# PERFORMANCE EVALUATION OF A LOCALIZATION SYSTEM RELYING ON MONOCULAR VISION AND NATURAL LANDMARKS

Eric Royer[†] [*], Maxime Lhuillier[†] , Michel Dhome[†], Jean-Marc Lavest[†]

[†]LASMEA UMR6602 CNRS et Université Blaise Pascal, 24 avenue des Landais - 63177 AUBIERE Cedex
Eric.ROYER@lasmea.univ-bpclermont.fr — http://www.lasmea.univ-bpclermont.fr/Personnel/Eric.Royer/

**KEY WORDS:** Vision, Reconstruction, Performance, Real-time, Navigation

**ABSTRACT:**

We present a real-time localization system based on monocular vision and natural landmarks. In a learning step, we record a reference video sequence and we use a structure from motion algorithm to build a model of the environment. Then in the localization step, we use this model to establish correspondences between the 3D model and 2D points detected in the current image. These correspondences allow us to compute the current camera localization in real-time. The main topic of this paper is the performance evaluation of the whole system. Four aspects of performance are considered : versatility, accuracy, robustness and speed.

## 1 INTRODUCTION

In this paper we evaluate the performance of an algorithm designed to compute the localization of a camera in real-time. Only one camera and natural landmarks are required. In a first step, we record a video sequence along a trajectory. Then this sequence goes through a structure from motion algorithm to compute a sparse 3D model of the environment. When this model has been computed, we can use it to compute the localization of the camera in real-time as long as the camera stays in the neighborhood of the reference trajectory. We have developed this system for outdoor autonomous navigation of a robotic vehicle, but other applications such as indoor robotics or augmented reality can use the same localization system. The main topic of the paper is the performance evaluation of the localization system. The algorithm is only briefly presented here, more details can be found in (Royer et al., 2005).

As soon as a map of the environment is available, it is possible to compute a localization for the camera with reference to the map. Several approaches for building the map are possible. Simultaneous Localization And Mapping (SLAM) is very attractive because localization is possible as soon as the system starts working. But map building is the most computer intensive part, so doing this with monocular vision in real-time is difficult. However, monocular SLAM has been achieved in real-time (Davison, 2003). But the main drawback is that it's not possible to handle a large number of landmarks in the database. Computing a localization from the video flow can also be done by ego-motion estimation or visual odometry (Nistér et al., 2004). But this method is subject to error accumulation because there is no global optimization and the localization accuracy decreases with the distance covered.

Another possible approach is to build a map first and use this map to localize the camera. The main advantage is that there is no real-time constraint on map building. So algorithms providing more accuracy can be used. This approach has been used several times for robot localization. Cobzas et al. (2003) use a camera mounted on a rotating platform and a laser range finder to build a panoramic image enhanced with 3D data of the environment. After the 3D model is built, a single 2D image is enough to compute the localization of the camera. Kidono et al. (2002) also use a map building step before the localization. Map building consists in recording the video sequence along a reference trajectory, then localization is possible in the neighborhood of this trajectory as in our method. It works under the assumption that the ground is planar and the sensors used are a stereo vision rig and an odometer. In our case, the ground can be irregular and we use only one calibrated camera. Camera calibration is important in order to use fish eye lenses with up to $130°$ field of view. Map building is done with a structure from motion algorithm.

In section 2 we briefly present the algorithms we use to build the map from the reference video sequence, and how this map is used for the localization process. In section 3 we show some localization results and we discuss the performance of the system. Four aspects of performance are considered : versatility, accuracy, robustness and speed. The results come from experiments carried out indoors and outdoors. The results provided by the vision algorithm are compared to the ground truth whenever possible.

## 2 ALGORITHM

### 2.1 Map building

Every step in the reconstruction as well as the localization relies on image matching. Interest points are detected in each image with Harris corner detector (Harris and Stephens, 1988). For each interest point in image 1, we select some candidate corresponding points in a rectangular search region in image 2. Then a Zero Normalized Cross Correlation score is computed between their neighborhoods, and the pairs with the best scores are kept to provide a list of corresponding point pairs between the two images. This matching method is sufficient when the camera doesn't rotate much around the optical axis which is the case when the camera is mounted on a wheeled robot. Matching methods with rotational invariance might be used depending on the application but they would require more computing power.

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 in a global coordinate system. The structure from motion problem has been studied for several years and multiple algorithms have been proposed depending on the assumptions we can make (Hartley and Zisserman, 2000). For our experiments, the camera was calibrated using a planar calibration

[*]Corresponding author.

pattern (Lavest et al., 1998). Camera calibration is important because the wide angle lens we use has a strong radial distortion. With a calibrated camera, the structure from motion algorithm is more robust and the accuracy of the reconstruction is increased. In our robotic application, the motion is mostly along the optical axis of the camera. Point triangulation must be done with small angles, which increases the difficulty of obtaining an accurate 3D reconstruction.

In the first step of the reconstruction, we extract a set of key frames from the reference sequence. Then we compute camera motion between key frames. Additionally, the interest points are reconstructed in 3D. These points will be the landmarks used for the localization process.

**2.1.1 Key frame selection** If there is not enough camera motion between two frames, the computation of the epipolar geometry is an ill conditioned problem. So we 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 the first key frame $I_1$. The second key frame $I_2$ is chosen as far as possible from $I_1$ but with at least $M$ common interest points between $I_1$ and $I_2$. When key frames $I_1 \ldots I_n$ are chosen, we select $I_{n+1}$ (as far as possible from $I_n$) so that there is at least $M$ interest points in common between $I_{n+1}$ and $I_n$ and at least $N$ common points between $I_{n+1}$ and $I_{n-1}$. In our experiments we detect 1500 interest points per frame and we choose $M = 400$ and $N = 300$.

**2.1.2 Camera motion computation** For the first three key frames, the computation of the camera motion is done with the method given by Nistér (2003) 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. There are at most 10 solutions for $E$. Each matrix $E$ gives 4 solutions for camera motion. 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 (Fischler and Bolles, 1981) approach : each 5 point sample produces a number of hypothesis for the 3 cameras. The best one is chosen by computing the reprojection error over the 3 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. With a calibrated camera, three 3D points whose projections in the image are known are enough to compute the pose of the camera. Several methods are compared by Haralick et al. (1994). We chose Grunert's method with a RANSAC approach.

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 $X^i$ 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 $X^i$. Then from the set of $X^i$ and their projections in image $N+1$, we use Grunert's calibrated pose estimation algorithm to compute the location of camera $C_{N+1}$. In addition the 3D locations of the reconstructed interest points are stored because they will be the landmarks used for the localization process. The advantage of this iterative pose estimation process is that it can deal with virtually planar scenes. After the pose computation, a second matching step is done with the epipolar constraint based on the pose that has just been computed. This second matching step allows to increase the number of correctly reconstructed 3D points by about 20 %.

**2.1.3 Hierarchical bundle adjustment** 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 which provides a better solution. The bundle adjustment is a Levenberg-Marquardt minimization of the cost function $f(C_E^1, \cdots, C_E^N, X^1, \cdots, X^M)$ where $C_E^i$ are the external parameters of camera $i$, and $X^j$ are the world coordinates of point $j$. For this minimization, the radial distortion of the 2D point coordinates is corrected beforehand. The cost function is the sum of the reprojection errors of all the inlier reprojections in all the images :

$$f(C_E^1, \cdots, C_E^N, X^1, \cdots, X^M) = \sum_{i=1}^{N} \sum_{j=1, j \in J_i}^{M} d^2(x_i^j, P_i X^j)$$

where $d^2(x_i^j, P_i x^j)$ is the squared euclidian distance between $P_i X^j$ the projection of point $X^j$ by camera $i$, and $x_i^j$ is the corresponding detected point. $P_i$ is the $3 \times 4$ projection matrix built from the parameters values in $C_E^i$ and the known internal parameters of the camera. And $J_i$ is the set of points whose reprojection error in image $i$ is less than 2 pixels at the beginning of the minimization. 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 increases.

Computing all the camera locations and use the bundle adjustment only once on the whole sequence could cause problems because 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. So we use the adjustment hierarchically (Hartley and Zisserman, 2000). 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 in section2.1.2. After each triplet has been computed we run a bundle adjustment over its three frames. Then we merge small subsequences into larger subsequences and we use a bundle adjustment after each merging operation. In order to merge two subsequences, we compute a best-fit rigid transformation so that the first two cameras of the second subsequence are transformed into the last two cameras of the first subsequence. Merging is done until the whole sequence has been reconstructed. The reconstruction ends with a global bundle adjustment. The number of points used in the bundle adjustment is on the order of several thousands.

## 2.2 Real-time localization

The output of the learning process is a 3D reconstruction of the scene : we have the pose of the camera for each key frame and a set of 3D points associated with their 2D positions in the key frames. At the start of the localization process, we have no assumption on the vehicle localization. So we need to compare the current image to every key frame to find the best match. This is done by matching interest points between the two images and computing a camera pose with RANSAC. The pose obtained with the higher number of inliers is a good estimation of the camera pose for the first image. This step requires a few seconds but is needed only at the start. After this step, we always have an approximate pose for the camera, so we only need to update the pose and this can be done much faster.

The current image is noted $I$. First we assume that the camera movement between two successive frames is small. So an approximate camera pose (we note the associated camera matrix

$P_0$) for image $I$ is the same as the pose computed for the preceding image. Based on $P_0$ we select the closest key frame $I_k$ in the sense of shortest euclidian distance between the camera centers. $I_k$ gives us a set of interest points $A_k$ reconstructed in 3D. We detect interest points in $I$ and we match them with $A_k$. To do that, for each point in $A_k$, we compute a correlation score with all the interest points detected in $I$ which are in the search region. For each interest point in $A_k$ we know a 3D position, so with $P_0$ we can compute an expected position of this point in $I$. In the matching process the search region is centered around the expected position and its size is small (20 × 12 pixels). After this matching is done, we have a set of 2D points in image $I$ matched with 2D points in image $I_k$ which are themselves linked to a 3D point obtained during the reconstruction process. With these 3D/2D matches a better pose is computed using Grunert's method through RANSAC to reject outliers. This gives us the camera matrix $P_1$ for $I$. Then the pose is refined using the iterative method proposed by Araújo et al. (1998) with some modifications in order to deal with outliers. This is a minimization of the reprojection error for all the points using Newton's method. At each iteration we solve the linear system $J\delta = e$ in order to compute a vector of corrections $\delta$ to be subtracted from the pose parameters. $e$ is the error vector formed with the reprojection error of each point in $x$ and $y$. $J$ is the Jacobian matrix of the error. In our implementation, the points used in the minimization process are computed at each iteration. We keep only the points whose reprojection error is less than 2 pixels. As the pose converges towards the optimal pose, some inliers can become outliers and conversely. Usually, less than five iterations are enough.

## 3 PERFORMANCE EVALUATION

### 3.1 Versatility

This localization system was used with several cameras in different kind of environments. We used normal and fish eye lenses with a field of view ranging from 50° to 130°. The localization system is performing well both indoors and outdoors with changing weather conditions (cloudy, sunny, or with snow on the ground) with a single learning sequence. According to the environment we used different methods to evaluate the accuracy and the robustness of the algorithm. The results of these experiments are detailed in the following paragraphs.

### 3.2 Accuracy

**3.2.1 Indoor experiments** To evaluate the accuracy of the localization we used a table where we could measure the position of the camera with a 1 millimeter accuracy in a 1.2 $m$ × 1.0 $m$ rectangle. We first recorded a reference video sequence on the left side of the table. The trajectory was a 1.2 m long straight line oriented along the optical axis of the camera ($Z$). Figure 1 illustrates the setup with two images taken on each side of the localization area (1 m apart). Another pair of such images is present on Figure 9. Most of the objects visible were along the wall of the room which was about 3.5 m in front of the localization area. There were 13 key frames and we built a 3D reconstruction from these images. Then we moved the camera by 10 cm increments in $X$ or $Z$ in the localization area in order to cover the whole rectangle. For each position we ran the localization algorithm and compared the position given by the vision algorithm to the true position measured on the table. This gave us 131 measurements: the position error $e_{i,j}$ was made for $X = 0.1i$ and $Z = 0.1j$ for each $(i, j) \in \{0..11\} \times \{0..10\}$. For each lateral deviation ($X = constant$) we computed the average value of the error and the standard deviation. The result is shown on Figure 2. As

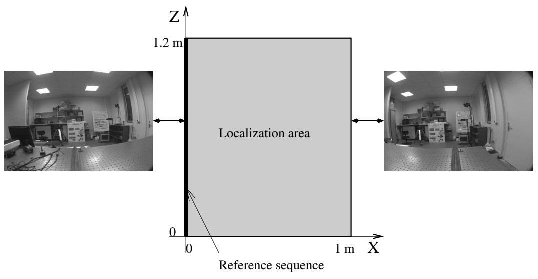

Figure 1: Setup for the indoor experiment

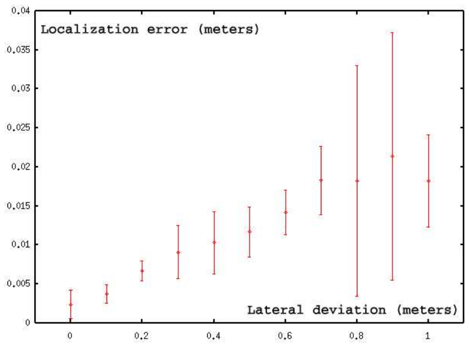

Figure 2: Localization error for a given lateral deviation (average value and standard deviation)

long as we stay on the reference trajectory, the localization error is only a few millimeters. The order of magnitude of the error depends on the distance of the observed 3D points. The outdoor experiments show a ten fold increase in localization error because the objects observed can be at 30 m rather than 3 m.

We also made an experiment to evaluate the rotational accuracy. The camera was mounted on a rotating platform. The angle of the platform can be read with about ±0.1° accuracy. We compared the orientation $\alpha$ provided by the vision algorithm to the angle $\alpha_0$ given by the platform. We used the same fish eye lens as in the previous experiment, providing a 130° field of view (in the diagonal) and we made a measurement for each angle from $\alpha_0 = -94°$ to $\alpha_0 = 94°$ with a 2° increment. The reference trajectory was a straight line (1 m long) oriented along the optical axis (which was in the 0° direction). The result of this experiment appears on Figure 3. The algorithm was not able to provide the pose of the camera when the angle reached 95° because there were not enough point correspondences. The angular accuracy measured with this setup is about ±0.1°, which is about the same as what can be read on the platform. The algorithm provides a useful angular information for a deviation up to 94° on either side with this camera. Of course, with such an angular deviation from the reference frame, the part of the image which can be used is very small, and the localization becomes impossible if there is an occultation in this area. Images captured for 0°, 45° and 90° are shown on Figure 4.

**3.2.2 Outdoor experiment** For outdoor situations, the camera is mounted on the roof of a robotic vehicle along with a Differential GPS (DGPS) sensor to record the ground truth. According to the manufacturer, the DGPS has an accuracy of 1 cm in an horizontal plane (it is only 20 cm along a vertical axis with our hardware). Measuring the accuracy of our algorithms is not straightforward. Two operations are needed so that both data sets can be compared. First the GPS sensor is not mounted on the vehicle at the same place as the camera. The GPS is located at the

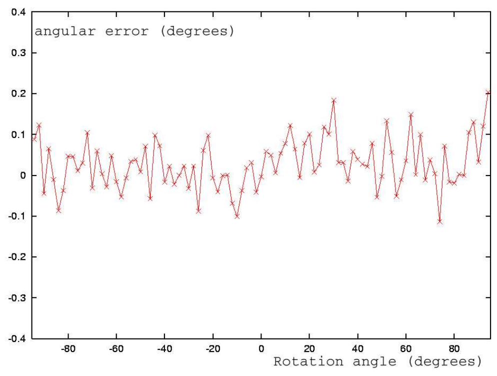

Figure 3: Angular error

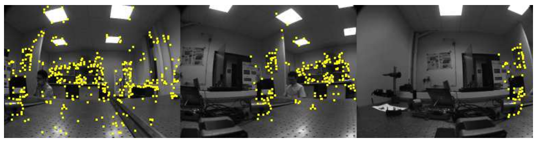

Figure 4: From left to right images taken at $0^\circ$, $45^\circ$ and $90^\circ$ orientation, with interest points correctly matched

mid-point between the rear wheels of the car, while the camera is between the front wheels. So the two sensors don't have the same trajectory. From the GPS positions, we computed a "virtual" GPS which indicates what a GPS would record if it was at the same place as the camera. In addition, the 3D reconstruction is done in an arbitrary euclidian coordinate system, whereas the GPS positions are given in another coordinate system. So the whole 3D reconstruction has to be transformed using a rotation, translation and scale change. The approach described by Faugeras and Herbert (1986) is used to compute this transformation. After these transformations have been made, for each camera we are able to compute the error on the position in meters. Because of the lack of accuracy of the DGPS along the vertical axis, all the localization errors reported for the outdoor experiments are measured in an horizontal plane only.

Four sequences called $outdoor_1$ through $outdoor_4$ were recorded by driving manually the vehicle along a 80 m trajectory. The four sequences were made approximately on the same trajectory ( with at most a 1 m lateral deviation), the same day. Each sequence was used in turn as the reference sequence. So we made twelve experiments : we computed a localization for $outdoor_i$ using $outdoor_j$ as the reference sequence for each $j \in \{1, 2, 3, 4\}$ and $i \neq j$. A few images extracted from $outdoor_1$ are shown in Figure 5. The positions of the key frames computed from this sequence are shown in Figure 6 (as seen from the top) along with the trajectory recorded by the DGPS. Depending on the sequence, the automatic key frame selection gave between 113 and 121 key frames. And at the end of the reconstruction there were between 14323 and 15689 3D points.

We define two errors to measure the reconstruction and the localization accuracy. We want to distinguish the error that is at-

Figure 5: A few images from $outdoor_1$

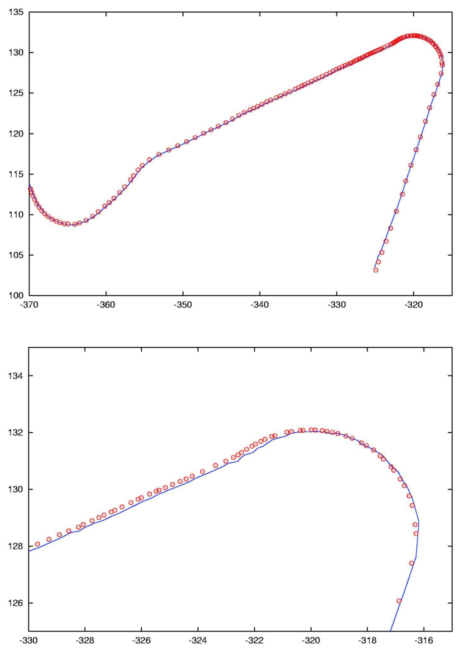

Figure 6: Position of the key frames (circles) with reference to the trajectory recorded by the DGPS (continuous line). Whole trajectory on top and close up view at the bottom (units in meters)

tributed to the reconstruction algorithm and the error coming from the localization algorithm. The reconstruction error is the average distance between the camera positions obtained from the structure from motion algorithm and the true positions given by the DGPS (after the two trajectories have been expressed in the same coordinate system). The reconstruction error for each of the sequences was 25 cm, 40 cm, 34 cm and 24 cm for a 80 m long trajectory with two large turns. This error is mostly caused by a slow drift of the reconstruction process. It increases with the length and complexity of the trajectory. That means the 3D model we build is not perfectly matched to the real 3D world and computing a global localization from this model would give at least about 30 cm of error.

However, in many applications, a global localization is not required. For example, in our application a robot needs to compute a self-localization so that it is able to follow the reference trajectory. In this case, we only need to compute the distance between the current robot position and the reference trajectory as well as the angular deviation from the reference trajectory. A global localization is not necessary, only a relative position with respect to the reference trajectory is needed. We define the localization error in order to measure the error we make in computing this relative localization with the vision algorithm. We need a somewhat more complicated definition for the localization error. First we compute the lateral deviation between the current robot position and the closest robot position on the reference trajectory. This is illustrated on Figure 7. The robot position is always defined by the position of the middle point of the rear axle of the vehicle. This position is directly given by the DGPS. When working with vision it must be computed from the camera position and orientation. First we apply a global scale to the 3D reconstruction so that the scale is the same between the GPS data and vision data. We start with the localization of the camera $C_1$ given by the localization part of the vision algorithm. From $C_1$ we compute the corresponding GPS position $G_1$ (it is possible because we measured the positions of the GPS receiver and the camera on the vehicle). Then we find the closest GPS position in the reference

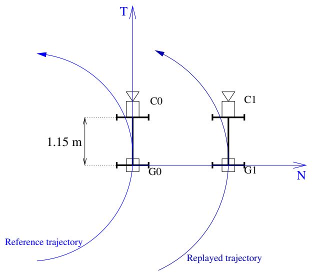

Figure 7: Computing the lateral deviation from the reference trajectory

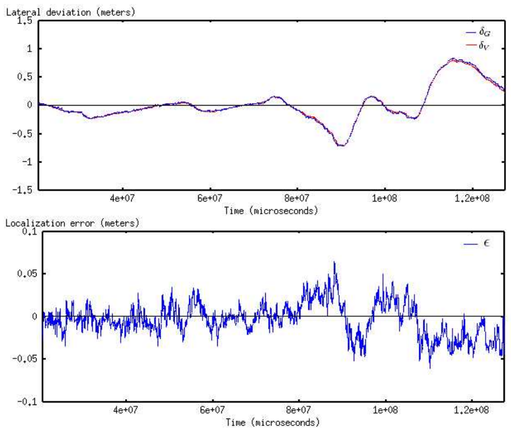

Figure 8: Lateral deviation (top) measured with the DGPS $\delta_G$ (blue) or with vision $\delta_V$ (red) and localization error $\epsilon$ (bottom)

trajectory : we call it $G_0$. At point $G_0$ of the reference trajectory, we compute the tangent $\overrightarrow{T}$ and normal $\overrightarrow{N}$ to the trajectory. The lateral deviation computed with vision is $\delta_V = \overrightarrow{G_0G_1} \cdot \overrightarrow{N}$. The lateral deviation is computed from the GPS measurements as well and we get $\delta_G$ (in this case we have directly $G_0$ and $G_1$). $\delta_G$ and $\delta_V$ are the same physical distance measured with two different sensors. Then the localization error is defined as $\epsilon = \delta_V - \delta_G$. From this we can compute the standard deviation of $\epsilon$ for a whole trajectory : we call this the average localization error.

We computed the average localization error for each of the twelve experiments : the smallest was 1.4 cm, the largest was 2.2 cm and the mean over the twelve videos was 1.9 cm. Figure 8 shows the lateral deviation and localization error for one experiment with a 1.9 cm average localization error. To make sure that it is a valid method to measure the localization accuracy, we used a control law to drive the robotic vehicle. We used in turn the GPS sensor and the vision algorithm to control the robot. Both methods allowed to drive the robot with the same accuracy (4 cm in straight lines and less than 35 cm lateral deviation in curves for both sensors). This shows that the accuracy of the GPS and the vision algorithm is equivalent for the autonomous navigation application. The error can be attributed more to the difficulty of controlling the robot than to the localization part.

## 3.3 Robustness

**3.3.1 Indoor experiment** We made two experiments to evaluate the robustness of the localization algorithm. First, we made no change to the environment between the reference sequence and the localization step, but up to 6 persons went in front on the camera to mask a part of the scene. In the second experiment, we started the localization process with the same environment as in the reference sequence and we gradually modified the scene.

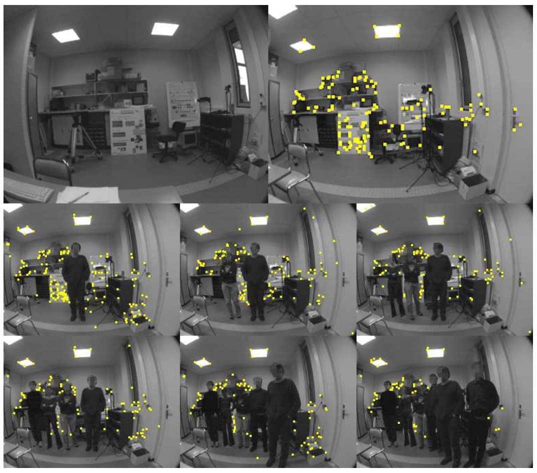

Figure 9: Images for the off-axis occultation experiment. Top left : reference image on axis, top right : off-axis image with no occultation. Second and third rows : occultation by 1 to 6 persons

| Number of persons | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| position error on axis (mm) | 2 | 1 | 1 | 1 | 1 | 1 | 2 |
| position error off axis (mm) | 8 | 11 | 4 | 11 | 20 | 44 | 132 |

Table 1: Localization error for the occultation experiment

We moved or removed some objects, changed the illumination, and added some occultations. The modifications were made in 8 steps. For both experiments, we recorded the error between the computed localization and the true localization. We did this for two different camera positions : one on the reference sequence (on axis) and one for a position with 1 m lateral deviation from the reference trajectory (off axis). The reference trajectory was the same as in the indoor accuracy experiment. Figure 10 shows the closest key frame found and some of the images for which the localization was computed. Correctly identified interest points are also drawn. Figure 9 shows the images used in the off axis occultation experiment. The localization error is given in Table 1 for the occultation experiment and in Table 2 for the scene modification experiment. These results show that the algorithm is robust to large changes in the environment (modifications of the scene, occultations and changing light conditions). The reason is that we have a large number of features stored in the database and only a few of them are needed to compute an accurate localization. Moreover the constraints on feature matching are severe enough so that additional objects that are added to the scene are not taken erroneously as inliers. The performance degradation is visible only with a large lateral deviation and strong changes to the environment.

**3.3.2 Outdoor experiments** For outdoors use, a localization system must be robust to changes in illumination and weather. Since the system was developed, we have had the opportunity to

| Modification step | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| error on axis (mm) | 1 | 1 | 2 | 0 | 2 | 5 | 2 | 5 |
| error off axis (mm) | 29 | 16 | 18 | 24 | 51 | 100 | 21 | 183 |

Table 2: Localization error for the scene modification experiment

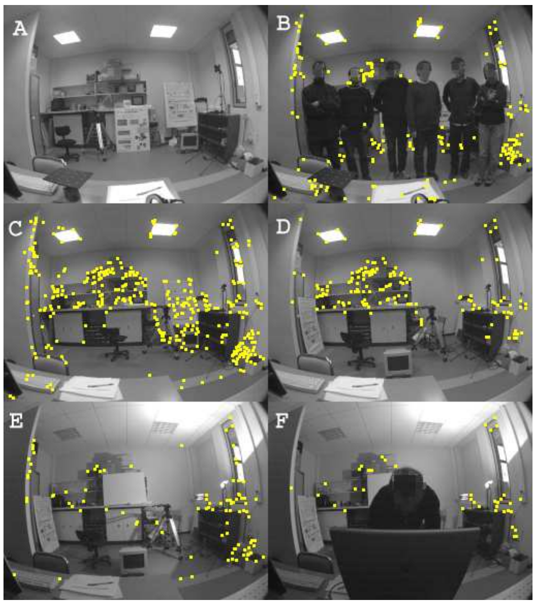

Figure 10: Images for robustness evaluation on axis : original image (A), occultation by 6 persons (B), modifications step 2 (C), step 4 (D), step 6 (E) and step 8 (F)

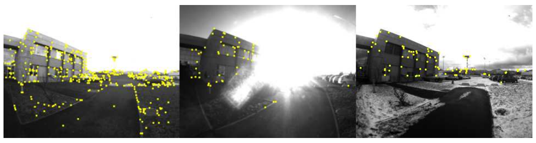

Figure 11: Localization robustness to weather changes

try it under different conditions. The robot was able to localize itself and to navigate autonomously in bright sunlight (even with the sun in the field of view of the camera) and with snow on the ground even if the reference sequence was recorded on a cloudy day without snow. Figure 11 shows the reference sequence on the left with all the interest points available in the database. Two images extracted from navigation experiments are shown on the right with the interest points correctly identified. The map building process is also robust to moving objects in the scene. We have been able to compute 3D reconstructions for sequences with up to 500 m long including pedestrians and moving vehicles (Royer et al., 2005).

### 3.4 Speed

The timings were made on a 3.4 GHz Pentium 4 processor with an image size of 640x480 pixels and 1500 interest points detected in each frame. The code uses the SSE2 instruction set for all the image processing. The reconstruction time for a sequence such as $outdoor_1$ is about 1 hour. The whole localization runs in 60 ms. Detecting interest points takes 35 ms, matching takes 15 ms and computing the pose takes 10 ms.

## 4 CONCLUSION

We have presented a localization algorithm and shown its performance under different conditions. It has been used both indoors and outdoors and with various cameras. The accuracy with reference to the learning trajectory is good enough for most robotic applications. Guidance applications based on this localization system have been successfully conducted outdoors with an accuracy similar to those obtained with a differential GPS sensor. The algorithm runs in real-time for the localization part. The approach proposed here works well for our intended application : that is driving a robot near the reference trajectory. For more complex navigation tasks either wide baseline matching techniques or a map with more keyframes from different viewing locations would be necessary. Future work will be more directed towards an improvement of robustness to changes in the environment. Even if the experiments presented in this paper have shown that the localization algorithm is robust to some changes, it may not be enough for an ever changing environment. For example in a city, cars parked along the side of the road change from day to day, trees evolve according to the season, some buildings are destroyed while others are built or modified. So our goal is to have a method to update the map automatically in order to take these modifications into account.

## REFERENCES

Araújo, H., Carceroni, R., and Brown, C., 1998. A fully projective formulation to improve the accuracy of Lowe's pose estimation algorithm. *Computer Vision and Image Understanding*, 70(2):pp. 227–238.

Cobzas, D., Zhang, H., and Jagersand, M., 2003. Image-based localization with depth-enhanced image map. In *International Conference on Robotics and Automation*.

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

Faugeras, O. and Herbert, M., 1986. The representation, recognition, and locating of 3-d objects. *International Journal of Robotic Research*, 5(3):pp. 27–52.

Fischler, O. and Bolles, R., 1981. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, 24:pp. 381–395.

Haralick, R., Lee, C., Ottenberg, K., and Nolle, M., 1994. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):pp. 331–356.

Harris, C. and Stephens, M., 1988. A combined corner and edge detector. In *Alvey Vision Conference*, pp. 147–151.

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

Kidono, K., Miura, J., and Shirai, Y., 2002. Autonomous visual navigation of a mobile robot using a human-guided experience. *Robotics and Autonomous Systems*, 40(2-3):pp. 124–1332.

Lavest, J. M., Viala, M., and Dhome, M., 1998. Do we need an accurate calibration pattern to achieve a reliable camera calibration ? In *European Conference on Computer Vision*, pp. 158–174.

Nistér, D., 2003. An efficient solution to the five-point relative pose problem. In *Conference on Computer Vision and Pattern Recognition*, pp. 147–151.

Nistér, D., Naroditsky, O., and Bergen, J., 2004. Visual odometry. In *Conference on Computer Vision and Pattern Recognition*, pp. 652–659.

Royer, E., Lhuillier, M., Dhome, M., and Chateau, T., 2005. Localization in urban environments : monocular vision compared to a differential GPS sensor. In *Conference on Computer Vision and Pattern Recognition*.