Real-Time Stereo Visual SLAM in Large-Scale Environments based on SIFT Fingerprints

David Schleicher, Luis M. Bergasa, Rafael Barea, Elena López,
Manuel Ocaña, Jesús Nuevo, Pablo Fernández
Department of Electronics, University of Alcala, Alcalá de Henares, 28805 Madrid, Spain
dsg68818@telefonica.net, {bergasa,barea,elena,mocana,jnuevo}@depeca.uah.es

Abstract **– This paper presents a new method for real-time SLAM calculation applied to autonomous robot navigation in large-scale environments without restrictions. It is exclusively based on the visual information provided by a cheap wide-angle stereo camera. Our approach divide the global map into local sub-maps identified by the so-called SIFT fingerprint. At the sub-map level (low level SLAM), 3D sequential mapping of natural land-marks and the robot location/orientation are obtained using a top-down Bayesian method to model the dynamic behavior. A high abstraction level to reduce the global accumulated drift, keeping real-time constraints, has been added (high level SLAM). This uses a correction method based on the SIFT fingerprints taking for each sub-map. A comparison of the low SLAM level using our method and SIFT features has been carried out. Some experimental results using a real large environment are presented.**

Keywords **– SLAM, Computer Vision, Intelligent Vehicles.**

I. INTRODUCTION

Real-time Simultaneous Localization and Mapping (SLAM) is a key component in robotics. In the last years several approaches have been used [1][2]. Recent researches have demonstrated that camera-based SLAM is very useful in domains where the goal is to recover 3D camera position in real-time moving rapidly in normal human environments, based on mapping of sparse visual features, potentially with minimal information about motion dynamics [3]. In [4], a 3D visual SLAM method, based on a stereo camera and SIFT (*Scale Invariant Feature Transform*) features, is presented. Currently, the main goal in SLAM research is to apply consistent, robust and efficient methods for large-scale environments. One of the main milestones is to achieve large closing loops in robot paths running in real-time.

Several approaches can be found to solve the related issues by using *metric* methods, *topological* methods or hybrid ones. One example of the last ones is described in [5]. This solution tries to build a topological map composed by several simple metric maps. After that, as long as the robot explores new places, the algorithm decides whether to build a new sub-map or create a new *branch* to one of the already visited maps. The links contain the coordinate system relations as well as uncertainties transformations among sub-maps. In [6] they propose a 3D SLAM using SIFT similarity matrix, which are based on visual appearance. This allows the recognition of pre-visited places that can be quite repetitive. The paper presented in [7] proposes a method that is able to close a very large loop with a very high number of landmarks on a simulated environment. It uses a hierarchical method to represent the different probabilistic magnitudes associated to several map regions. Also, means to transfer the updates and predictions from the top to the bottom of the tree and vice versa are provided.

This paper presents a real-time SLAM method based on stereo-vision. The basis of this work was previously presented by the authors in [8]. The system is based on a stereo wide-angle camera mounted on a mobile robot. Several visual landmarks are sequentially captured, using the Shi and Tomasi operator (see [8]), and introduced on an EKF filter in order to model the probabilistic behavior of the system. A measurement model is used for landmarks perception and a motion model is implemented for the dynamic behavior of the robot. As it is well known, one of the major problems on the EKF implementation is the quadratic (n^2) increase of computational cost as a function of the number of landmarks, making it unsuitable for large environments where this number can be potentially high. In order to solve this problem, we present a modified SLAM implementation, which adds an additional processing level that we will call "high level SLAM" to the explained SLAM method, which

This work was supported in part by the Spanish Ministry of Science and Technology (MCyT) under Grant TRA2005-08529-C02 (MOVICON Project) and by the Community of Madrid under Grand CM: S-0505/DPI/000176 (RoboCity2030 Project).

1-4244-0830-X/07/$20.00 ©2007 IEEE.

will be known as "low level SLAM". The global map is divided into local sub-maps identified by SIFT fingerprints. A fingerprint characterize the visual appearance of an image from some SIFT visual landmarks and the relation among them. Then, the robot is locally positioned in a sub-map using the low level SLAM. The sub-map generation technique is performed based on the movement of the robot and the environment appearance. Then, a *fingerprint* is taken at the beginning, in case that the robot changes its direction, when the visual appearance of the images changes a lot, if the total reference is lose, etc. This process identifies a sub-map as the landmark positions obtained from the images taken from the current *fingerprint* to the next one. A *fingerprint* matching between previously captured fingerprints and current one is carried out to detect pre-visited zones. The information extracted from the matched SIFT features of the *fingerprint* is used to update the EKF filter and correct the robot state as well as the whole map covered by the loop.

This paper is structured into two main parts. The first one presents the low level SLAM implementation and the second one the high level SLAM. After that, a large set of results is given to show the behavior of our system. A comparison of the low SLAM level using our method and SIFT features has been carried out. The paper finishes with some conclusions and future work.

II. Low Level SLAM

This level implements all the algorithms and tasks needed to locate and map the robot on its local sub-map.

A. Extended Kalman Filter application

In order to apply the EKF, a state vector X and its covariance matrix P need to be defined. The purpose of the algorithm is to continuously estimate the position and orientation of the camera, via the linearization of the *next state function, f(X),* at each time step. Because of the *impulse motion model* used for the camera movement, which will be explained later, it is needed to add two more variables to the camera state vector X_v: the linear and angular speed: $X_v = (X_{rob} \quad q_{rob} \quad v_{rob} \quad \omega)^T$. On this equation , X_{rob} is the 3D position vector of the camera relative to the global frame, q_{rob} represents the rotation vector, v_{rob} is the linear speed and ω is the angular speed. On the other hand, as the whole map has to be included into the filter, all the features global position state vectors Y_i have to be included into the total state vector X. So, the state vector $X = (X_v \quad Y_1 \quad Y_2 \quad \cdots)^T$ and its corresponding covariance matrix P are defined.

B. Motion Model

The first stage to build the motion model is to *predict* the next state vector and covariance matrix. In this case the object to model is a mobile robot. Our model is based on a more general application (see [8]). It assumes constant speed (both linear and angular) during each time step. There will only be random speed changes, which will lead to the so-called *impulse model.* The motion model has been adapted to the navigation of a mobile robot by using some restrictions. These restrictions are to reduce the uncertainty on the "y" linear movement direction as well as the uncertainty on rotations around the "z" and "x" axes. According with this model, to predict the next state of the camera the function, $f_v = (X_{rob} + v_{rob} \cdot \Delta t \quad q_{rob} \times q[\omega \cdot \Delta t] \quad v_{rob} \quad \omega)^T$ is defined. The term $q[\omega \cdot \Delta t]$ represents the transformation of a 3 components vector into a *quaternion*. Assuming that the map does not change during the whole process, the absolute feature positions Y_i should be the same from one step to the next one.

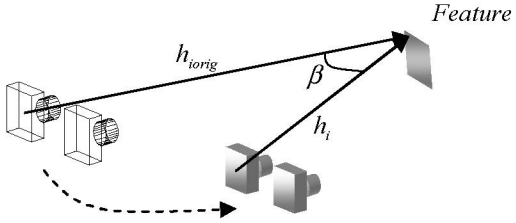


Fig. 1. Original and current feature measurement vectors.

C. Measurement Model

Visual measurements are obtained from the "visible" features positions. In our system we define each individual *measurement prediction* vector $h_i = (h_{ix} \quad h_{iy} \quad h_{iz})^T$ as the corresponding 3D feature position relative to the camera frame. To choose the features to measure, some selection criteria have to be defined. These criteria will be based on the feature "visibility", that is whether its appearance is close enough to the original one (when the feature was initialized). This is based on the relative distance and point of view angle respect to the one at the feature initialization phase (see Fig. 1).

The first step is to predict the measurement vector h_i. To look for the actual measurement vector z_i, we have to define a search area on the projection images. This area will be around the projection points of the predicted measurement h_i on both *left* and *right* images: $U_L : (u_L, v_L)$, $U_R : (u_R, v_R)$. To obtain the image projection coordinates, first we apply the simple "pin-hole" model and then it is distorted using the radial and tangential distortion models, which are detailed in [8]. To obtain z_i we need to solve the inverse geometry problem, applying the distortion models as well (see [8]).

Respecting to the search areas, they will be calculated based on the uncertainty of the feature 3D position, what is called *innovation covariance* S_i. (see [13]). As we have two different image projections, S_i needs to be transformed into the projection covariance P_{U_L} and P_{U_R} using equation (1).

$$P_{U_L} = \frac{\partial U_L}{\partial h_i} \cdot S_i \cdot \left(\frac{\partial U_L}{\partial h_i} \right)^T \quad ; \quad P_{U_R} = \frac{\partial U_R}{\partial h_i} \cdot S_i \cdot \left(\frac{\partial U_R}{\partial h_i} \right)^T \qquad (1)$$

These two covariances define both elliptical search regions, which are obtained taking a certain number of standard deviations (usually 3) from the 3D Gaussians. Once the areas, where the current projected feature should lie, are defined, we can look for them. At the initialization phase, the left and right images representing the feature *patches* are stored. Then, to look for a feature patch, we perform normalized *sum-of-squared-difference correlations* across the whole search region (see [13]). The path appearance is modified depending on the robot point of view using the *Patch Adaptation* method described in [8]. This helps on the search correlations phase in the sense of extending the tracking of the patch.

III. High Level SLAM

As it was stated before, in large environments, as the number of landmarks grows the covariance matrix P size increases until the processing time exceeds real time constraints. To avoid this, only a local visible window of landmarks is introduced into the EKF. This, as we will show on the results, allows an almost constant processing time with number of landmarks increasing (see Fig. 6).

On the other hand, we need to keep the global map error as low as possible. Therefore we need to assure the whole map consistency as well. As we only keep local uncertainty information on the visible features, it is necessary to add a higher level process that preserves the global map uncertainty history along the robot's path, that is the cumulative covariance matrix P_G, which is obtained as follows:

$$P_G(k)=P_G(k-1)+P_{XX}(k) \qquad (2)$$

To do that we divide the global map into local sub-maps identified by the *SIFT fingerprints* $SF=\{sf_l | l\in 0...L\}$, each of them composed by a set of *SIFT features* $YF^l=\{Yf_m^l | m\in 0..M\}$. The process consists in periodically take fingerprints along the path covered by the robot (see Fig. 2). Each time a new fingerprint is to be taken it is evaluated comparing it to the previously acquired fingerprints within an uncertainty search region. This region is obtained from P_G because it keeps the global uncertainty information of the whole map. In case that the result of the evaluation gives that the robot is in a previously visited place, a *closed loop* situation will be identified. This situation is explained later on. The philosophy of the EKF sub-maps was previously presented in [12]. One of the main differences between their approach and ours is the local map management. In [12], the EKF process is carried out taking into account all landmarks within the local maps. We take into account only the visible ones. This allows us to keep always a reduced number of landmarks being processed within the EKF. On the other hand, in [12], a global map is tried to be built joining all local maps. Then, a process is carried out to identify all duplicated landmarks, closing all possible loops inside. Instead, we continuously keep the accumulated global uncertainty, allowing the global map correction at any time as soon as a closed loop situation is detected.

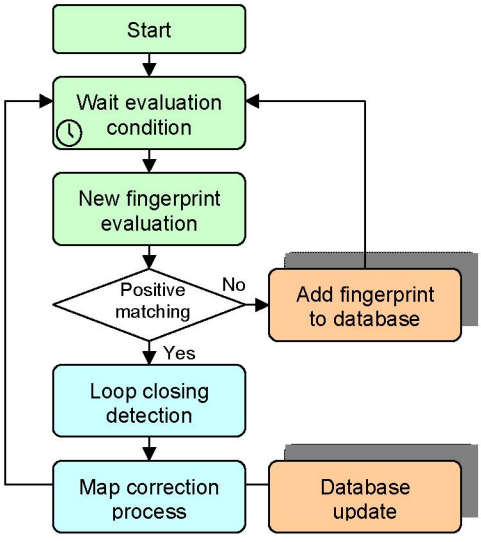


Fig. 2. High level map management.

A. SIFT Fingerprints

As we explained before, the way to identify a place is based on the so-called SIFT fingerprint. These fingerprints are composed by a number of SIFT landmarks distributed across the reference image and characterize the visual appearance of the image. SIFT features were introduced by D. Lowe in [9]. SIFT features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability. This is achieved by the association of a 128 length descriptor to each of the features, which will identify uniquely all of them. The overall SIFT features extraction is described as follows:

1. Scale-space extreme detection: The first stage of computation searches over all *scales* and image locations. It is implemented by using a difference-of-Gaussian function to identify potential interest points (local maxima and minima).
2. Keypoint localization: At each candidate location, a detailed model is used to determine the feature location and scale.
3. Orientation assignment: One or more orientations are assigned to each keypoint location based on local image gradient directions. Relating the future operations to these directions the invariance to orientation is achieved.
4. Keypoint descriptor: The local image gradients are measured at the selected scale in the region around each keypoint. This information is transformed into a representation that will provide an identification of the feature.

In our global process we will store these SIFT feature descriptors $\vec{\delta}$ within the global database, using them for the fingerprints matching process. The left image coordinates and the 3D position are also stored $Yf_m^l=\left(u_L \quad v_L \quad X \quad Y \quad Z \quad \vec{\delta}\right)$.

B. Matching Process

One of the main issues on SLAM in large environments is the *loop-closing* problem. The first issue to solve is the recognition of previously visited places, as it was stated before. Once a new fingerprint is identified it is evaluated, that is, it is compared against all stored fingerprints within the uncertainty area. This comparison is carried out through a matching process which takes into account, for each pair of fingerprints (sf_A, sf_B), both the number of recognized SIFT features and their relative positions within the images to compare. The overall process is as follows:

1. Computation of the euclidean distance between all detected SIFT features on both images, and selection of those close enough.

2. Lines connecting each pair of matched features are calculated. The corresponding lengths $Ln_{i,j}^{A-B}$ and slopes $Sp_{i,j}^{A-B}$ are computed as well (see Fig. 8 (b)).

3. The global fingerprint matching probability is computed as a weighted function of 3 parameters: *Number of matched features*, *lengths scatter* and *slopes scatter* (see (3)).

$$P_{fp\_match} = m_1\sigma_{Sp}^{-1} + m_2\sigma_{Ln}^{-1} + m_3 \cdot num\_matches \tag{3}$$

An *outlayers* filter is carried out to discard erroneous individual feature matches different enough from the mean showed on (3).

Once the loop-closing situation has been detected, the whole map must be corrected according to the old place recognized. The first step is then, to update the current robot state $(X_{rob} \quad q_{rob} \quad v_{rob} \quad \omega)^T$ with the detected pre-visited place. To do that we use the epipolar geometry applied to the matched SIFT features in the same way as in the low level SLAM stage. This is achieved thanks to the stored fingerprint states X_{fp}, which represent the robot states at the time of the fingerprint creation. After that, the rest of the map, including feature positions Y_i and fingerprint states X_{fp}, along the loop must be updated accordingly. However, we must assure that the resultant map is *consistent*. The idea behind is that the global uncertainty P_G will always grow as long as the robot moves on the environment. That means: the error in self-locating will increase until the robot revisits an old place that helps to reduce its own uncertainty. In terms of map construction, we can conclude that the oldest map features will have less associated uncertainty and will have to be corrected in a lower degree. On the opposite side, a higher degree of correction will be applied to more recent features. This degree of correction will be modulated as a function of the historic accumulated uncertainty along the whole loop (see Fig. 3). On this figure we can also observe the location of the fingerprints right after each of the corners on the path.

On equations (4), (5) and (6) we show the 3 consecutive steps applied to the position of a single feature in order to calculate its estimated new corrected value. $T(P_{Gi})$ represents the trace of the global covariance of fingerprint associated to the feature i. $\vec{D}_R$ is the 3 components rotation expression of

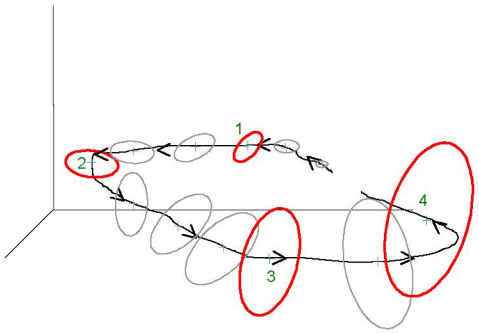


Fig. 3. Representation of the fingerprints global uncertainty P_G, in red colour, increasing along the robot path. Numbers represent each fingerprint.

the robot rotation matrix R_{rob}. Finally, *Rot* represents the transformation of the 3 components vector to the rotation matrix expression. The attributes *init* and *end* refer to the situation before and after the map correction.

$$Y'_i = R_{init}^{-1}(Y_i^{init} - X_{init}) \tag{4}$$

$$Y''_i = Y'_i \cdot Rot\left[\frac{T(P_{Gi})}{T(P_{Gend})} \cdot (\vec{D}_{R\,fin})^{-1} + \left(1 - \frac{T(P_{Gi})}{T(P_{Gend})}\right) \cdot (\vec{D}_{R\,init})^{-1}\right] \tag{5}$$

$$Y_i^{fin} = Y''_i - \left[\frac{T(P_{Gi})}{T(P_{Gend})} \cdot X_{fin}^{-1} + \left(1 - \frac{T(P_{Gi})}{T(P_{Gend})}\right) \cdot X_{init}^{-1}\right] \tag{6}$$

Once the loop is closed, P_G takes the value of the associated old fingerprint identified. Thus, we update the global uncertainty to the new situation. In addition, old visited features will become visible again, and can be incorporated to the low level EKF process.

IV. Results

In order to test the behaviour of our system, a test video sequence has been used. The cameras employed were the Unibrain Fire-i IEEE1394 modules with additional wide-angle lens, which provide a field of view of around 100º horizontal and vertical. Both cameras are synchronized at the time of commanding the start of transmission. The calibration is performed offline using a chessboard panel using the method referenced in [10]. The test video sequence was taken by moving the robot along the upper floor of our Polytechnic School building. The complete path, from the start point to the loop-closing place, has a perimeter of 283.25 m (see Fig. 4). We have implemented the low level SLAM using two techniques: our proposal and the SIFT method introduced by D. Lowe [9], in order to compare its performance in a large environment. Fig. 5 depicts the obtained results based on the low level SLAM. Both ours and SIFT results are showed together with the ground truth data. From these results we observe two main deviation points on each implementation. The first deviation is accused with the SIFT method on the first curve of the path, while using our implementation, the robot tend to deviate on the third curve. If we represent the cumulative mean error $\varepsilon_n = (1/n_{total}) \cdot \sum_{i=0}^{n} |X_i - Xref_i|$ on Fig. 7,

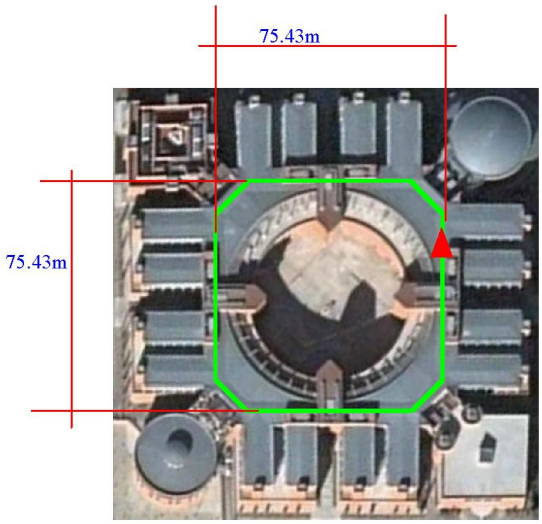


Fig. 4. Aerial view of the path covered by the robot, marked on green colour. The red arrow indicates the starting and loop closing position.

we appreciate that it appears to be higher on SIFT implementation. The reason is that the first deviation causes a cumulative drift over the rest of the path. As a general rule, low level SIFT landmarks should be correctly recognized within a longer period of time, that is, SIFT landmarks should have a longer lifetime. Thus, the local cumulative error should be lower using SIFT landmarks. However, as we will show later, the big disadvantage of using SIFT within the low level SLAM, is the significant processing time increase. Respecting to the high level SLAM, Fig. 8 depicts the representation of the map estimated, including all landmarks and fingerprints. The sequence represents the map right before and after the loop-closing situation. As it is shown, the map keeps the consistency also after its correction. After that situation, the system keeps the consistency re-introducing the old visible landmarks on the low level EKF and still detecting old fingerprints. Respecting to the processing time, the real-time implementation imposes a time restriction, which shall

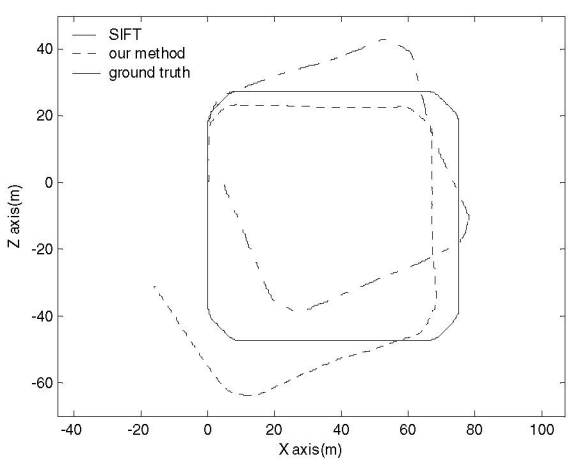


Fig. 5. Estimation of the path covered by the robot using SIFT and our method. The reference (ground truth) is drawn on solid line.

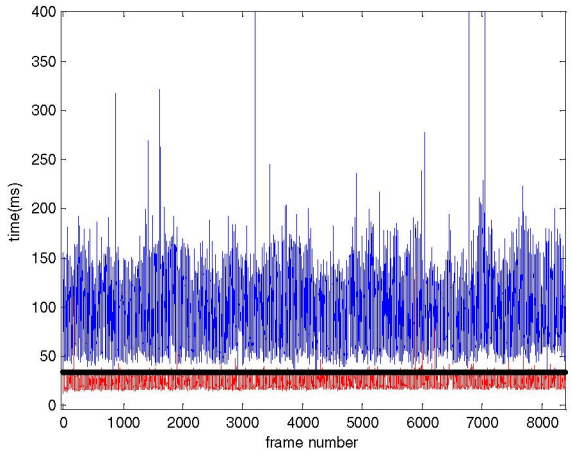


Fig. 6. Processing times for low level SIFT (in blue colour) and our method in red colour. Real time limit is represented as a constant 33 ms black line.

not exceed 33 ms for a 30 frames/second capturing rate. All results were taken using a 2.0 GHz speed CPU. Fig. 6 depicts the processing times along the whole robot path for both SIFT and our implementation. We can see that only our method is able to work under the real time constraint, remaining the average processing time quite constant along the whole path, showing the benefit of using our method instead of the SIFT one. On Table 1 we show the average processing times for some of the most important tasks in the process. Respect to the low level SLAM tasks, we can see the main cause of the higher processing time for SIFT implementation, which is due to the increase of *measurements* and *feature initialization* phases computational costs. Even though we restricted the keypoints search to the minimum needed area, the successive Gaussian blurring phases contribute to increase the processing time. This is particularly evident for the case of the feature initialization phase, where the search area is extended along the whole epipolar line, though we restricted its length for

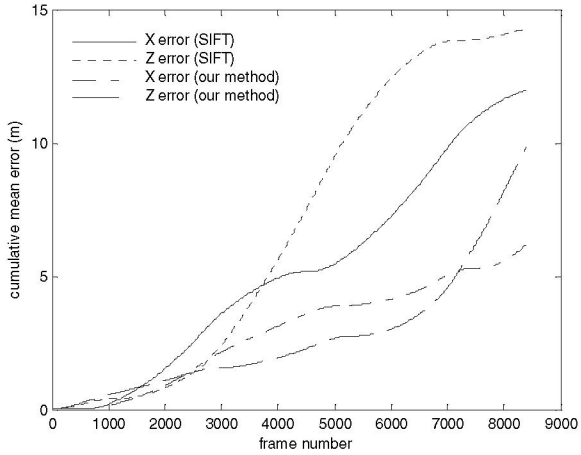
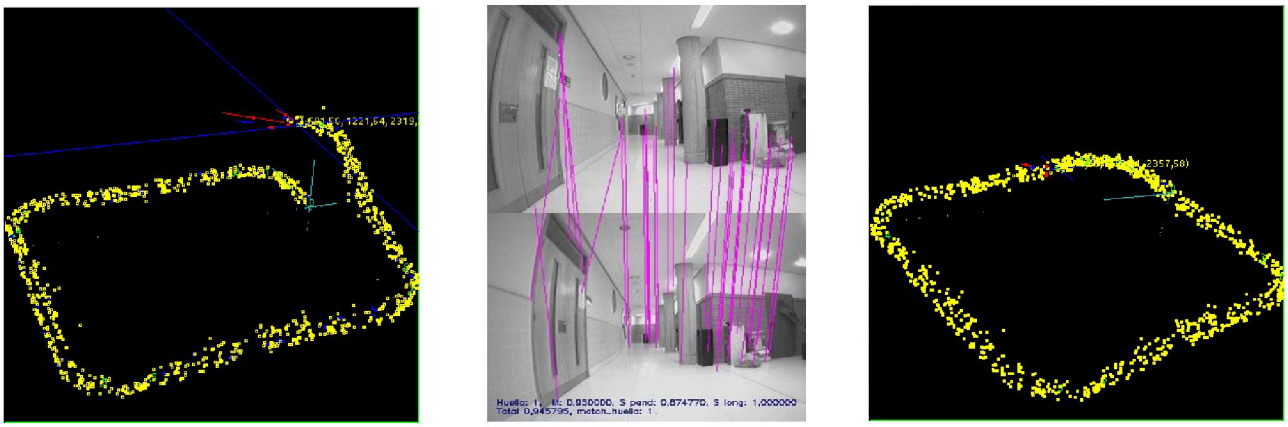


Fig. 7. Cumulative mean error for X axis and Z axis respect to the frame number *n*. Results are shown both for the SIFT implementation and our method.



(a) (b) (c)

Fig. 8. Map representation of a loop-closing situation. The low level landmarks are represented in yellow colour, while the visible and correctly measured are represented in red colour. The green numbers show the fingerprint locations. The robot position is represented on the centre of the big blue cross: P_G. a) Representation right before the map correction. b) Fingerprint SIFT features matching. c) Representation right after the map correction.

$\mathrm{lm} \to \infty$ search range. Regarding the high level SLAM, as it is shown, the time dedicated to *fingerprint matching* process as well as the correction of the map at the time of loop closing, having 1630 landmarks, is significantly higher than real time. However, both tasks do not belong to the continuous self-locating process carried out by the low level SLAM, so there is no need to complete them within a single frame time slot. Therefore, we can obtain a positive fingerprint matching result a few frames after it was really detected. Then, we can go back and start loop-closing task. This implies that both of these tasks can be computed in *parallel*, keeping them outside the real time computation.

TABLE I.

PROCESSING TIMES

Low level SLAM processing times			High level SLAM processing times (parallelized).	
Implementation	Our method	SIFT		
Number of features	5	5	Number of features	1630
Filter step	Time			Time
Measurements	3 ms	47 ms	Fingerprint matches	3 s
Filter update	5 ms	5 ms	Loop closing + graphic representation time	4 s + 10s
Feature initializations	7 ms	62 ms		

V. CONCLUSION

We have presented a two hierarchical SLAM levels method that allows self-locating a robot by measuring the 3D positions of different natural landmarks. Several benefits have been shown on the low level SLAM respect to the use of SIFT features mainly on the processing time area. Respecting to the high level SLAM based on SIFT fingerprints, there has been proved that it solves the loop-closing problem keeping the real time behavior constant along the path. Nowadays we are working on improving our taking fingerprints technique.

ACKNOWLEDGMENT

The authors would like to express their gratitude to the Comunidad de Madrid and the University of Alcalá for their support through the projects RoboCity2030 (CAM-S-0505/DPI/000176) and LOMUCO (CCG06-UAH/DPI-0721).

REFERENCES

[1] Lopez E, Bergasa L M, Barea R, Escudero M. "A Navigation System for Assistant Robots Using Visually Augmented POMDPs". Autonomous Robots, Vol. 19, No. 1. pp. 67-87. 2005.

[2] P.M. Newman, J.J. Leonard, J. Neira and J. Tardós. "Explore and return: Experimental validation of real time concurrent mapping and localization." Proceedings of the IEEE Conference on Robotics and Automation, pp.1802-1809, 2002.

[3] A. J. Davison. "Real-time simultaneous localisation and mapping with a single camera." *Proceedings of the 9th International Conference on Computer Vision, Nice*, 2003.

[4] P. Elinas, R. Sim, J. J. Little. "SLAM: Stereo Vision SLAM Using the Rao-Blackwellised Particle Filter and a Novel Mixture Proposal Distribution." *ICRA 2006.*

[5] M. Bosse, P. Newman, J. Leonard and S. Teller. "SLAM in Large-scale Cyclic Environments using the Atlas Framework." *IJRR 2004.*

[6] P. Newman, D. Cole and K. Ho. "Outdoor SLAM using Visual Appearance and Laser Ranging." *ICRA 2006.*

[7] U. Frese and L. Schroeder. "Closing a Million-Landmarks Loop." *IROS 2006.*

[8] D. Schleicher, L.M. Bergasa, E. Lopez and M. Ocaña. "Real-Time Simultaneous Localization and Mapping using a Wide-Angle Stereo Camera and Adaptive Patches" *IROS2006.*

[9] D.G. Lowe and J. Little. "Vision-based mobile robot localization and mapping using scale-invariant features." In International Conference on Robotics and Automation, Seoul, Korea, pp. 2051-58. 2001.

[10] Heikkila and Silven. "A Four-step Camera Calibration Procedure with Implicit Image Correction." *CVPR97.*

[11] J. Shi and C. Tomasi. "Good features to track." *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 1994, pp. 593-600.

[12] J. Tardós, J. Neira, P. Newman, and J. Leonard, "Robust mapping and localization in indoor environments using sonar data," Int. J. Robotics Research, vol. 21, no. 4, pp. 311–330, 2002.

[13] A. J. Davison. "Mobile Robot Navigation Using Active Vision." *PhD Thesis, University of Oxford, 1998.*