Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Real-Time Simultaneous Localization and Mapping using a Wide-Angle Stereo Camera and Adaptive Patches

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

***Abstract* - This paper presents a new method for real-time ego-motion calculation applied to the location/orientation of a cheap wide-angle stereo camera in a 3D environment. The objective is to apply it to a mobile robot navigation system. To achieve that, the goal is to solve the Simultaneous Localization and Mapping (SLAM) problem. Our approach consists in the 3D sequential mapping of natural land-marks by means of a stereo camera, which also provides means to obtain the camera location/orientation. The dynamic behavior is modeled using a top-down Bayesian method. The results show a comparison between our system and a monocular visual SLAM system using a hand-waved camera. Several improvements related to no priori environment knowledge requirements, lower processing time (real-time constrained) and higher robustness is presented.**

Index Terms* – *SLAM, wide-angle vision, stereo vision, real-time.

I. Introduction

Real-time Simultaneous Localization and Mapping (SLAM) is a key component in robotics. In last years several approaches have been used [1][2][3][4]. Successful implementation of SLAM in robotics have generally been achieved with laser or sonar range sensors and built maps for controlled robots moving in 2D and using accurately modeled dynamics. 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 [5]. In [6] and [7] several 3D visual SLAM methods based on a stereo camera and SIFT features are presented. The first one uses in one hand a Particle Filter (PF) to estimate the robot pose and in the other an EKF to estimate each landmark state. Although no additional information apart from the visual one is used, the main disadvantage of the method resides on the high processing time, making it not suitable for real-time applications. The second work uses SIFT features as well, however the motion model is based on odometry information, which, depending on the terrain could have a high drift error. Processing time is too high as well. In [8], a landmark-based 3D environment model is built as well. In this case scale-invariant features are also used. The feature positions prediction is based on the vehicle odometry information, therefore this approach is not suitable for hand-waved camera systems as well as any other one that relies only on visual information.

The most important researches in visual SLAM have been achieved by Andrew J. Davison from the University of Oxford. He started using an active stereo vision system in order to recover 2D position of a robot [9] but then he has focused his work on real-time 3D SLAM using monocular vision [5]. The baseline for this work is the solution described in [5]. Their approach is to implement a vision based SLAM method using the *Extended Kalman Filter* (EKF) [10]. One of the major issues when trying to solve the SLAM problem is related to the map building process. Due to the error on measurements, it is common the appearance of a drift during the map building. This leads to a bad correspondence of the map when visiting "old" places. Therefore, in order to allow a long-term well-located camera it is needed to include the complete map into the state vector. In this case, the map will be composed of a number of natural landmarks (identified by their corresponding features), which will grow at the same time the camera moves over the scene and visits new places. These marks will not be only the "output" of the process, but also the mean to self-locate the camera within the environment. Having the standard perspective camera model, it is possible to obtain two coordinates of the features relative positions, but it is not possible to directly know the depth of these positions. That leads to the following 3 limitations:

1) When a new feature, which identifies the mark, is to be initialized, it cannot be done in one single step. The new feature has to be modeled as a semi-infinite line that represents all the possible depths. Then, by mean of a PF algorithm, at each time step measurement, the belief of the feature depth trends to concentrate on the final value.

2) At the beginning there is no prior knowledge of the camera position/orientation, therefore it is not possible to obtain the final depth required for the first captured features. This leads to the need of having a number of known features that will have to be located manually.

3) Lenses normally used in computer vision have a narrow field of view (40 to 50 degrees). Then, all the features measured lie very close together and the sets of features to be visible through large motion is small. As consequence, in such situations small rotations and translations are ambiguous and camera movement range must be low.

This paper presents a solution to the limitations of the monocular visual SLAM using a cheap wide angle stereo camera instead of a standard single one. The use of wide-angle cameras improves SLAM results, with increased movement range, accuracy and ability to track agile motion, as can be seen in [11]. Additionally, our system is able to completely locate any feature in one single step, avoiding the two issues mentioned above. Besides that, having two cameras we obtain some redundant information on each feature position, allowing then a more robust location.

II. 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:

$$X_v = \begin{pmatrix} X_{rob} & q_{rob} & v_{rob} & \omega \end{pmatrix}^T \quad (1)$$

In equation (1), 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. A special clarification needs the rotation vector q_{rob}. For representing a rotation, it is enough to use a three components vector. However, using a four components vector *quaternion*, it is easier to compose sequenced rotations. This vector defines a rotation angle θ around the unit vector $\begin{pmatrix} u_x & u_y & u_z \end{pmatrix}^T$ in the following way:

$$q_{rob} = \begin{pmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} \cos\frac{\theta}{2} \\ u_x \sin\frac{\theta}{2} \\ u_y \sin\frac{\theta}{2} \\ u_z \sin\frac{\theta}{2} \end{pmatrix} \quad (2)$$

The corresponding rotation matrix R can be obtained as a function of q_{rob} as it is explained in [5].

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 $X = \begin{pmatrix} X_v & Y_1 & Y_2 & \cdots \end{pmatrix}^T$ and its corresponding covariance matrix P are defined. With these two parameters, the EKF implementation can be described as follows, assuming k the step index.

1) Prediction step:

$$\hat{X}(k+1|k) = f(X(k|k)) = f(k|k) \quad (3)$$

$$\hat{P}(k+1|k) = \frac{\partial f}{\partial X}(k|k) \cdot P(k|k) \cdot \left(\frac{\partial f}{\partial X}(k|k)\right)^T + Q(k) \quad (4)$$

2) Update step:

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + W(k+1) \cdot \eta(k+1)_{tot} \quad (5)$$

$$P(k+1|k+1) = P(k+1|k) - W(k+1) \cdot S(k+1) \cdot (W(k+1))^T \quad (6)$$

On this implementation, Q and S are the *process noise* and *measurement uncertainty* covariances, respectively. Also, η_{tot} is the *innovation* vector, it means, the difference between the current measurement vector and the predicted measurement one: $(\eta_{tot} = z_{tot} - h_{tot})$. For clarity reasons, the step index k will be omitted in the rest of the paper.

III. 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 stereo camera, which can be carried by a person. It means that it can be freely but smoothly moved. As we do not have any influence on the camera movement, the motion model 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.* In our work the motion model is adapted to the movement of a mobile robot. Therefore, some restrictions will be applied. These restrictions will be to reduce the uncertainty on the "y" linear movement direction as well as the uncertainty on rotations around the "z" and "x" axes.

In order to predict the next state of the camera the function, $f_v = \begin{pmatrix} X_{rob} + v_{rob} \cdot \Delta t & q_{rob} \times q[\omega \cdot \Delta t] & v_{rob} & \omega \end{pmatrix}^T$ is defined. The function $q[\omega \cdot \Delta t]$ 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. Therefore, the global prediction function f will be composed as follows: $f = \begin{pmatrix} f_v & Y_1 & Y_2 & \cdots \end{pmatrix}^T$. To calculate Q, a noise vector $n = \begin{pmatrix} V & \Omega \end{pmatrix}^T$ is defined. This vector represents the random speed changes mentioned before. Assuming that linear and angular speeds are independent, the covariance matrix of n will be diagonal. Then, Q can be calculated via the corresponding jacobian function: $Q = (\partial f/\partial n) \cdot P_n \cdot (\partial f/\partial n)^T$.

IV. Measurement Model

Visual measurements are obtained from the "visible" features positions. As difference as [5] in our system we define each individual *measurement prediction* vector $h_i = \begin{pmatrix} h_{ix} & h_{iy} & h_{iz} \end{pmatrix}^T$ 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). To evaluate that, three tests are applied to each feature (see Fig. 1):

1. First, check that the feature image projection lies within the field of view for each of the cameras.

2. Then, check that the angle β between the current point of view and the original point of view is small enough.
3. The last check is to test that the distance from the point of view to the feature is not so different to the original one.

A. *Measurement Prediction*

Prior to perform the actual measurement, for establishing the region to look for the actual feature position of each of the selected features, each h_i has to be obtained. It can be calculated as the result of a coordinate frame change (from the global reference Y_i to the camera reference h_i).

$$h_i = R^{-1}(Y_i - X_{rob}) \tag{7}$$

B. *Measurement Search*

To obtain z_i, first we have to calculate the projection coordinates of h_i on both *left* and *right* images: $U_L : (u_L, v_L)$, $U_R : (u_R, v_R)$. Taking into account the use of *wide-angle* camera optics, it is not a good approximation to apply directly the *pin-hole* model to obtain such coordinates. It is recommendable to use a direct and inverse *radial* and *tangential distortion models.* Therefore, to obtain the final image projection coordinates, first the simple "pin-hole" model is applied and then, the result is "distorted" by means of the distortion models.

As an example, if we apply the direct pin-hole projection model to the left image, as it is described in [12], we obtain the "undistorted" projection coordinates. To make it easier, we take the common camera frame coincident with the left camera reference frame. $u_{LS} = CC1_L - FC1_L(h_{ix}/h_{iz})$, $v_{LS} = CC2_L - FC2_L(h_{iy}/h_{iz})$. The parameters $FC1$ and $FC2$ correspond to the camera focal lengths, while $CC1$ and $CC2$ are the principal point coordinates, i.e. the camera *intrinsic parameters*. The corresponding jacobians $\partial U_{LS}/\partial h_i$ and $\partial U_{RS}/\partial h_i$ can be easily calculated from (11) and the equivalent equation in the right camera, respectively. Applying the same distortion models, the jacobians $\partial U_L/\partial U_{LS}$ and $\partial U_R/\partial U_{RS}$ can be also calculated (see VII). As a remark, to apply the procedure for the right camera, first we have to calculate h_i relative to the right camera reference frame h_{iR}:

$$h_{iR} = R_{\text{int}}^{-1} \cdot (h_i - T_{\text{int}}) \tag{8}$$

In equation (8), R_{int} and T_{int} are the *extrinsic parameters* between left and right cameras. The jacobian $\partial h_{iR}/\partial h_i$ needs to be calculated as well. Once the transformation is done, the right camera projection coordinates can be obtained following the same procedure as for the left camera.

Besides the calculation of this projection coordinates, here we need to calculate the projection of all the pixels within the patch following the procedure explained in VIII.

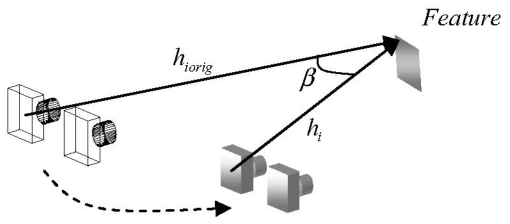

Fig. 1 Original and current feature measurement vectors.

1) *Search Area calculation*

In order to look for the actual feature projections, we must define the search area around the predicted projections to limit the search. This will be calculated based on the uncertainty of the feature 3D position, what is called *innovation covariance* S_i. It essentially depends on three parameters: The camera state uncertainty P_{XX}, the feature position uncertainty $P_{Y_iY_i}$ and the measurement noise R_i (see [9]). As we have two different image projections, S_i needs to be transformed into the projection covariance P_{U_L} and P_{U_R}.

$$P_{U_L} = \frac{\partial U_L}{\partial h_i} \cdot S_i \cdot \left(\frac{\partial U_L}{\partial h_i}\right)^T \; ; \; P_{U_R} = \frac{\partial U_R}{\partial h_i} \cdot S_i \cdot \left(\frac{\partial U_R}{\partial h_i}\right)^T \tag{9}$$

These two covariances define both elliptical search regions, which are obtained taking a certain number of standard deviations (usually 3) from the 3D Gaussians.

2) *Correlation Search*

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 [9]).

The best correlation matching is then compared to a threshold value. If both correlations are good enough, the new measured projection coordinates are captured in order to perform the update process. Otherwise, the feature is marked as "unsuccessfully measured."

C. *Measurement Vector Calculation*

To obtain z_i we need to solve the inverse geometry problem described in [12]. We take the measured new projection coordinates U_L, U_R as a basis. First, we need to obtain the "undistorted" projection coordinates U_{LS}, U_{RS}, as it is explained in chapter VII. The projection coordinates are related to the measurement vector z_i by means of the so-called *projection equations* (see [12]), where m_{Lij} and m_{Rij} are the elements of the *projection matrices* M_L and M_R for left and right camera. They can be calculated as a function of the known intrinsic parameters matrices: I_{CL} and I_{CR}. Then, to obtain M_L and M_R from I_{CL} and I_{CR}, we just need to express I_{CL} and I_{CR} in the left and right camera reference frames, using the extrinsic parameters as it was showed in (8). For the left camera, $M_L = I_{CL}$ because the camera reference frame is actually the left camera reference frame.

From the projection equations we can form the redundant equation system showed in [12]. Transforming it into the matrix form $A \cdot z_i = b$, the system can be solved giving the following result: $z_i = \left(A^T A\right)^{-1} A^T b$. At the end, before to performing the filter update, all feature measurements must be combined to form the "total" vectors.

D. *Filter Update*

In order to perform the update, the Kalman gain W must be calculated using the following expression: $W = P \cdot \left((\partial h / dX)_{tot}\right)^T \cdot S^{-1}$. For each individual feature, the jacobians $\partial h_i / dX_v$ and $\partial h_i / dY_i$ are calculated from (7), which conveniently grouped form the total jacobian $(\partial h / dX)_{tot}$. Following the same procedure z_{tot}, which will contain all feature measurements, is formed as well.

In the other hand, as it is stated in (11), to be able to calculate S we still need to calculate the measurement noise covariance R_{tot}. Because the 3D feature position vector is used as the measurement, this calculation is not so evident.

1) Measurement Noise Covariance Calculation

Starting from the projection coordinates, we can assume an intrinsic uncertainty on its determination, which will be one pixel for each coordinate and for each image: (u_L, v_L), (u_R, v_R). Furthermore, uncertainty in u and v is assumed independent and gaussian distributed. Therefore, we can define a vector $T_i = \begin{pmatrix} u_L & v_L & u_R & v_R \end{pmatrix}$ with the four coordinates of both images. The covariance matrix R_{Ti} for this vector will be diagonal. To calculate the feature measurement noise R_i, the following transformation will be done: $R_i = (\partial h_i / \partial T_i) \cdot R_{Ti} \cdot (\partial h_i / \partial T_i)^T$. Starting from the equation system $A \cdot h_i = b$, we calculate $\partial (A \cdot h_i) / \partial T_{iS} = \partial b / \partial T_{iS}$, where T_{iS} refers to the undistorted coordinates vector. Regrouping the equation we obtain $A \cdot (\partial h_i / \partial T_{iS}) = C$, and therefore $\partial h_i / \partial T_{iS}$ can be found. The matrix C is obtained as a result of the regrouping:

$$C = \begin{pmatrix} (1) & 0 & 0 & 0 \\ 0 & (1) & 0 & 0 \\ 0 & 0 & (2) & 0 \\ 0 & 0 & 0 & (2) \end{pmatrix}, \text{ where } \begin{cases} (1) = -m_{L31} h_{ix} - m_{L32} h_{iy} - m_{L33} h_{iz} - m_{L34} \\ (2) = -m_{R31} h_{ix} - m_{R32} h_{iy} - m_{R33} h_{iz} - m_{R34} \end{cases} \quad (10)$$

In order to obtain $\partial h_i / \partial T_i$, the jacobian $\partial h_i / \partial T_{iS}$ is transformed using $\partial U_{LS} / \partial U_L$ and $\partial U_{RS} / \partial U_R$. Assuming measurements are independent, R_{tot} can formed by all individual R_i in a diagonal arrangement.

The total covariance S is obtained using the previous calculated values:

$$S = \left[(\partial h / dX)_{tot} \cdot P \cdot (\partial h / dX)_{tot}^T\right] + R_{tot} \quad (11)$$

V. Feature Initialization

One important aspect on this implementation is the way new features are incorporated into the filter process. When a new feature needs to be initialized, its corresponding patch will be searched within a rectangular area randomly located on one of the camera images (usually the left one). If the search process does not success, a new random location for the region is generated. The maximum number of attempts is limited to 10. Then the filter is one step moved forward and the process is reinitiated.

A. *Best Feature Search*

At the time to look for the best feature to introduce in the filter, we need to assure "good tracking" properties. It means that this feature must be correctly distinguished from the rest of the image along the camera movement. In [15], an operator to measure the "goodness" of a feature is described. It applies the *intensity gradient* on both vertical and horizontal directions. The operator is calculated on each of the pixels of the feature patch to evaluate in an efficient way. In case that the absolute maximum value is good enough, the corresponding feature is selected.

B. *Feature 3D Position Calculation*

Once the feature is selected, for including it into the filter, the absolute position vector Y_i and its covariance matrix $P_{Y_i Y_i}$ needs to be obtained. Unlike the approach used in [5], here we can obtain Y_i just in one step because, as in the measurement process, we can solve the four equations redundant system. Besides that, at this time, we obtain the estimated 3D position of all the pixels within the patch as it is explained in VIII.

1) Epipolar Correspondence Search

Taking the feature patch found on the left image, the first step is to look for its corresponding one on the right image. According to the stereo theory this patch must lie over a line called *epipolar line* (see [12]). Then, we must limit the search region to be close to that line. Once the region is defined, the process will consist in make correlations with the left patch, as it is done in the measurement process. The epipolar line equation is defined as $ax + by + c = 0$. The three coefficients are calculated using the *fundamental matrix F.* This matrix is obtained by expressing the *essential matrix E* in image pixel coordinates: $F = \left(C_R^{-1}\right)^T \cdot E \cdot C_L^{-1}$. Therefore, the mentioned coefficients are calculated in this way: $\begin{pmatrix} a & b & c \end{pmatrix}^T = F \cdot \begin{pmatrix} u_L & v_L & 1 \end{pmatrix}^T$.

2) Absolute Position Calculation

Once the left and right projection coordinates are obtained, Y_i can be calculated. The procedure to follow is the same that the one used for the measurement vector calculation. However, in this case, the feature position is relative to the global reference frame. It means that the projection matrices M_L and M_R have to be also relative to the global reference frame.

3) Covariance matrix Calculation

The feature position covariance $P_{Y_i Y_i}$ depends on two sources: the camera state uncertainty P_{XX} and the feature measurement noise R_i: $P_{Y_i Y_i} = (\partial Y_i / \partial X_v) P_{XX} (\partial Y_i / \partial X_v)^T + (\partial Y_i / \partial h_i) R_i (\partial Y_i / \partial h_i)^T$. We easily obtain $\partial Y_i / \partial h_i$ from (7). In order to calculate $\partial Y_i / \partial X_v$, we use $\partial Y_i / \partial h_i$ and $\partial h_i / \partial X_v$. This last jacobian is calculated from (7) as well, but we have to take into account the relationship between the quaternion q_{rob} and the rotation matrix R described in [1].

VI. Features Management

In order to maintain the map up to date, we need to define criteria about when to introduce (capture) new features and when to delete them.

At the beginning, the first feature captured is supposed to be a certain one; it means that it will have zero covariance. In the following steps, the rules to follow will be to capture new features to maintain, at least, 5 visible features at the same time. In addition to that, there will have to be, at least, 4 successfully measured features at the same time in order to avoid the complete loss the camera tracking.

In the other hand, some of the captured features can be "bad" features; i.e. features that are often unsuccessfully measured. This could be as a consequence of reflections, frequently occluded objects, etc. The rule to follow is to eliminate any feature that has been unsuccessfully measured more than a half of the attempts.

When a new feature is added to the filter, not only the total state vector X has to be modified, but also the total covariance matrix P. This is done by adding an extra row and column in P. To eliminate any feature, P will be modified by removing the corresponding row and column.

VII. Image Distortion Model

Due to the use of a wide-angle lens, we need to make use of a model that allows obtaining the equivalent "undistorted" projection coordinates from the distorted ones and the other way around. To do that, *radial* and *tangential* distortion models are applied. The models definition described in [14] is expressed in metric units; therefore in order to apply them, first we need to transform the pixel coordinates used on the filter. It is done by using the previously described intrinsic parameters $FC1$, $FC2$, $CC1$ and $CC2$. After applying the model, the result must be re-expressed in pixel. Taking the left camera, we define, in metric coordinates, the distorted (u_{LC}, v_{LC}) and undistorted (u_{LNC}, v_{LNC}) projection coordinates. Then, we can relate them as follows, using the direct model:

$$u_{LC} = f \cdot u_{LNC} + 2P_{1L} u_{LNC} v_{LNC} + P_{2L}\left(r^2 + 2u_{LNC}^2\right) \qquad (12)$$

$$v_{LC} = f \cdot v_{LNC} + P_{1L}\left(r^2 + 2v_{LNC}^2\right) + 2P_{2L} u_{LNC} v_{LNC} \qquad (13)$$

On the previous equations the following parameters are defined as: $r^2 = u_{LNC}^2 + v_{LNC}^2$, $f = 1 + K_{1L} r^2 + K_{2L} r^4$. On this model, we use K_{1L}, K_{2L} as the *radial distortion coefficients* and P_{1L}, P_{2L} as the *tangential distortion coefficients*. For the inverse model, we apply an iterative procedure, starting from the assumption that $u_{LNC} = u_{LC}$ and $v_{LNC} = v_{LC}$.

The jacobians $\partial U_L / \partial U_{LS}$ and $\partial U_R / \partial U_{RS}$ are calculated from (12) and (13), while the inverse jacobians are also calculated by inverting the previous ones.

VIII. Patch Transformation

As it was explained before, as the robot moves within the environment, the appearance of the patches changes respect to the original one (at the initialization time). At the same time as the robot moves away from the original state, the difference increases. This leads to an inaccurate matching correlation as well as an increase in unsuccessfully measurements. In order to reduce its impact a method to transform the appearance of the patch is applied. When the feature is initially captured, the 3D positions of all the pixels within the patch is estimated and stored. A flat robot parallel plane is assumed for the 3D patch representation. From now on, each time the corresponding feature is measured the patch appearance is estimated by predicting the projection of each of the pixels within the patch. An efficient interpolation method is applied to obtain the full patch appearance.

IX. Results

In order to test the behaviour of our system several video sequences have been used. The cameras used 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 [14]. To check the ability of revisiting "old" features, a 360 degrees turn around sequence was taken, which is shown on Fig 2. In addition to that, a long lateral translation sequence was also taken, where the ability to distinguish between rotations and translations was checked. The state estimation accuracy was tested by using another video sequence, which registered an 80 cm forward movement and 80 cm lateral movement. The results showed a 75 cm forward movement and 86 cm lateral movement. On the Fig 3, the system is located onto a mobile robot. The robot covered a path along different corridors within our laboratory. The real path covered by the robot is shown in green colour while the estimated one is shown in yellow. Along this path there were people crossing in front of the camera in different points. The number of captured marks was 984.

Respect to the processing time, the real-time implementation imposes a time restriction, which shall not exceed 33 ms for a 30 frames/second capturing rate. Testing both *mono* and *stereo* implementations showed the results on Table 1. The results were taken using a 2.0 GHz speed CPU. It was seen that, using the same video sequence, the number of features needed for a stable behaviour using the mono implementation was significantly higher than using the stereo one. The consequence is that, even with a 3 components feature measurement vector, the time needed for the filter updates is lower with the stereo implementation. Respect to the initialization phase, it appears to be slower with the stereo system due to the epipolar correlation phase. However, it has to be taken into account that, in this case, the feature is completely initialised in one single step, giving a more robust implementation.

TABLE I
PROCESSING TIMES

Using stereo camera		**Using mono camera**	
Number of features	16	Number of features	35
Filter step	Time	Filter step	Time
Measurements	3 ms	Measurements	5 ms
Filter update	5 ms	Filter update	42 ms
Feature initializations	20 ms	Feature initializations	10 ms

X. CONCLUSION

We have presented a system that allows self-locating a stereo camera by measuring the 3D positions of different natural landmarks. Several benefits have been showed comparing it with a single camera system, like avoidance of the prior-known features or the processing time improvements described above. Some improvements can be done in the distortion model in order to allow more accurate feature position estimations on lower distance ranges.

ACKNOWLEDGMENT

The authors would like to express their gratitude to the Community of Madrid and the University of Alcala for their support through the projects RoboCity2030 (CM: S-0505/DPI/000176) and SIMCA (CM-UAH: 2005/018).

REFERENCES

[1] Howie Choset and Keiji Nagatani. “Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization without Explicit Localization.” *IEEE Transactions on Robotics and Automation,* 2003, 17(2).

[2] 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.

[3] S. Thrun, W. Burgard and D. Fox. “A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. Proccedings of the IEEE International Conference on Robotics and Automation, 2000.

[4] 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.

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

[6] 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.*

[7] T. D. Barfoot. “Online Visual Motion Estimation using FastSLAM with SIFT Features.” *IROS 05.*

[8] S. Se, D. Lowe, and J. Little. “Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks.” *The International Journal of Robotics Research, volume 21, number 8, pages 735-758, 2002.*

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

[10]Zarachan, Paul and Howard Musoff. *Fundamentals of Kalman Filtering: A Practical Approach. Progress in Astronautics and Aeronautics.* Volume 190, American Institute of Aeronautics and Astronautics, Inc., 2000.

[11]A. J. Davison. “Real-time 3D SLAM with wide-angle vision”. *Proceedings of the 5th Symposium on Intelligent Autonomous Vehicles*. Lisboa, Portugal, 2004

[12]Faugeras, O. “Three-Dimensional Computer Vision a Geometric Point of View.” *MIT Press, Cambridge, Massachusetts, 1996.*

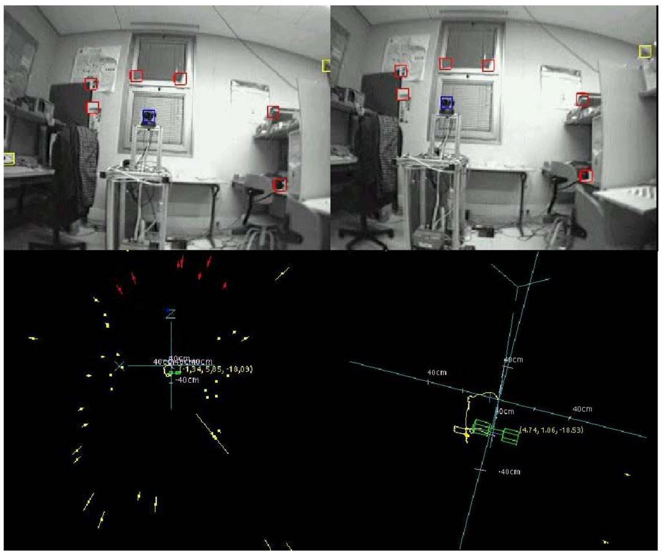

Fig. 2 Above: *Left* and *right* images from the same frame. This frame shows a 360 close loop movement. All features shown are revisited ones since the initial position. Successfully measured features are showed in red while unsuccessfully measured ones are showed in blue. No visible features are showed in yellow.

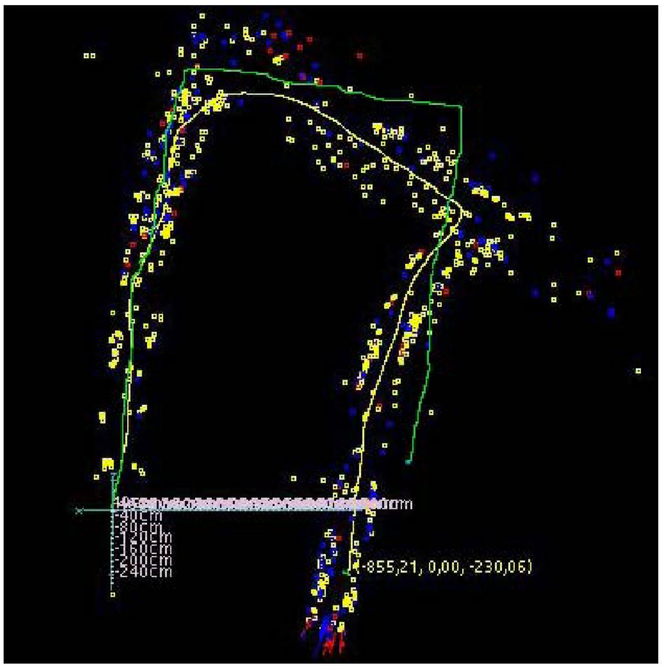

Fig. 3 Path covered by the robot. In yellow colour the path estimated by the system is shown. In green colour, the real path is shown.

[13]Web based literature. “Camera Calibration Toolbox for Matlab.” URL http://www.vision.caltech.edu/bouguetj/calib_doc/index.html, unpublished.

[14]Heikkila and Silven. “A Four-step Camera Calibration Procedure with Implicit Image Correction.” *CVPR97.*

[15]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.

[16]D. Schleicher, L.M. Bergasa, E. López and M. Ocaña. “Real-Time Simultaneous Localization and Mapping using a Wide-Angle Stereo Camera.” DIS2006.