ (left)}, r \text{ (right)} \Rightarrow \begin{cases} \alpha_s^d = 0 \text{ (no edge)} \Rightarrow \delta_s^d = \emptyset \\ \beta_s^l = -1; \beta_s^r = 1 \end{cases} \quad \begin{cases} \alpha_s^d = 1 \text{ (edge)} \Rightarrow \delta_s^d = 1 \end{cases} \quad (3)$$

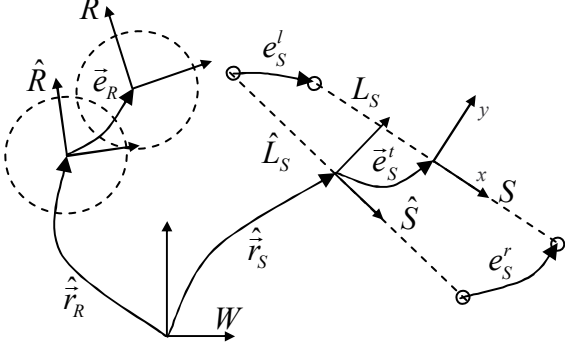


Fig. 1. Segment with edges map.

Unfortunately the actual errors are unknowns, so the error vectors are modeled as Gaussians and will be estimated by an EKF. Feature state estimations are mathematical (non stochastic) variables in the equations. The errors are always maintained centered in zero. The Map (eq.4) is defined by the state vectors of its 'n' features and the covariance matrix P of their error vectors. Typically, feature number 1 is the robot ($F_1=R$) (always exists in the map), and the rest of the features are the segments that have been observed by the robot.

$$\begin{aligned} \text{Map} &\triangleq (\hat{x}, P) & \bar{e}_{F_i} &\sim N(\hat{e}_{F_i} = \vec{0}, \text{Cov}(\bar{e}_{F_i})) \\ \hat{x} &= [\hat{x}_{F_i}] & P &= [\text{Cov}(\bar{e}_{F_i}, \bar{e}_{F_j})] \end{aligned} \quad (4)$$

3. Building the Map

3.1. Robot movement

When the robot moves an increment defined by its odometry, the robot estimation must be updated (eq. 5), as well as matrix P, as explained in [1]. Because the actual increment is unknown, its estimation and the covariance of its error are used (EKF prediction step). It is important to note that the equation that must be differentiated in the process of matrix P prediction with respect to the error vectors of the robot and the odometry, is the second one.

$$\begin{aligned} \bar{r}_{Odrom} &= \hat{r}_{Odrom} \oplus \bar{e}_{Odrom}; & \bar{e}_{Odrom} &\sim N(\hat{e}_{Odrom} = \vec{0}, \text{Cov}(\bar{e}_{Odrom})) \\ \bar{e}_{R(k+1)} &= \ominus \hat{r}_{Odrom} \oplus \bar{e}_{R(k)} \oplus \hat{r}_{Odrom} \oplus \bar{e}_{Odrom} \\ \hat{r}_{R(k+1)} &= \hat{r}_{R(k)} \oplus \hat{r}_{Odrom}; & \hat{e}_{R(k+1)}^- &= \vec{0} \end{aligned} \quad (5)$$

3.2. Data association

When the robot makes 'm' observations from its coordinate frame, the first step is to check if the i^{th} feature F_i of the map, matches the j^{th} observation O_j , computing their error-distance vector $\bar{e}_{F_i O_j}$ and its covariance matrix referenced to the F_i frame. This error vector is also variable in size depending on the edges correspondence and it is easily decoupled as shown in eq.6. B matrices are used for desired components selection. A minimum geometric feature-observation overlap is required for avoiding pairings with far collinear segments.

$$\begin{aligned} \bar{e}_{F_i O_j} &= \begin{bmatrix} \bar{e}_{F_i O_j}^e \\ \bar{e}_{F_i O_j}^t \end{bmatrix}; & \begin{cases} \bar{e}_s^d \triangleq [e_s^d \ 0 \ 0]^T; \bar{\lambda}_s^d \triangleq [\beta_s^d \hat{L}_s / 2 \ 0 \ 0]^T \\ B^t \triangleq \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; B^e \triangleq (1 \ 0 \ 0) \end{cases} \\ \bar{e}_{F_i O_j}^t &= B^t \left(\ominus \begin{bmatrix} 0 \\ \bar{e}_{F_i}^t \end{bmatrix} \ominus \hat{r}_{F_i} \oplus \hat{r}_R \oplus \bar{e}_R \oplus \hat{r}_{O_j} \oplus \begin{bmatrix} 0 \\ \bar{e}_{O_j}^t \end{bmatrix} \right) \\ \bar{e}_{F_i O_j}^e &= \delta_{F_i}^d \delta_{O_j}^d B^e \left(\ominus \bar{e}_{F_i}^d \ominus \bar{\lambda}_{F_i}^d \ominus \hat{r}_{F_i} \oplus \hat{r}_R \oplus \bar{e}_R \oplus \hat{r}_{O_j} \oplus \bar{\lambda}_{O_j}^d \oplus \bar{e}_{O_j}^d \right) \end{aligned} \quad (6)$$

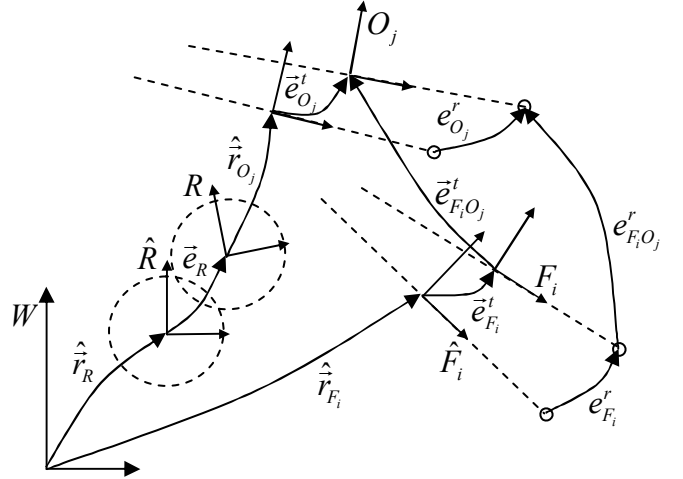


Fig. 2. Distance from observation to map feature.

This distance must be theoretically equal to zero in case of an actual pairing (eq.7), and it is used as an implicit measurement equation as in [2].

$$\begin{aligned} \hat{f}_{F_i O_j} &= \bar{e}_{F_i O_j} = \vec{0}; & \hat{f}_{F_i O_j} &= \hat{e}_{F_i O_j} = \bar{e}_{F_i O_j} \Big|_{(\bar{e}_R, \bar{e}_{F_i}, \bar{e}_{O_j} = \vec{0})} \neq \vec{0} \\ \left| \begin{aligned} H_{F_i O_j}^R &= \frac{\partial \bar{e}_{F_i O_j}}{\partial \bar{e}_R}; H_{F_i O_j}^{F_i} = \frac{\partial \bar{e}_{F_i O_j}}{\partial \bar{e}_{F_i}}; G_{F_i O_j} = \frac{\partial \bar{e}_{F_i O_j}}{\partial \bar{e}_{O_j}} \end{aligned} \right|_{(\bar{e}_R, \bar{e}_{F_i}, \bar{e}_{O_j} = \vec{0})} & (7) \\ H_{F_i O_j} &= \frac{\partial \bar{e}_{F_i O_j}}{\partial \bar{e}_{F_k}} \Big|_{k=1, \dots, n} = \begin{bmatrix} H_{F_i O_j}^R & 0 & \dots & H_{F_i O_j}^{F_i} & \dots & 0 \end{bmatrix} \end{aligned}$$

The Mahalanobis distance test is applied to this distance (eq.8). Contrary to the nearest neighbour approach, multiple pairings from a single observation are allowed as long as they satisfy the test. This is done to allow the fusion of several segments that actually match the same actual one. If one observation matches more than one feature, these features are stored in a list, so the fusing segments algorithm can be applied as explained in section 4.2.

$$\begin{aligned} C &= \text{Cov}(\hat{f}_{F_i O_j}) = H_{F_i O_j} P H_{F_i O_j}^T + G_{F_i O_j} \text{Cov}(\bar{e}_{O_j}) G_{F_i O_j}^T \\ \text{if } (\hat{f}_{F_i O_j}^T C^{-1} \hat{f}_{F_i O_j} < \mathcal{X}^2_{\dim(\hat{f}_{F_i O_j}), \alpha}) &\Rightarrow \text{Pairing accepted} \end{aligned} \quad (8)$$

In eq.8 α is the desired confidence level. If the test is passed, then the observation is paired with the feature and used in the EKF update step to improve the map estimation and for adding new edges to the feature if necessary as described in section 3.5. If not paired, the observation is added to the map as explained in section 3.4.

3.3. EKF update step.

Once all the 'q' pairings have been defined, all of them are simultaneously used in the EKF update step to improve the whole map. This approach slightly improves the robustness of the filter in case of an incorrect data association. The l^{th} pairing, which matches feature F_i with observation O_j , accumulates its measurement vector, jacobian and covariance matrices (eq.7) in rows to form the full Kalman matrices and vectors (eq.9).

$$\begin{aligned} \hat{f} &= \left[\dots \left(\hat{f}_{F_i O_j}^T \right)_l \dots \right]^T; \quad H = \left[\dots \left(H_{F_i O_j}^T \right)_l \dots \right]^T \\ G &= \text{diag} \left(\left(G_{F_i O_j} \right)_l \right); \quad R = \text{diag} \left(\text{Cov}(\bar{e}_{O_j})_l \right) \\ S &= HPH^T + GRG^T; \quad W = PH^T S^{-1} \\ \text{EKF Update Step} &\begin{cases} \hat{e}^+ = -W\hat{f} \\ P^+ = P^- - WHP^- \end{cases} \end{aligned} \quad (9)$$

When the full vector of the error estimation \hat{e}^+ is obtained, it is decomposed and applied to each feature following eq.2, so the error estimation is maintained centered in 0 as established in eq.4.

3.4. Adding new segments

The observed segments that were not paired with any feature are added to the map after the EKF update step, following eq.10. The new feature F_{n+1} obtained with the composition of the observation with the robot location

(fig.3) is appended to the map state vector. The matrix P increases its size to accommodate in its last column and row the covariances of the new error vector as described in [1], but based in the new feature error vector of eq.10.

$$\begin{aligned} \hat{x}_{F_{n+1}} &= \begin{cases} \hat{r}_{F_{n+1}} = \hat{r}_R \oplus \hat{r}_{O_j} \\ \hat{L}_{F_{n+1}} = \hat{L}_{O_j}; \alpha_{F_{n+1}}^d = \alpha_{O_j}^d \end{cases}; \quad \bar{e}_{F_{n+1}} = \begin{bmatrix} \bar{e}_{F_{n+1}}^e \\ \bar{e}_{F_{n+1}}^t \end{bmatrix} \\ \bar{e}_{F_{n+1}}^t &= B^t \left(\ominus \hat{r}_{O_j} \oplus \bar{e}_R \oplus \hat{r}_{O_j} \oplus \begin{bmatrix} 0 \\ \bar{e}_{O_j}^t \end{bmatrix} \right) \\ e_{F_{n+1}}^d &= \delta_{O_j}^d B^e \left(\ominus \bar{\lambda}_{O_j}^d \ominus \hat{r}_{O_j} \oplus \bar{e}_R \oplus \hat{r}_{O_j} \oplus \bar{\lambda}_{O_j}^d \oplus \bar{e}_{O_j}^d \right) \end{aligned} \quad (10)$$

It can be observed in fig. 3 that the error vector of the new segment doesn't depend on the position of the robot, but on its error vector.

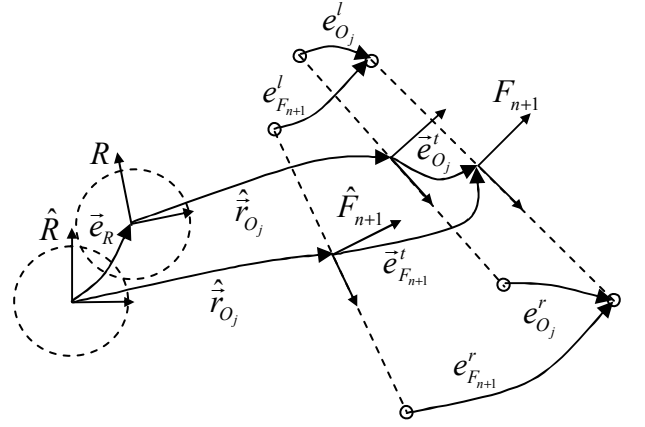


Fig. 3. Adding a new observation

3.5. Augmenting segment edges

The error vectors of the features that have been paired with observations might be augmented with new edges, after the EKF update step. If the observation has an edge not present in the feature, then the feature error vector of must be augmented to hold the new edge. The process is similar to adding new segments, but the new covariances are inserted in the corresponding matrices instead of being added at the last position. The existing error covariances remain unchanged. Eq.11 shows the error vector $\bar{e}_{F_i}^a$ for augmenting the edges of feature F_i with the edges of matched observation O_j . Noting the similarity with eq.10, fig.3 can also represent this edge augmentation.

$$\begin{aligned} \alpha_{F_i}^d &= \alpha_{F_i}^d \vee \alpha_{O_j}^d; \quad \bar{e}_{F_i}^a = \begin{bmatrix} \bar{e}_{F_i}^e \\ \bar{e}_{F_i}^t \end{bmatrix} \\ e_{F_i}^d &= \delta_{O_j}^d (!\delta_{F_i}^d) B^e \left(\ominus \bar{\lambda}_{O_j}^d \ominus \hat{r}_{O_j} \oplus \bar{e}_R \oplus \hat{r}_{O_j} \oplus \bar{\lambda}_{O_j}^d \oplus \bar{e}_{O_j}^d \right) \end{aligned} \quad (11)$$

4. Map maintenance

After the previous steps, some operations that are not inherent to the SLAM-EKF must be performed:

4.1. Geometric update of paired segments

If the paired segment hasn't both edges defined, the EKF correction is done only in the transverse, angular and one (or none) edge error, but the other (or both) extreme point of the segment must be geometrically updated. The observation extreme points are projected onto the feature. If this projection is farther than the extreme point considered, this projection is chosen to be the new extreme point. Then, the new reference frame and length of the segment are computed to update the feature state vector.

4.2. Fusing segments

Fusing segments is a important issue as it avoids an unnecessary increment on the number of features. The fusing algorithm is applied to the list of map features that were paired with the same observation. A virtual "correspondence" observation (distance) between a pair of features belonging to the list is done, and it is handled as an actual observation eq.12, so no observation uncertainty exists now. If the Mahalanobis test matches feature F_i with feature F_j , then the error (distance) between both is used in a extra EKF update step. F_i is then updated with information of F_j , and F_j is deleted from the map.

$$\begin{aligned} \hat{f}_{F_i F_j} &= \bar{e}_{F_i F_j} = \bar{0}; \quad \hat{f}_{F_i F_j} = \hat{e}_{F_i F_j} = \bar{e}_{F_i F_j} \Big|_{(\bar{e}_{F_i}, \bar{e}_{F_j} = \bar{0})} \neq \bar{0} \\ \bar{e}_{F_i F_j}^t &= B^t \left(\ominus \begin{bmatrix} 0 \\ \bar{e}_{F_i}^t \end{bmatrix} \ominus \hat{r}_{F_i} \oplus \hat{r}_{F_j} \oplus \begin{bmatrix} 0 \\ \bar{e}_{F_j}^t \end{bmatrix} \right) \\ e_{F_i F_j}^d &= \delta_{F_i}^d \delta_{F_j}^d B^e \left(\ominus \bar{e}_{F_i}^d \ominus \bar{\lambda}_{F_i}^d \ominus \hat{r}_{F_i} \oplus \hat{r}_{F_j} \oplus \bar{\lambda}_{F_j}^d \oplus \bar{e}_{F_j}^d \right) \end{aligned} \quad (12)$$

4.3. Applying parallelism

To maintain the ortogonal distribution of walls (segments) of a typical indoor environment while constructing the map, some extra virtual "parallelism" observations are done comparing each segment F_i added in the last step (because it didn't match any feature) with each

$$\begin{aligned} e_{F_i F_j}^o &= \bar{0}; \quad \hat{f}_{F_i F_j} = \hat{e}_{F_i F_j} \Big|_{(\bar{e}_{F_i}, \bar{e}_{F_j} = \bar{0})} \neq \bar{0}; \quad B^p = (0 \quad 0 \quad 1) \\ e_{F_i F_j}^p &= B^p \left(\ominus \begin{bmatrix} 0 \\ \bar{e}_{F_i}^t \end{bmatrix} \ominus \hat{r}_{F_i} \oplus \hat{r}_{F_j} \oplus \begin{bmatrix} 0 \\ \bar{e}_{F_j}^t \end{bmatrix} \right) \end{aligned} \quad (13)$$

feature F_j of the map previous to add the new segments. This is necessary if trying to close large loops and wall detection is not always absolutely continuous (the most common case).

5. System implementation

A whole implementation of this SLAM approach, suitable for any robot with odometry and a laser scanner, has been fully programmed. The software has been done in C++ with extensive use of the STL. Although the GUI for experiments is done with MFC and OpenGL for rendering, the SLAM Kernel has been done fully independent and portable. The experiments are done on a P4 2,4GHz PC with 512MB RAM, under WinXP.

5.1. Segment extraction

An algorithm for segment detection from an individual scan of good enough range sensors (i.e. SICK laser or similar) has been developed. It has four steps: 1) Splitting the scan using point sequencing to get clusters of points that are almost colinear. In this part, the minimum number of points, the minimum length, the maximum range of the sensor and the maximum tolerance are used as parameters. 2) Performing a least squares fit to get the base line. 3) Computing the left and right extreme points from the projection of the first and last points of the cluster, and then the segment reference frame and length. 4) Deciding if the extreme points of the segments are edges. This is done by analyzing the following or previous points, which should be located beyond the segment to define an edge.

The segment uncertainty depends on its length and the uncertainty of a single scan point ($\pm 5\text{cm}$ for SICK).

5.2. Code Optimization

Several optimizations have been integrated in the SLAM kernel code for improving processing speed. Symbolic solutions for most equations are directly programmed. A crude Euclidean filter is applied before every Mahalanobis test. EKF matrix calculations are highly optimized using the knowledge of the existence of block-symmetric (P), block-sparse structured (H) and block-diagonal (R, G) matrices as described in [5], but it still remains $O(n^2)$. Programming the behavior of variable δ (eq.3) requires a great effort on dynamic memory handling, but it has been completely implemented. The result of these optimizations is discussed in next section.

6. Experiments and Results

Several maps have been built from data captured in text files while manually operating a robot. Results are qualitatively analyzed because ground truth isn't available. More information at: <http://www.disam.upm.es/~drodri/>

The robot always starts at the origin with null uncertainty. The odometry uncertainty is modeled as proportional to the position increment, with an experimentally set factor of about 7% (depends on the robot). In the lab map (fig.4), the not corrected trajectory of the robot is shown. Our B21r robot with a SICK laser (running at 5Hz) has been used in experiments 1,2 and 5, driven in straight motion along the hallways, without any "looking back" behavior: "Exploration". At the end, the robot was returned to its initial physical position: "Return". Exactly the same algorithm is applied in the whole round trip. The built maps are correct and the robot always returns to the map origin (with errors < 1cm) confirming map coherence and suitability for navigation.

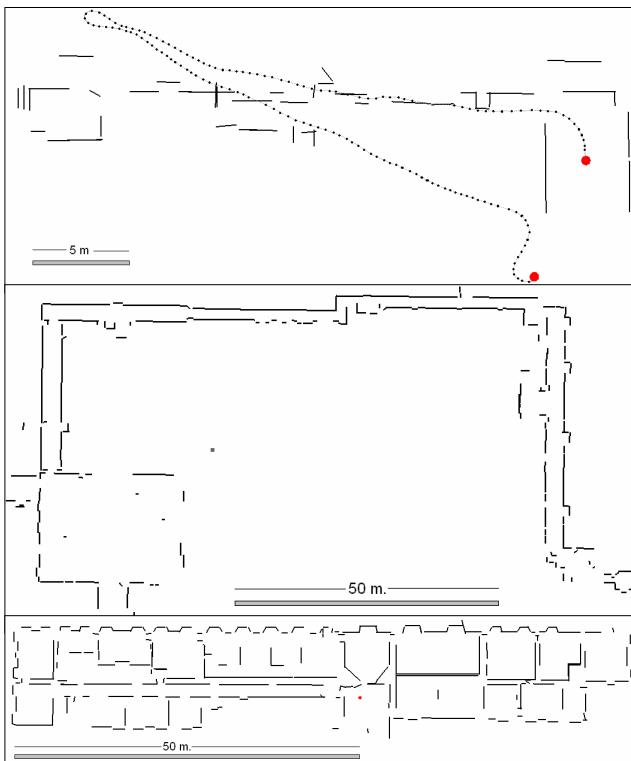


Fig.4 Lab (top), University (center) and Belgoioso (bottom) Maps

In the first 3 experiments (fig. 4), edge information is not required, as many orthogonal features exist. The lab map is done with a robot exploration of 30m during 1'27".

The 37 segments map is built in 15sec. The university ground floor is explored traveling 353m in 20'25", storing 7.6Mb of data. Although the computations of more than 170 segments can't be done in real time, the 225 features map is built in only 8'16". A 25x20m loop is successfully closed. There are people in the university hallways, so this map is correctly done in a dynamic environment.

In the 3rd experiment, Belgoioso castle (Italy) is mapped with the data (8Mb) collected by Dirk Haennel [3] with a Pioneer robot and a Sick laser. Here the laser data rate is variable, decreasing sometimes down to <1Hz. Only the laser position, laser maximum range and odometry noise parameters are changed for this experiment. The robot is driven (though this movement uses "looking back" behavior) 228m in 16'27", building a map with 238 features in 9'54". Two large loops (~100m each) are successfully closed in this experiment.

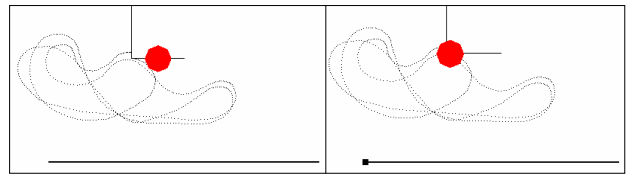


Fig. 5. Experiment 4 without edge (left), with edge (right).

In the very simple 4th experiment (fig. 5) the B21r robot is driven 25.5m in 2' in front of a single wall, with only one visible edge, returning to the initial position. The final error obtained in the wall longitudinal direction is 4cm if using the edge information, and 9.5cm if not using it. But it is more important to note that if the trajectory is repeated, the edges approach maintains this error bounded to 4 cm, while the no-edges one increases again the error in 9.5cm again, incorrectly increasing the length of the segment in 9.5cm too. Fig.5 shows the results when the trajectory is repeated 4 times.

In the 5th experiment (fig.6) the robot is driven in a 45m long hallway, where there are few transversal features. The length of the whole experiment (exploration + return) is 97m in 4'5". Some small extra noise has been added to odometry so figures are clearer. In the case of not using edges, the error in the length of the hallway is 2.33m, while the error when using edges is only 0.86m (there is still some significant error because edges are not always available). But when the robot returns, it is not able to correctly match the features if not using edges, due to large longitudinal errors. So new features are incorrectly added

and the robot does not successfully return to the origin, with a final robot position error of 2.6m. If the edges are used, the robot adequately corrects this longitudinal error, matches the features and returns home.

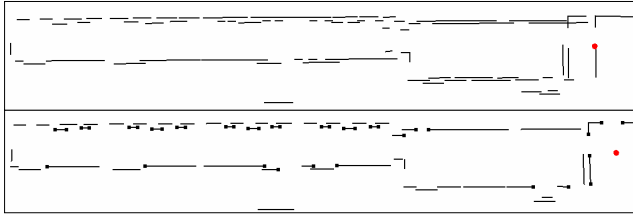


Fig. 6. Experiment 5 without edges (top), with edges (bottom).

Fig. 7 shows processing time (milliseconds) vs. the number of total error vector components of the map (P size), particularized for 8 paired errors (e.g. 4 paired segments without edges), which is a good observation. Each line shows the total time of the Kalman update step (which is the most time consuming). Line ‘a’ represents the time without edge detection neither optimizations. Line ‘b’ is the result of using optimizations without edge detection and line ‘c’ is obtained with optimizations and edges.

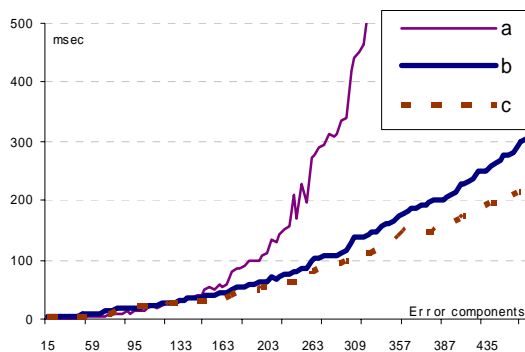


Fig. 7. Processing time.

Approximately 220 (110 segments without edges) errors can be handled in real time without optimization and 340 (170 segments without edges) with optimization. The effect of optimization increases with the number of features, for example, to map without optimization Belgoioso Castle it was employed 50’19”, 5 times more than with optimizations. When optimizing, the time used if using edges is slightly smaller than if not using edges. This is due to matrix block packing. It is important to note that the improvement achieved with the edge and segment integration is not shown here, but the computational cost is reduced down to 44% of the computational cost if using edges modeled as 2 dimensional points.

7. Conclusions and future work

The experiments show that using edges is not completely necessary in some cases, but also that it may be essential in other occasions, and it will always be useful for a more robust localization. The implementation has proved to be a computationally efficient good solution. Future enhancements will be the integration of a joint compatibility test for data association, adding some autonomous exploration behaviors, improvement of computational complexity, and the integration of the developed software with the whole navigation system.

Acknowledgment

This work is funded by Spanish Ministry of Science and Technology (URBANO: DPI2001-3652C0201) and EU 5th R&D Framework Program (WebFAIR: IST-2000-29456).

References

- [1] R. Smith, M. Self, P. Cheeseman “Estimating uncertain spatial relationships in robotics” in *Uncertainty in Artificial Intelligence 2*, J.F. Lemmer and L. N. Kanal, Eds. New York: Elsevier, 88
- [2] J.A. Castellanos, J.M.M. Montiel, J. Neira, and J.D. Tardos “The SPMAP: A Probabilistic Framework for Simultaneous Localization and Map Building” in *IEEE Trans. on Robotics and Automation*, vol. 15, n5 Oct 99.
- [3] W. Burgard, P. Trahanias, A. Argyros, H. Baltzakis, D. Haennel, M. Moors, D. Schultz, “TOURBOT and WebFAIR: Web operated robots for Tele-presence in Populated Exhibitions”, WS9 IEEE IROS’02, Laussane, Switzerland.
- [4] D. Rodriguez-Losada, F. Matia, R. Galan, A. Jiménez, “Blacky, an Interactive Mobile robot at a Trade Fair” *Proc. IEEE Int. Conf. ICRA’02*. Washington DC, USA
- [5] J. Guivant, E. Nebot, “Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation” *IEEE transactions on robotics & automation*, vol.17,n°3, june 2001 p242-257
- [6] S. Thrun. “Robotic mapping: A survey” In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.
- [7] P. Newman J. Leonard, J. D. Tardos, J. Neira “Explore and Return: Experimental Validation of Real-Time Concurrent Mapping and Localization” *Proc. ICRA02*, Washington DC. May 2002, p 1802-1809