Towards an alternative GPS sensor in dense urban environment from visual memory

Eric Royer, Maxime Lhuillier, Michel Dhome and Thierry Chateau
LASMEA, UMR6602 CNRS-Université Blaise Pascal
24 Avenue des Landais F-63177 Aubière
{royer,lhuillie,dhome,chateau}@lasmea.univ-bpclermont.fr

Abstract

In this paper we present a method for computing the localization of a mobile robot with reference to a learning video sequence. The robot is first guided on a path by a human, while the camera records a monocular learning sequence. Then the computer builds a map of the environment. This is done by first extracting key frames from the learning sequence. Then the epipolar geometry and camera motion are computed between key frames. Additionally a hierarchical bundle adjustment is used to refine the reconstruction. The map stored for the localization include the position of the camera associated with each key frame as well as a set of interest points detected in the images and reconstructed in 3D. Using this map it is possible to compute the localization of the robot in real time during the automatic driving phase.

1 Introduction

In this paper, we deal with the problem of mobile robot localization with reference to a video sequence. Our goal is to be able to guide a robot manually for some time and to record a reference video sequence. Then the robot should be able to follow the same trajectory by itself. For this application, we have developed a system that builds a three dimensional model of the robot's environnement using only visual data recorded during the learning phase. After that, when the robot is near the learning trajectory, it is possible to use the current frame taken by the camera to localize the robot in real time. The results presented in this paper deal only with the reconstruction of the environment and its use for localization.

A common solution for localization of a robot in outdoor situations is the GPS. But GPS is difficult to use in urban environments because buildings can occult the visibility of satellites so localization may not be possible at some times. Reflections of the GPS signal on the buildings can also occur and corrupt the localization. On the other hand, vision works better if there are a lot of features near the robot. This is the case in downtown areas where GPS doesn't work very well. So these two approaches to localization are complementary. A typical example is the case of an urban canyon where the robot is in a straight and narrow street. In that case all the visible satellites must be in a plane, which is not a good situation to compute the localization.

BMVC 2004 doi:10.5244/C.18.22

A solution for visual navigation with reference to a prerecorded image sequence was presented by Yoshio Matsumoto at al. [10] but the method did not provide the location of the robot at each frame. Seung-Hun Jeon et al. [7] proposed a method that allows to compute the pose of the robot in the case of indoor environments with horizontal and vertical planes. In our case, we want to be able to find a localization in an outdoor environment without assumptions on the geometry of the scene. An approach closer to ours has been proposed by Kiyosumi Kidono et al. [8]. After a human guided phase, the robot builds a map of the environment and uses it to localize itself in autonomous navigation. This system relies both on vision and odometry. Additionally it assumes a calibrated stereo rig for vision data acquisition and movements are done in 2D on a ground plane. In our case we use only one camera and no odometry, the ground doesn't have to be planar. We only suppose that the camera's internal parameters are known very roughly.

The human guided approach is different from the Simultaneous Localization And Mapping (SLAM). For example, Se et al. [13] describe a robot which is able to build a map with the 3D position and appearance of visual landmarks. This map is used for localization. But the landmarks can be observed several times as the robot comes back to a previous position. In our case, the map has to be built from only one human guided path. Moreover the hypothesis are different because we use only one camera instead of a calibrated stereo rig with three cameras.

Map building is the more complex part of the algorithm. Fortunately, this part of the computation can be done off line. So the real time constraint applies only to the localization of the robot. That means that we can use computation intensive algorithms to build a good map. We think that a global and costly optimization step is necessary for practical cases, both for accuracy and robustness under our hypothesis. We can also prepare in advance every computation on the reference sequence that could be needed for the localization process.

In section 2 we present the method used to build the map from the reference image sequence. Then in section 3 the method used to compute the localization of the robot is detailed. Finally we present some results in section 4.

2 Reconstruction of the reference sequence

2.1 Overview

The goal of the reconstruction is to obtain the position of a subset of the cameras in the reference sequence as well as a set of landmarks and their 3D location, all of these given in a global coordinate system. For the reconstruction we use a monocular image sequence and we make the assumption that we roughly know the camera's internal parameters. It allows to use more robust algorithms for the initial computation of the epipolar geometry. It works even if the scene is planar in some images, which is not the case for uncalibrated approaches (for example [1], [12]). Then it is possible to refine the internal parameters in the global optimization of the reconstruction.

Every step in the reconstruction as well as the localization relies on image matching. Matching a pair of images is done by detecting interest points in each image. Harris corner detector [5] is used for this step. Then a Zero Normalized Cross Correlation score is computed between interest points neighborhoods. And the pairs with the best scores are kept to provide a list of corresponding point pairs between the two images. Matching

images this way may sound very time consuming and not really suited to a real time application, but it is possible to implement a very efficient corner detector using SIMD extensions of modern processors.

In the first step of the reconstruction, we extract a set of key frames from the reference sequence. Then we compute the epipolar geometry and camera motion between key frames. Additionally, the interest points used to compute the epipolar geometry are reconstructed in 3D. These points will be the landmarks used for the localization during the on-line phase. They are stored with their neighborhood in the images so that it is possible to match them with interest points detected in new images.

2.2 Key frame selection

Key frame selection is necessary because if there is not enough camera motion between two images, the computation of the epipolar geometry is an ill conditioned problem. So we try to select images so that there is as much camera motion as possible between key frames while still being able to match the images. The first image of the sequence is always selected as a key frame. When image I_n has been selected as a key frame, we match interest points between I_n and the following images $I_{n+1}, I_{n+2}, \ldots$. The wrong matches are filtered out by computing a global dominant homography with Ramdom Sample Consensus (RANSAC) [3] and a large reprojection error threshold (20 pixels). RANSAC is a simple and robust technic for model fitting in presence of data outliers. The next key frame is chosen as the last image with a number of matches above a threshold (300 points). This is an heuristic method but it ensures that we have a large enough number of point matches between key frames. Figure 1 shows three consecutive key frames extracted from the Seaport sequence.

Figure 1: Three successive key frames extracted from the Seaport sequence

2.3 Camera motion computation

At this point we have a subset of the frames from the video sequence recorded during the learning step. The structure and motion algorithm used to build a 3D reconstruction of the scene and camera motion can be separated in three parts. With the first three key frames we compute the camera motion over the three frames by computing an essential matrix. Then for each successive set of three images, the motion is computed using a pose estimation algorithm. These computations produce an initial estimate of the camera motion, and a hierarchical bundle adjustment is used to refine this initial estimation.

For the first image triplet, the computation of the camera motion is done with the method proposed by Nister [11] for three views. It involves computing the essential matrix between the first and last images of the triplet using a sample of 5 point correspondences. Computing the essential matrix E between two images is done by writing the epipolar constraint each of the 5 points projections must verify : $q'^{i^T} E q^i = 0, \forall i \in \{1..5\}$ and a 6^{th} relation that is verified by any essential matrix :$EE^T E - \frac{1}{2} trace(EE^T)E = 0$. These relations lead to a 10^{th} order polynomial equation, so there are at most 10 solutions for E. Each matrix E gives 4 solutions for (R,T). The solutions for which at least one of the 5 points is not reconstructed in front of both cameras are discarded. Then the pose of the remaining camera is computed with 3 out of the 5 points in the sample. This process is done with a RANSAC approach : each 5 point sample produces a number of hypotheses for the three cameras. The best one is chosen by computing the reprojection error over the three views for all the matched interest points and keeping the one with the higher number of inlier matches.

We need an algorithm to compute the pose of the second camera. A review of calibrated pose estimation algorithms is given by Haralick et al. [4]. If the internal parameters of the camera are known, with three 3D points P^i whose projections in the image are known, it is possible to compute the pose of the camera. We chose to use Grunert's method as it is described in [4]. This method relies on trigonometrical computations in the tetrahedron formed by the three 3D points and the optical center of the camera. Relations are computed between the distance of each of the 3D points to the optical center and this leads to a 4^{th} order polynomial equation. There are at most 4 solutions for each sample of three points. The solution is choosen in the RANSAC process.

For the next image triplets, we use a different method for computing camera motion. Assume we know the location of cameras C_1 through C_N, we can compute camera C_{N+1} by using the location of cameras C_{N-1} and C_N and point correspondences over the image triplet $(N-1,N,N+1)$. We match a set of points P^i whose projections are known in each image of the triplet. From the projections in images $N-1$ and N, we can compute the 3D coordinates of point P^i. Then from the set of P^i and their projections in image $N+1$, we use a calibrated pose estimation algorithm to compute the location of camera C_{N+1}. In addition the 3D location of the reconstructed interest points are stored as they will be the landmarks used for the localization process. The pose estimation algorithm is based on Grunert's method and a RANSAC selection process. Random samples of three points are used to compute the pose of camera C_{N+1}. The advantage of a such an iterative pose estimation process is that it can deal with virtually planar scenes.

2.4 Hierarchical bundle adjustment

The computation of camera motion previously presented doesn't give a very good solution. Moreover, the computation of camera C_N depends on the results of the previous cameras and errors can build up over the sequence. In order to correct this problem, we use a bundle adjustment algorithm which provides a better solution. The bundle adjustment is a Levenberg-Marquardt minimization of the cost function $f(C_I, C_E^1, \cdots, C_E^N, P^1, \cdots, P^M)$ where C_I are the internal parameters of the camera (they are constant throughout the sequence), C_E^i are the external parameters of camera i, and P^j are the world coordinates of point j. The cost function is the sum of the reprojection errors of all the inlier reprojec-

tions in all the images :

$$f(C_I, C_E^1, \cdots, C_E^N, P^1, \cdots, P^M) = \sum_{1 \leq i \leq N} \sum_{1 \leq j \leq M, j \in J_i} d^2(p_i^j, K_i P^j)$$

where $d^2(p_i^j, K_i P^j)$ is the squared euclidean distance between $K_i P^j$ the projection of point P^j by camera i, and p_i^j is the corresponding detected point. K_i is the 3×4 projection matrix built from the parameters values in C_I and C_E^i. And J_i is the set of points whose reprojection error in image i is less than 2 pixels at the beginning of the minimisation. After a few iteration steps, J_i is computed again and more minimization iterations are done. This inlier selection process is repeated as long as the number of inliers increase.

It's not a good idea to compute all the camera locations and use the bundle adjustment only once on the whole sequence. In that case, increasing errors could produce an initial solution too far from the optimal one for the bundle adjustment to converge. Thus it is necessary to use the bundle adjustment throughout the reconstruction of the sequence. Using an adjustment after each new frame is reconstructed is possible but very time consuming, and impractical for large sequences. A faster solution as described in [6] is to use the adjustment hierarchically. A large sequence is divided into two parts with an overlap of two frames in order to be able to merge the sequence. Each subsequence is recursively divided in the same way until each final subsequence contains only three images. Each image triplet is processed as described above. For the first triplet we obtain the geometry by computing an essential matrix. Each remaining triplet has its first two frames in common with the previous one. So the first two cameras of the triplet are deduced from the previous triplet. The third camera is computed using the pose estimation algorithm. After each triplet has been computed we run a bundle adjustment over its three frames.

In order to merge two sequences S^1 and S^2, we use the last 2 cameras S_{N-1}^1 and S_N^1 of S^1 and the first 2 cameras S_1^2 and S_2^2 of S^2. As the images are the same, the cameras associated after merging must be the same. So we apply a rotation and a translation to S^2 so that S_N^1 and S_2^2 have the same position and orientation. Then the scale factor is computed so that $d(S_{N-1}^1, S_N^1) = d(S_1^2, S_2^2)$, where $d(S_n^i, S_m^j)$ is the euclidean distance between the optical centers of the cameras associated with S_n^i and S_m^j. This doesn't ensure that S_{N-1}^1 and S_1^2 are the same, so a bundle adjustment is used on the result of the merging operation. Merging is done until the whole sequence has been reconstructed. The reconstruction ends with a global bundle adjustment. It should be noted that the final bundle adjustment optimizes also the focal length (known only roughly at the beginning). The number of points used in the bundle adjustment is on the order of five to ten thousands.

3 Localization

In order to locate the robot at the current frame, the first step is to retrieve the closest key frame in the database. Interest points are detected in the current frame and the points are matched to the interest points associated with each reference image. Incorrect matches are filtered out by computing the global dominant homography with a RANSAC process with a loose error threshold (20 pixels). The chosen key frame is the one with the greatest number of inlier matches. At the start of the localization, it is necessary to match the current image with all the key frames. After this step, we always have a localization for

the robot, so we need only to match the current image to the last used key frame and the next one.

At this point we have a set of point matches between the current frame and a key frame in the database. For each interest point of the key frame, we know its 3D position since it has been computed in the reconstruction process. So we have a set of matches between the 2D points in the current image and 3D points in the world coordinate system. The current pose of the camera is then computed using the same perspective pose estimation algorithm as in the reconstruction : Grunert's method used in a RANSAC process. We need random samples of only three points, so this computation doesn't take a lot of computing power. The solution we get from this algorithm is then refined with an iterative minimization algorithm developed by Lowe [9] and improved by Dhome et al. [2]. The minimization is done with a Newton-Raphson algorithm with the cost function $g(R_{\alpha,\beta,\gamma}, T_{u,v,w})$ where the six parameters characterize the rotation and translation between the world and camera coordinates systems. The cost fucntion is :

$$g(R_{\alpha,\beta,\gamma}, T_{u,v,w}) = \sum_{i=1}^{i=N} (d^2(P_i, N_i^{col}) + d^2(P_i, N_i^{row}))$$

where $d^2(P_i, N_i^{col})$ (resp. $d^2(P_i, N_i^{row})$) is the squared euclidean distance between P_i and the plane passing through the optical center and containing the image column (resp. row) of the projection of point P_i.

4 Results

In order to evaluate the localization results, two cameras were mounted on a car (one on the right side and one on the left) in order to record two video sequences at the same time. The reconstruction was computed using only the images from the right sequence. Then, the left sequence was used as a test sequence to simulate a small lateral localization error of the robot. The images from the left sequence were localized with the algorithm described previously and the right sequence serving as the reference sequence.

A few images from the Seaport sequence (right camera) are displayed on figure 2. For this sequence 76 key frames were selected among the 500 images of the complete video. The total length of the path is about 100 meters. On this sequence, a few persons were walking on the side of the street but the interest points on these persons were correctly rejected in the RANSAC process and it did not affect the reconstruction. The reconstruction appears on figure 2 as seen from above. The squares represent the position of the cameras of the key frames, and the dots are the landmarks reconstructed (there are 9698 of them for the Seaport sequence). There are more key frames when there is a turn because the interest points go out of the field of view faster. Using the reconstruction of the right sequence, both sequences were used with the localization process to produce the trajectories of the left and right cameras. For each image in the sequence, we computed the distance between the estimated positions of the left camera and the right camera. Since the length unit in the reconstruction is arbitrary, we set it so that the mean of the distance between the right and left cameras equals the real distance which was about 1.2m. The computation gave some variations with a standard deviation of about 25 cm.

Other sequences have been used to test our algorithm. The number of features and the result of the localization are detailed in table 1. The Canyon sequence is a good example

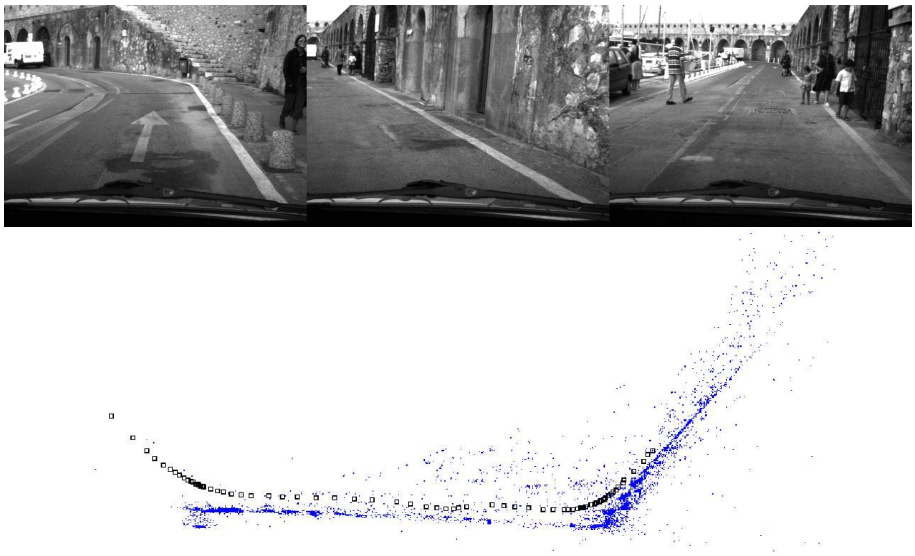

Figure 2: Three images from the Seaport sequence and the associated reconstruction

of a urban canyon where GPS can't be used for localization because the street is narrow and GPS signals are blocked by buildings. In that kind of place, there is a lot of visual data available for localization. In this sequence there are two speed bumps. It illustrates why in the case of urban navigation, it is not possible to make the assumption that the ground is planar. Three images of the sequence and the associated reconstruction are shown on figure 3. The distance between the left and right cameras for the Canyon sequence appear on figure 4. The standard deviation is about 5 cm.

Figure 3: Three images from the Canyon sequence and the associated reconstruction

The Road sequence was made on a wide road in an environment with less visual features. It is still possible to build a reconstruction of the trajectory and to compute a

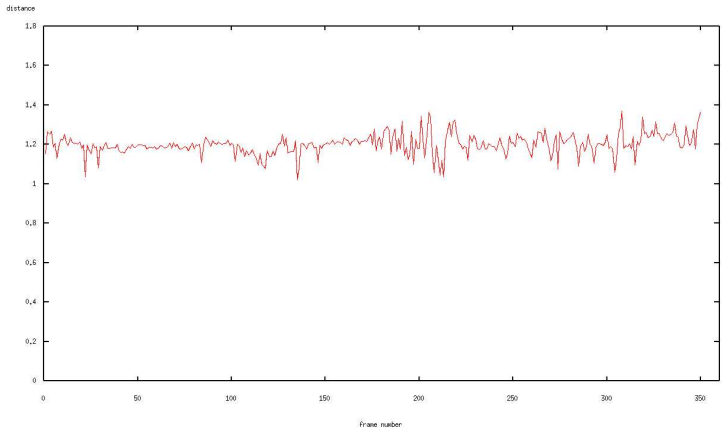

Figure 4: Distance between the left and right cameras for the Canyon sequence

localization. A few images are visible on figure 5 along with the reconstruction.

The distance between the right and left cameras computed with the localisation process was computed for each of the three sequences. The standard deviation appears in table 1. The localization works best in the Canyon sequence where there is much more visual information. In this sequence, a lot of points are detected on the walls on each side of the street, whereas in the Seaport sequence only one wall is visible. That's important because points which are closer to the cameras offer more information for the localization than points far away. At the extremes, points at infinity are only useful for computing the orientation of the camera. In the Road sequence, no points are visible near the vehicle. This explains why localization is worse than in the other sequences. In dense urban areas, these results are acceptable for commanding a robot. In more open environments, the results lack some accuracy, but those are only raw results of the localization algorithm. The results should be better when coupled with a Kalman filter or when using a movement model of the robot.

Sequence	Seaport	Canyon	Road
Number of images	500	380	360
Number of key frames	76	60	33
Number of 3D points	9698	4633	2611
Standard deviation	25 cm	5 cm	50 cm

Table 1: Number of frames and points and standard deviation for the distance between the left and right cameras

For these examples, the image size was 728x532 pixels, and 1500 interest points were detected in each frame. The whole localization process runs at about 5 Hz on a Pentium 4 2.4GHz processor. Detecting interest points in the curent frame takes about 40 ms, matching takes 100 ms, choosing the closest key frame takes 20 ms, and computing the

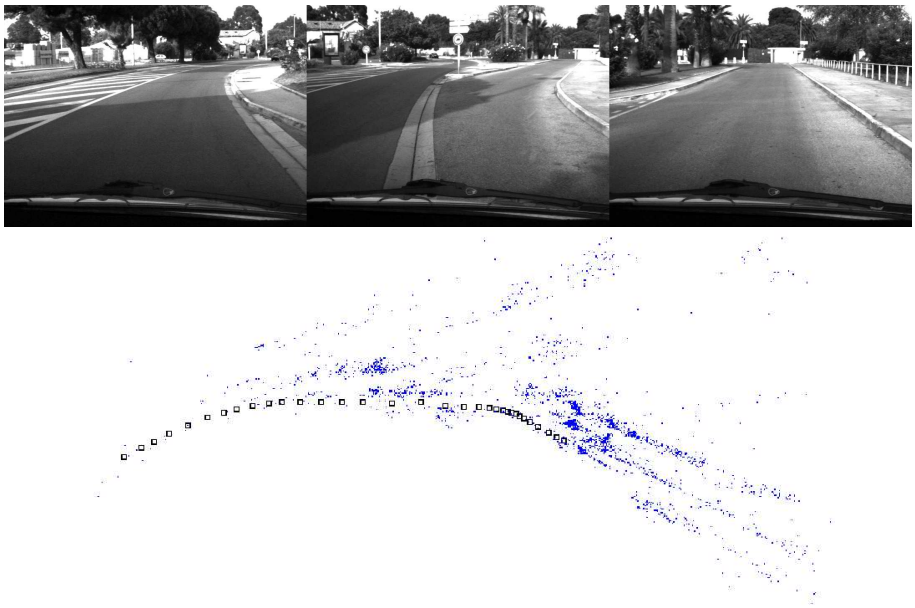

Figure 5: Three images from the Road sequence and the associated reconstruction

pose takes again 20 ms. The time spent for detecting the points depends on image size, the time for matching depends on the number of interest points detected in each frame. With smaller images (360x266 pixels) the whole localization process takes about 70 ms and with some optimizations of the code it should be possible to reach the video frame rate (40 ms per image).

We are grateful to ICARE project team at INRIA Sophia Antipolis for providing the video sequences.

5 Conclusion

We have presented a real time localization algorithm for a mobile robot with reference to a learning sequence. The robot is first guided on a path by a human, while the camera records the learning sequence. Then the computer builds a map of the environment. With this map, it is possible to compute the localization of the robot. We have shown some maps obtained on a few sequences.

The project is just beginning so we have plans to improve the results in the near future. We also plan to make a comparison between the results obtained from our vision algorithm with the ground truth measured using a differential GPS sensor. In order to improve the localization process, we would like to improve the matching method. Some possibilities would be real time wide baseline matching, or a matching method robust to weather and season changes.

References

[1] P. Beardsley, P. Torr and A. Zisserman. *3D Model acquisition from extended image sequences*. Proceedings of the 4th European Conference on Computer Vision (also LNCS 1065), Cambridge, England, pages 683-695, April 1996.

[2] M. Dhome, M. Richetin and J.-T. Lapreste. *Determination of the Attitude of 3D Objects from a Single Perspective View*. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1265-1278, vol. 11, Number 12, 1989.

[3] M.A. Fischler and R.C. Bolles. *Random Sample Consensus: a paradigme for model fitting with application to image analysis and automated cartography.* Communications of the ACM, Volume 24, Issue 6, pages 381-395, June 1981

[4] Robert M. Haralick, Chung Nan Lee, Karsten Ottenberg, Michael Nolle. *Review and analysis of solutions of the three point perspective pose estimation problem*. International Journal of Computer Vision, 1994.

[5] C. Harris, M. Stephens, *A Combined Corner and Edge Detector.* Alvey Vision Conference, pages 147-151, 1988.

[6] R. Hartley, A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[7] Seung-Hun Jeon, Byung Kook Kim. *Monocular-based Position Determination for Indoor Navigation of Mobile Robots*. in Proc. of the 1999 IASTED international conference, 1999.

[8] Kiyosumi Kidono, Jun Miura, Yoshiaki Shirai. *Autonomous Visual Navigation of a Mobile Robot Using a Human-Guided Experience*. Robotics and Autonomous Systems, Vol. 40, Nos. 2-3, pp 124-1332, 2002.

[9] D.G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.

[10] Yoshio Matsumoto, Masayuki Inaba, Hirochika Inoue. *Visual Navigation using View-Sequenced Route Representation*. In Proceedings of IEEE Conference on robotics and Automation, pages 83-88, 1996.

[11] D. Nister, *An efficient solution to the five-point relative pose problem* 2003 Conference on Computer Vision and Pattern Recognition - volume II, IEEE, June 2003

[12] M. Pollefeys, R. Koch and L. Van Gool. *Self-Calibration and metric reconstruction in spite of varying and unknown internal camera parameters.* Proceedings of the 6th International Conference on Computer Vision, pages 90-95, January 1998.

[13] Stephen Se, David Lowe and Jim Little. *Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks.* International Journal of Robotic Research, volume 21, pages 735-758, August 2002.