Real-Time Simultaneous Localization and Mapping Using a Wide-Angle Stereo Camera

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. 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.

1. 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 [8]. In [5] and [6] several 3D visual SLAM methods based on a trinocular system are presented. The first one builds a 3D environment model, which is used to locate a mobile robot. However, due to the big amount of recovered features, this approach is not valid for real-time implementation. The second one builds a landmark-based 3D environment model as well. In this case scale-invariant features are 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. Regarding image distortion correction methods, in [7] a method for distorted image pairs correlation is presented. This method allows lower processing times on distortion but reduced accuracy. As it is shown later, the processing time associated to lens distortion correction in our implementation is not significant compared with other steps of the method. Therefore, it is worthwhile to apply the standard radial/tangential distortion correction methods.

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 [8]. The baseline for this work is the solution described in [8]. 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

Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)
0-7695-2589-X/06 $20.00 © 2006 IEEE

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 three limitations:

1) When a new feature, which identifies the mark, has 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. Also, having a number of tracked features allows spurious moving objects crossing the camera field of view without loosing the global tracking.

2. Extended Kalman Filter Application

In order to apply the EKF (see [9]), a state vector $X = \begin{pmatrix} X_v & Y_1 & Y_2 & \ldots \end{pmatrix}^T$ and its covariance matrix P need to be defined, where Y_i are the features global position state vectors representing the map and $X_v = \begin{pmatrix} X_{rob} & q_{rob} & v_{rob} & \omega \end{pmatrix}^T$ is the camera state vector representing its linear/angular position and speed respectively.

3. 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 order to predict the next state of the camera the function, f_v is defined:

$$f_v = \begin{pmatrix} X_{rob} + v_{rob} \cdot \Delta t & q_{rob} \times q[\omega \cdot \Delta t] & v_{rob} & \omega \end{pmatrix}^T \quad (1)$$

The function $q[\omega \cdot \Delta t]$ represents the transformation of a 3 components vector into a *quaternion* vector (see [9]). 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 the *process noise* 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: $Q = (\partial f / \partial n) \cdot P_n \cdot (\partial f / \partial n)^T$. Where P_n is the covariance matrix of n.

4. Measurement Model

Visual measurements are obtained from the "visible" features positions. As difference as [8] 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:

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. $\beta = \cos^{-1}\left((h_i \cdot h_{iorig}) / (|h_i| |h_{iorig}|) \right)$
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. $|h_i| / |h_{iorig}|$

4.1. 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).

$$h_i = R^{-1}(Y_i - X_{rob}) \tag{2}$$

4.2. 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" projection model is applied, obtaining $U_{LS}:(u_{LS}, v_{LS})$ and $U_{RS}:(u_{RS}, v_{RS})$ (see [12]); then, the result is "distorted" by means of the distortion models. The corresponding jacobians $\partial U_{LS}/\partial h_i$ and $\partial U_{RS}/\partial h_i$ can be easily calculated from the left projection equations (pin-hole model) and the equivalent ones 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 chapter 7). 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}:

$$h_{iR} = R_{\text{int}}^{-1} \cdot \left(h_i - T_{\text{int}}\right) \tag{3}$$

In equation (3), 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.

4.2.1. 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 [8]). As we have two different image projections, S_i needs to be transformed into the projection covariance P_{U_L} and P_{U_R}.

$$P_{U_L} = \frac{\partial U_L}{\partial h_i} \cdot S_i \cdot \left(\frac{\partial U_L}{\partial h_i}\right)^T ; \; P_{U_R} = \frac{\partial U_R}{\partial h_i} \cdot S_i \cdot \left(\frac{\partial U_R}{\partial h_i}\right)^T \tag{4}$$

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

4.2.2 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 to be taken as a reference. 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."

4.3. 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. These coordinates are related to the measurement vector z_i by means of the so-called *projection equations*, where m_{Lij} and m_{Rij} are the elements of the *projection matrices* M_L and M_R for left and right camera, as we depicts in equation (5).

$$\begin{gathered} u_{LS} = \frac{m_{L11}z_{ix} + m_{L12}z_{iy} + m_{L13}z_{iz} + m_{L14}}{m_{L31}z_{ix} + m_{L32}z_{iy} + m_{L33}z_{iz} + m_{L34}} \; v_{LS} = \frac{m_{L21}z_{ix} + m_{L22}z_{iy} + m_{L23}z_{iz} + m_{L24}}{m_{L31}z_{ix} + m_{L32}z_{iy} + m_{L33}z_{iz} + m_{L34}} \\ u_{RS} = \frac{m_{R11}z_{ix} + m_{R12}z_{iy} + m_{R13}z_{iz} + m_{R14}}{m_{R31}z_{ix} + m_{R32}z_{iy} + m_{R33}z_{iz} + m_{R34}} \; v_{RS} = \frac{m_{R21}z_{ix} + m_{R22}z_{iy} + m_{R23}z_{iz} + m_{R24}}{m_{R31}z_{ix} + m_{R32}z_{iy} + m_{R33}z_{iz} + m_{R34}} \end{gathered} \tag{5}$$

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 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 (3). For the left camera, $M_L = I_{CL}$ because the camera reference frame is actually the left camera reference frame.

From (5) 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.

4.4. 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 (2), 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 (7), 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.

4.4.1. 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 $T_i = \begin{pmatrix} u_L & v_L & u_R & v_R \end{pmatrix}^T$ with the four coordinates of both images. Therefore, the covariance matrix for this vector R_{Ti} 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:

$$C = \begin{pmatrix} (1) & 0 & 0 & 0 \\ 0 & (1) & 0 & 0 \\ 0 & 0 & (2) & 0 \\ 0 & 0 & 0 & (2) \end{pmatrix}, \begin{cases} (1) = -m_{L31} h_{ix} - m_{L32} h_{iy} - m_{L33} h_{iz} - m_{L34} \\ (2) = -m_{R31} h_{ix} - m_{R32} h_{iy} - m_{R33} h_{iz} - m_{R34} \end{cases} \quad (6)$$

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:

$$S = \left[\frac{\partial h}{dX}_{tot} \cdot P \cdot \left(\frac{\partial h}{dX}_{tot} \right)^T \right] + R_{tot} \quad (7)$$

5. 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.

5.1. 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.

5.2. 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 $P_{Y_iY_i}$ needs to be obtained. Unlike the approach used in [8], here we can obtain Y_i just in one step because, as in the measurement process, we can solve the four equations redundant system.

5.2.1. 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: $\begin{pmatrix} a & b & c \end{pmatrix}^T = F \cdot \begin{pmatrix} u_L & v_L & 1 \end{pmatrix}^T$.

5.2.2. 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.

5.2.3. Covariance matrix calculation. The feature position covariance $P_{Y_iY_i}$ depends on two sources: the camera state uncertainty P_{XX} and the feature measurement noise R_i:

$$P_{Y_iY_i} = (\partial Y_i / \partial X_v) P_{XX} (\partial Y_i / \partial X_v)^T + (\partial Y_i / \partial h_i) R_i (\partial Y_i / \partial h_i)^T .$$

We easily obtain $\partial Y_i / \partial h_i$ from (2). 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 (2) as well, but we have to take into account the relationship between the quaternion q_{rob} and the rotation matrix R described in [1].

6. 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.

7. Image distortion model

Due to the use of wide-angle lens, we need to take 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, which are defined in [14] are applied. 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 initial assumption that the distorted projection coordinates are equal to the undistorted ones. The jacobians $\partial U_L / \partial U_{LS}$ and $\partial U_R / \partial U_{RS}$ are calculated from the direct distortion model equations, while the inverse jacobians are also calculated by inverting the previous ones.

8. 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. To check the ability of revisiting "old" features, a 360 degrees turn around sequence was taken. 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. All video sequences used for the test were successful; it means that there was no tracking lost. The video frame showed on Fig 1 was taken at the moment of a new feature initialization. On the corresponding 3D representation (Fig 2), the new feature shows an uncertainty region with an ellipsoid shape like the rest of the features. This means that, as difference as the *mono* implementation, any new feature is completely initialised since the first time it is captured. This allows higher speed changes on the camera movement due to the low initialisation time when loosing features out of the field of view.

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. 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

9. 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.

10. Acknowledgements

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).

11. References

[1] H. Choset and K. 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 ICRA, 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.

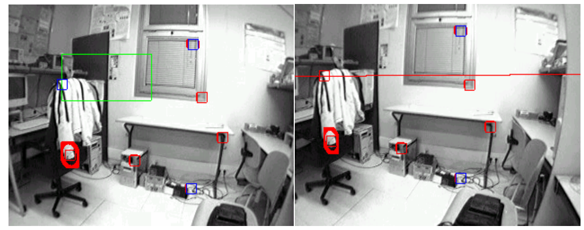

Fig. 3 *Left* and *right* images from the same frame. At this step, a new feature is being captured

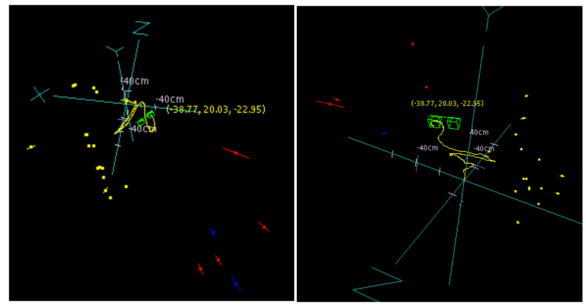

Fig. 4 Two different views of the 3D environment representation corresponding to the projection images showed above.

[5] M. A. Garcia and A. Solanas. “3D Simultaneous Localization and Modeling from Stereo Vision.” *ICRA, pages 847-853, 2004.*

[6] 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.*

[7] A. W. Fitzgibbon. “Simultaneous linear estimation of multiple view geometry and lens distortion.” *In Proc. CVPR, p. 125, 2001.*

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

[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.*

[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.