International Journal of Computer Vision 74(3), 237–260, 2007

DOI: 10.1007/s11263-006-0023-y

Monocular Vision for Mobile Robot Localization and Autonomous Navigation

ERIC ROYER,* MAXIME LHUILLIER, MICHEL DHOME AND JEAN-MARC LAVEST
LASMEA, UMR6602 CNRS and Université Blaise Pascal, 24 Avenue des Landais, 63177 Aubière, France
Eric.Royer@lasmea.univ-bpclermont.fr

Received October 14, 2005; Accepted December 11, 2006

First online version published in January, 2007

Abstract. This paper presents a new real-time localization system for a mobile robot. We show that autonomous navigation is possible in outdoor situation with the use of a single camera and natural landmarks. To do that, we use a three step approach. In a learning step, the robot is manually guided on a path and a video sequence is recorded with a front looking camera. Then a structure from motion algorithm is used to build a 3D map from this learning sequence. Finally in the navigation step, the robot uses this map to compute its localization in real-time and it follows the learning path or a slightly different path if desired. The vision algorithms used for map building and localization are first detailed. Then a large part of the paper is dedicated to the experimental evaluation of the accuracy and robustness of our algorithms based on experimental data collected during two years in various environments.

Keywords: localization, navigation, vision, mobile robot, structure from motion

1. Introduction

Localization is a key component in mobile robotics. The most popular localization sensor for outdoor robotic vehicles is the GPS receiver. A Real Time Kinematic (RTK) GPS allows a localization accurate to 1 cm. Such an accuracy is possible if enough satellites are visible from the receiver. Unfortunately, in dense urban areas, buildings can mask some satellites and in this case the accuracy of the localization drops considerably. For this reason, it is necessary to develop other localization sensors. The use of vision is very attractive to solve this problem because in places where the GPS is difficult to use such as city centers or even indoors, there are usually a lot of visual features. So a localization system based on vision could make a good complementary sensor to the GPS. Of course, for satisfactory reliability of the localization in a real world application, several sensors should be used (see for example, Georgiev and Allen, 2004). But each sensor itself should offer the best possible performance. Our purpose in this paper is to show what can be done with monocular vision only. We did not use odometry because it may not be available (for a hand held camera for example). We focused on monocular vision as opposed to stereo vision because it simplifies the hardware at the expense of a more complex software of course. We think, this is a good way to reduce the cost and size of the localization system.

In order to navigate autonomously, a robot needs to know where it must go. It also needs some knowledge about the environment so that it is able to compute its current localization. In our application, all these information are given to the robot in a simple way: the user drives the robot manually on a trajectory. After that the robot is able to follow the same trajectory in autonomous navigation. To do that we use only monocular vision and there is no artificial landmark. The robot is equipped with a wide angle camera in a front looking position. An overview of the vision system is presented on Fig. 1. In the learning step, the robot is manually driven and the camera records a learning video sequence. This sequence is processed off line to build a map of the environment with a structure from motion algorithm. Then the robot is able to localize itself in real-time in the neighborhood of the reference trajectory. It allows the robot to follow the same path as in the learning step or to follow a slightly different path which is useful to avoid an obstacle for example.

1.1. Related Work

Several approaches for robot localization using vision have been proposed. Simultaneous Localization And Mapping (SLAM) is attractive because localization is possible as soon as the system is started. But map building is something complex, so real time SLAM using only monocular vision is a difficult task. Some approaches

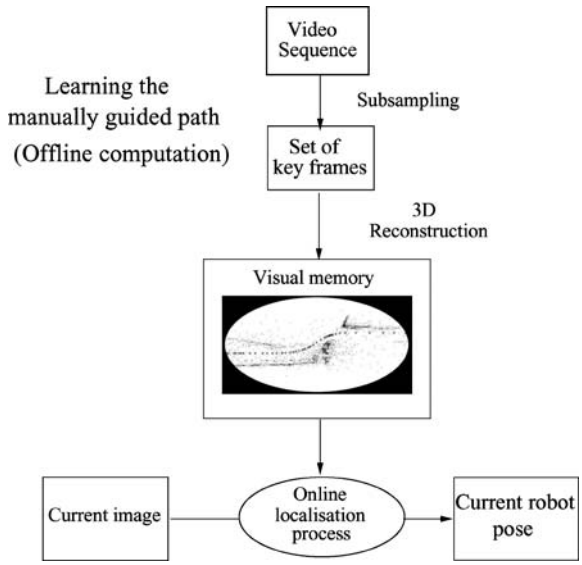

Figure 1. An overview of our vision system.

using more sensors such as stereo vision and odometry have been presented by Se et al. (2002). Real-time SLAM using only monocular vision has been achieved by Davison (2003), but only for a small environment with less than 100 landmarks. This would work well in a room for example, but it is not suitable if the robot needs to travel for hundreds of meters. It is also possible to compute ego motion by using only visual data as done by Nistér et al. (2004) or, in the specific case of urban environment (Simond and Rives, 2004). In this case, maintaining a large map is not required but the localization accuracy decreases as the distance traveled increases. Moreover, in two successive navigation experiments, the robot may not use the same landmarks and the resulting trajectory may be different.

Another approach for achieving robot navigation according to a human guided experience consists in representing the trajectory as a set of key images. Then the robot has to go from on key image to the next. With this approach, the robot will go through a number of well defined positions and the trajectory is repeatable from one navigation experiment to the next. The first possibility to do that is to go from one key frame to the next by visual servoing as done by Matsumoto et al. (1996), Remazeilles et al. (2004) and Blanc et al. (2005). In those algorithms, the current pose of the robot is not computed, so it is not possible to have the robot follow a path different from the one which was learnt. A variant of these approaches which uses an omnidirectional camera is described by Argyros et al. (). Another possibility was presented by Goedemé et al. (2005). In this algorithm, a relative localization of the robot with reference to the key frame is computed from features matched using a wide baseline technique. After that, a displacement vector is computed so that the robot can reach the next key frame.

The last approach consists in building a map of the environment first in an off line learning step. After that, the robot is able to localize itself with this map. The main advantage of these approaches is that they allow the robot to localize itself even if it is slightly off the path that was learnt. Since map building is done offline, computationally intensive algorithms can be used, resulting in an accurate and rich map. This is not always the case in SLAM approaches because map building and localization must be done simultaneously in real-time. The map can be built by using different techniques and different sensors. Vacchetti et al. (2003) use a CAD model given by the user and develop a very efficient localization system. Cobzas et al. (2003) use a rotating camera along with a laser range finder to build a set of panoramic images enhanced with 3D information. With this map, a single 2D image is enough to localize the camera. Kidono et al. (2002) build a 3D reconstruction from a stereo camera and an odometer under the assumption that the ground is planar. Then they use the map generated in the 3D reconstruction process to localize the robot in real-time. Our system works on the same principle, but we don't make the assumption that the ground is planar and we use only one calibrated camera.

1.2. Paper Structure and Contribution

This article is a synthesis of previously published conference papers (Royer et al., 2005a,b,c) with new theoretical developments and experimental data. The main added contribution of this paper is the real-time computation of the localization uncertainty and its experimental validation. Navigation on a closed loop sequence or on a path different from the learning path are also discussed.

Section 2 describes the off-line map building process and details the structure from motion algorithm. The method used for matching images, which is nearly the same for the reconstruction and the localization, is also detailed here. Section 3 describes how the localization of the robot is computed from a video frame. The computation of localization uncertainty is presented in Section 4. The experimental vehicle and the control law are presented in Section 5. Then, in Section 6 we evaluate the performance of the algorithms with experimental data obtained with a robotic vehicle equipped with the localization system presented in this paper.

2. Map Building

2.1. Overview

The goal of the map building process is to obtain the pose 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. This must be done from a monocular image sequence. 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 pattern (Lavest et al., 1998). Radial distortion is modelled with a fifth order polynomial. Camera calibration is important especially when using wide angle lenses with a strong radial distortion. We have used two lenses: a wide angle lens with 60° field of view and a fish-eye lenses with 130° field of view. Both lenses perform equally well for reconstruction, but a fish-eye is preferable for localization because it reduces the chances of occultations. With a calibrated camera, the structure from motion algorithm is more robust and the accuracy of the reconstruction is increased. This is important for us because in our video sequences with a front looking camera, the triangulation of the points must be done with very small angles. In spite of this difficulty, we chose a forward looking camera because there is always some free space in front of the vehicle, which means the field of views contain both near and far objects. If the camera was looking to the side, and the vehicle was moving along a wall very close to the camera, features would move very fast in the image which would make image matching very difficult (motion blur could also be a problem).

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

2.2. *Image Matching*

Every step in the reconstruction as well as the localization relies on image matching. In order to match two images, interest points are detected in each image with Harris corner detector (Harris and Stephens, 1988). Corner response R is computed as in Harris work and local maxima of R are potential interest points. In order to have interest points in each part of the image, we have divided the image area in 64 buckets. In each frame (512×384 pixels), we keep the 20 best local maxima in each bucket plus the 500 best maxima on the whole image. So we get at most 1780 interest points in each frame. For an interest point M^1 at coordinates (x, y) in image 1, we define a search region in image 2. The search region is a rectangle whose center has coordinates (x, y). For each interest point M_i^2 inside the search region in image 2, we compute a correlation score between the neighborhoods of M^1 and M_i^2. We use a Zero Normalized Cross Correlation over a 11×11 pixels window. All the pairs with a correlation score above 0.8 are sorted. The best one is kept as a good match and the unicity constraint is used to reject matches which have become impossible. Then the second best score is kept and so on until all the potential matches have been kept or discarded. This image matching scheme is used for the localization part and as the first step in the reconstruction part. In the reconstruction process, a second matching is done after the epipolar constraint has been computed. In the second matching step, the search region is a narrow strip around the epipolar line and the correlation score threshold is lowered to 0.8^2.

2.3. *Key Frame Selection*

Before computing camera motion, it is necessary to extract key frames from the reference video sequence. 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 find enough point correspondences between the key frames. Some algorithms have been proposed to do that. For example, Nistér (2001) considers the whole video sequence then drops redundant frames. Another method is possible based on the Geometric Robust Information Criterion (GRIC) proposed by Torr et al. (1999). Our method is much simpler and less general than these ones but works well for our purpose.

The first image of the sequence is always selected as the first key frame I_1. The second key frame I_2 is chosen so that there are as many video frames between I_1 and I_2 while there are 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} so that:

- there are as many video frames as possible between I_n and I_{n+1},
- there are at least M interest points in common between I_{n+1} and I_n,
- there are at least N common points between I_{n+1} and I_{n-1}.

In our experiments we choose $M = 400$ and $N = 300$. Figure 2 shows consecutive key frames extracted from the city sequence.

2.4. *Initial Camera Motion Computation*

We compute an initial solution for camera motion and a hierarchical bundle adjustment is used to refine this initial estimation.

Figure 2. Consecutive key frames extracted from the city sequence.

For the first image triplet, the computation of the camera motion is done with the method described in 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. This gives at most 40 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. With a calibrated camera, three 3D points whose projections in the image are known are enough to compute the pose of the second camera. Several methods are compared in Haralick et al. (1994). We chose Grunert's method with RANSAC.

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 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 had just been computed. This second matching step allows to increase the number of correctly reconstructed 3D points by about 20%. This is important for us for two reasons. Those 3D points are used in the computation of the next camera, and they are also the landmarks used in the localization process. So we need to get as many as possible which wouldn't be the case if the goal was just to recover the motion of the camera.

2.5. *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 greatly reduce this problem, we use a bundle adjustment. The bundle adjustment is a Levenberg-Marquardt minimization of the cost function $f(C_E^1, \ldots, C_E^N, X^1, \ldots, X^M)$ 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:

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

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 3×4 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 using the bundle adjustment only once on the whole sequence could cause problems. this is 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 a hierarchical bundle adjustment as in 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 Section 2.4. After each triplet has been computed we run a bundle adjustment over its three frames.

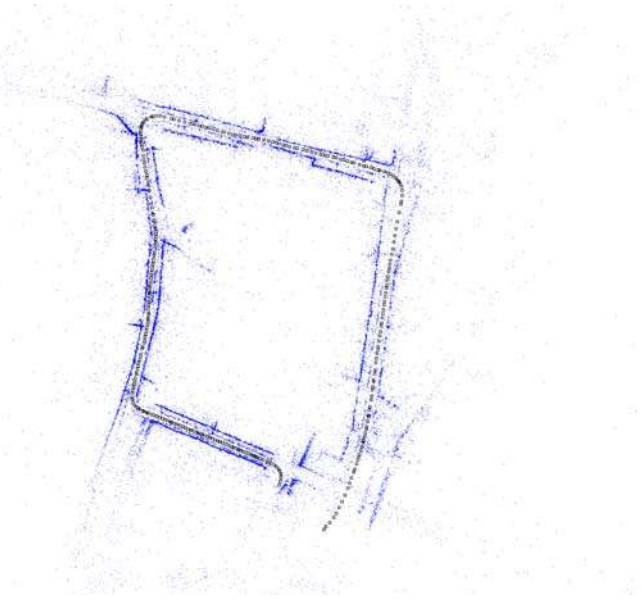

Figure 3. Top view of the 3D reconstruction computed from the city sequence. The squares are the camera position for each key frame. The dots are the reconstructed interest points projected on an horizontal plane.

In order to merge two sequences S^1 and S^2, we use the last 2 cameras S^1_{N-1} and S^1_N of S^1 and the first 2 cameras S^2_1 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^1_N and S^2_2 have the same position and orientation. Then the scale factor is computed so that $d(S^1_{N-1}, S^1_N) = d(S^2_1, S^2_2)$, where $d(S^i_n, S^j_m)$ is the euclidian distance between the optical centers of the cameras associated with S^i_n and S^j_m. This doesn't ensure that S^1_{N-1} and S^2_1 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. The number of points used in the bundle adjustment is on the order of several thousands. For example, Fig. 3 shows the result of the 3D reconstruction computed for the city sequence. There are 308 key frames and 30584 points. The path is about 600 meters long.

3. Real Time Localization

3.1. Camera 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 localization process can be divided in four steps. First, a set of the landmarks which should be visible is selected by finding the closest key frame. Then an approximate 2D position of each landmark in the current frame is computed based on the pose of the camera of the previous frame. With this information the landmarks are matched to the interest points detected in the current frame. And finally the pose is computed with these matches. Figure 4 shows interest points matched between the current video frame and the closest key frame in the reference sequence recorded with different weather conditions.

The current image is denoted 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 we can project it with P_0 so we know approximately its 2D position in the current frame I. This is illustrated on Fig. 5. 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 δ 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

Figure 4. Matching interest points between the reference key frame learned with snow (left) and the current video frame without snow (right).

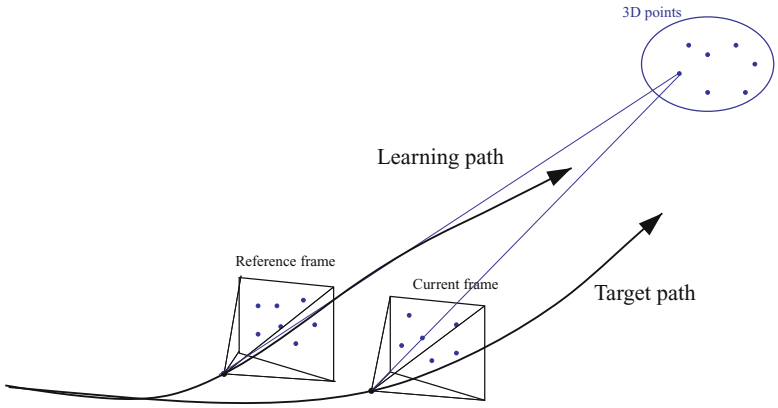

Figure 5. Computing an expected position for the landmarks in the current frame.

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.

It would be relatively easy to introduce a motion model of the vehicle or odometry measurements in this framework. For each new frame, we consider that the pose of the camera is the same as the previous pose in order to match features. Incorporating odometry would allow to have a better initial estimate of the camera pose. We have not done that in order to keep the possibility to use the localization algorithm with hand held cameras.

At this point we have a real-time algorithm that is able to compute the pose of the camera. We always keep track of the camera pose with 6 degrees of freedom, even if the control law works in a ground plane. It allows us to track interest points even if the ground is irregular.

3.2. Robot Localization

Since the localization algorithm is based only on vision the coordinate system has its center at the optical center of the first camera in the reference sequence, with one axis along the optical axis. Moreover, there is no scale information. In order to control the robot, we need to provide a position for the robot in a metric coordinate system. We achieve that by entering manually the length of the path to set the scale factor. The position of the camera on the robot has been measured so we can enter directly the rigid transformation between the camera and the robot. Once the camera pose has been computed, we compute the robot position and orientation with 3 degrees of freedom only because the robot can be controlled only along three dimensions (translation on the ground plane and rotation around the vertical axis). To do that, we assume that the ground is locally planar and horizontal at the current position of the robot.

4. Computing the Localization Uncertainty

Knowing the uncertainty of the robot localization computed for the current frame has some important applications. It allows data fusion with other localization sensors (such as an odometer). It could also be used for temporal

filtering of the localization result if vision is the only sensor.

Uncertainty in the localization can come from several sources. The main one comes from the uncertainty in the position of interest points. Interest points can be detected with an accuracy on the order of one pixel. In the structure from motion algorithm, this causes uncertainty in the 3D positions of the cameras and the points. When computing the localization of the camera, the result is affected by the uncertainty of the landmarks positions in 3D space and by the uncertainty on the positions of the interest points detected in the current frame. This is the only source of uncertainty we considered in our computation. So the first step is to compute the uncertainty in the 3D reconstruction which is detailed in Section 4.2, and we use that to compute the uncertainty for the localization result in Section 4.1. In both cases, we have a minimization problem and the covariance matrix is obtained from the inverse or the pseudo-inverse of the hessian (Hartley and Zisserman, 2000). Propagating the uncertainties throughout the reconstruction and the localization process is possible but very time consuming. Since we want to be able to compute the localization uncertainty in real-time, we had to make some trade-offs between speed and accuracy. Therefore, some sources of uncertainty were not taken into account in our computation. These include the inaccuracy of the internal parameters of the camera for the vision part, and for the robot localization part, uncertainty in the measurement of the transformation between the camera coordinate system and the robot coordinate system (we assumed a rigid transformation here, but in reality it is not rigid because of the suspension system).

4.1. *Uncertainty in the Localization*

Let's assume that the uncertainties of the 3D points have been computed (the method used to compute them will be detailed in Section 4.2). We want to compute the covariance matrix Cov_{cam} associated to the pose of the camera computed from n 3D/2D correspondences. The projection of point X_j is detected in the current frame at the 2D position m_j. The reprojection error for point X_j is $e_j = \pi(CX_j) - m_j$ where $\pi(CX_j)$ denotes the projection of point X_j with the camera parameters C computed for the current frame. We assume that e_j follows the normal distribution $\mathcal{N}(0, \Lambda_j)$. The vector made with the 3D points visible in the current frame is $X = (\dots, X_j, \dots)^T$. This vector follows a normal distribution with mean $X^0 = (\dots, X_j^0, \dots)^T$ given by the bundle adjustment and covariance matrix Cov_{3d} (which will be computed in Section 4.2). Computing the maximum likelihood estimation of the camera pose means finding the best values for $(C, X_1, \dots, X_n)$ so that $\|G(C, X_1, \dots, X_n)\|^2$ is minimum, with:

$$\|G(C, X_1, \dots, X_n)\|^2 = \sum_{j=1}^{n} e_j^T \Lambda_j^{-1} e_j + \begin{pmatrix} \dots \\ X_j - X_j^0 \\ \dots \end{pmatrix}^T Cov_{3d}^{-1} \begin{pmatrix} \dots \\ X_j - X_j^0 \\ \dots \end{pmatrix} \quad (1)$$

In this expression, we assume that the $n+1$ random vectors $e_1, \dots, e_n, X$ are independent. Obviously, we have:

$$G(C, X_1, \dots, X_n) = \left(e_1^T \Lambda_1^{-\frac{1}{2}}, \dots, e_n^T \Lambda_n^{-\frac{1}{2}}, (X - X^0)^T Cov_{3d}^{-\frac{1}{2}}\right)^T \quad (2)$$

Computing $\Lambda_j^{-\frac{1}{2}}$ and $Cov_{3d}^{-\frac{1}{2}}$ is possible because Λ_j and Cov_{3d} are real symetric definite positive matrices. The right member of Eq. (2) is a random variable with the normal distribution $\mathcal{N}(0, I_{5n})$. Once the minimum is found, The result on the backward tranport of covariance given in Hartley and Zisserman (2000) gives us the covariance matrix $(J_G^T J_G)^{-1}$ with J_G the Jacobian matrix of G. However, even by exploiting the sparseness of the matrices, doing a full minimization with these equations would be too slow for the real-time localization. So, as we have seen in Section 3, the pose is computed by simply minimizing $\tilde{G}(C)^2 = \sum_{j=1}^{n} e_j^T \Lambda_j^{-1} e_j$. Several iterations can be done in a few milliseconds when using this approximation. The more complex formulation of Eq. (1) is used only for the covariance matrix computation because it needs to be done only once per frame.

Computing J_G gives a sparse matrix :

$$J_G = \left(\begin{array}{c|cccc} A_1 & B_1 & 0 & \dots & 0 \\ A_2 & 0 & B_2 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ A_n & 0 & 0 & 0 & B_n \\ \hline 0 & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & C_{i,j} & \dots \\ \vdots & \dots & \dots & \dots & \dots \end{array}\right) \quad (3)$$

with $A_j = \Lambda_j^{-\frac{1}{2}} \frac{\partial e_j}{\partial C}$ a 2×6 matrix, $B_j = \frac{\partial e_j}{\partial X_j}$ a 2×3 matrix, $C_{i,j}$ a 3×3 matrix. $Cov_{3d}^{-\frac{1}{2}}$ is a $3n \times 3n$ matrix defined by blocks $C_{i,j}$.

The covariance matrix Cov_{cam} associated to the current camera pose is the 6×6 upper left block of $(J_G^T J_G)^{-1}$. Computing $J_G^T J_G$ gives:

$$J_G^T J_G = \begin{pmatrix} U & W \\ W^T & V \end{pmatrix} \quad (4)$$

with U a 6×6 block, W a $6 \times 3n$ block and V a $3n \times 3n$ matrix:

$$U = A^T A = A_1^T A_1 + A_2^T A_2 + \cdots + A_n^T A_n \quad (5)$$

$$W = \begin{pmatrix} A_1^T B_1 & A_2^T B_2 & \dots & A_n^T B_n \end{pmatrix} \quad (6)$$

$$V = \begin{pmatrix} B_1^T B_1 & 0 & 0 & 0 \\ 0 & B_2^T B_2 & 0 & 0 \\ & & & \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & B_n^T B_n \end{pmatrix} + Cov_{3d}^{-1} \quad (7)$$

Finally, the covariance matrix associated with the camera pose is:

$$Cov_{cam} = (U - WV^{-1}W^T)^{-1} \quad (8)$$

In the general case, V is not a sparse matrix and the computation of $WV^{-1}W^T$ can take a lot of time. In order to be able to compute the localization uncertainty in real-time, an additional hypothesis should be done. If we assume that the positions of the 3D points are independent random variables, then Cov_{3d} becomes a block-diagonal matrix and matrix V can be rewritten :

$$V = \begin{pmatrix} B_1^T B_1 + C_{1,1}^T C_{1,1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & B_n^T B_n + C_{n,n}^T C_{n,n} \end{pmatrix} \quad (9)$$

With this assumption Cov_{cam} can be computed very quickly. Additionally, the computation of the covariance matrix of the 3D points can also be simplified because we need to compute only the blocks on the diagonal. Section 4.2 details the computation of the covariance matrix of the 3D points, then Section 4.3 compares the various methods that can be used.

4.2. Uncertainty in the 3D Reconstruction

Note that the reconstruction of the reference sequence is obtained by minimizing the sum of squared reprojection errors using bundle adjustment. We obtain a maximum likelihood estimation (MLE) of 3D points and extrinsic camera parameters assuming that the reprojection errors obeys independent and identical zero mean Gaussian distributions. This perturbation of errors propagates to a Gaussian perturbation of the estimated parameters, such that the covariance matrix may be estimated (Hartley and Zisserman, 2000). Thus, for each key-frame of the reference sequence, we estimate the covariance matrix of the 3D points seen by the key-frame (named Cov_{3d} in the previous paragraph) as a $3n \times 3n$ diagonal block of the full covariance matrix relating all the points and all the cameras.

We use the general method of gauge constraints to fix the reconstruction coordinate system and choose our covariance matrix (Triggs et al., 2000). Among other choices, the symmetric constraint on the reference camera location (e.g. Lhuillier et al., 2006) is chosen since it usually spreads evenly and minimizes the uncertainties on all reference cameras. Many details about a practical computation of points and cameras covariances for complex reconstructions like ours are given in Lhuillier et al. (2006).

A simplification of this method can also be used if we make the additional assumption that there is no uncertainty for the cameras in the 3D reconstruction. In this case, the covariance matrix for each point can be computed independently. We consider a point X_j seen by n cameras C_i, $i \in \{1, \dots, n\}$. We want to compute its covariance matrix $C_{j,j}$. We assume that the n reprojection errors of this point denoted e_i in image i follow independent and normal distributions $\mathcal{N}(0, \Lambda_i)$. Furthermore, we assume that $\Lambda_i = \sigma^2 I_2$, with σ^2 an unknown parameter. Thus, the probability density function of the statistical model is:

$$f(X_j, \sigma^2) = \frac{1}{(2\pi\sigma^2)^n} e^{-\frac{\sum_i \|e_i\|^2}{2\sigma^2}} \quad (10)$$

The MLE of X_j and σ^2 are $\hat{X}_j$ minimizing $X_j \mapsto \sum_i \|e_i\|^2$ and $\hat{\sigma}^2 = \frac{\sum_i \|e_i\|^2}{2n}$. We replace the MLE of σ^2 by the unbiased estimator $\hat{\sigma}^2 = \frac{\sum_i \|e_i\|^2}{2n-3}$, with $2n$ the number of observations, and 3 the number of degrees of freedom of X_j. This gives us $\Lambda_i = \hat{\sigma}^2 I_2$. Finally, we obtain the estimation of $C_{j,j}$ by the inverse of $J_F^T J_F$ where J_F is the jacobian matrix of the vector error F with $F = (e_1^T \Lambda_1^{-\frac{1}{2}}, \dots, e_n^T \Lambda_n^{-\frac{1}{2}})^T$.

Figure 6 shows the confidence ellipsoids computed with this method for the points visible in one key frame of the loop sequence (see Fig. 20). Because of the forward motion of the camera, most ellipsoids have a major axis much longer than the minor axis.

4.3. Comparison

In order to choose between all the possible trade-offs we made a comparison of four methods based on different assumptions:

- In method 1, Cov_{cam} is computed with the full matrix Cov_{3d}.
- In method 2, Cov_{3d} is computed with the general method, but only the diagonal blocks are used in the computation of Cov_{cam}.

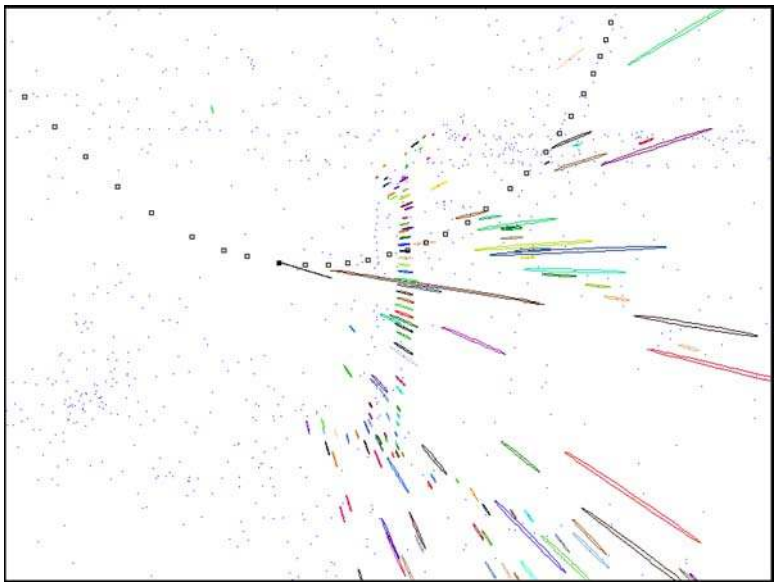

Figure 6. 99% confidence ellipsoids computed for the points visible in one frame (view from the top).

- In method 3, Cov_{3d} is computed with the assumption that the camera have no uncertainty in the reconstruction process.
- In method 4, we simply ignore the uncertainty of the 3D positions of the points and the only source of uncertainty comes from the positions of the 2D points in the current frame.

Method 1 takes too much time to be used in a real-time system, but we included it in this comparison to know if using the non diagonal blocks of Cov_{3d} was mandatory. We computed the length of the major semi-axis of the 90% confidence ellipsoid associated to the camera positions for all four methods. The result is shown on Fig. 7 for a few frames of the video used in Section 6.6.

As we could expect, taking into account the uncertainty of the 3D points increases the size of the ellipsoids. Taking into account the uncertainty of the cameras in the reconstruction has the same effect. However, the difference between method 1 and method 2 is small, and since method 2 can be used in real-time, it seems that it is a good method for computing localization uncertainty. Computation time is the same for methods 2 and 3, the difference lies in the complexity of the implementation for computing the diagonal elements of Cov_{3d}. With the simplification made in method 3, the size of the confidence ellipsoid is slightly underestimated, but the result is still much closer to the one obtained with the full computation than with method 4. A comparison between the size of the confidence ellipsoid and the localization error measured with the RTK GPS is also done in Section 6.6.3.

5. Experimental Vehicle and Control Law

This part is presented only for the completeness of the paper. It refers to a work made by another team of the laboratory (see Thuilot et al. (2004) for more details).

5.1. Vehicle Modeling

The experimental vehicle is called the Cycab. With its small dimensions (length: 1.90 m, width: 1.20 m), it can transport simultaneously two passengers. This vehicle is entirely under computer control. For the experimentations detailed in this paper, only the front wheels are steered, so the mobility is the same as a common car.

A classical kinematic model, the tricycle model, where the two actual front wheels are merged as a unique virtual wheel is used (see Fig. 8). The vehicle configuration can be described without ambiguity by a 3 dimensional state vector composed of s, curvilinear coordinate along the reference path of the projection of vehicle rear axle center, and of y and $\tilde{\theta}$, vehicle lateral and angular deviations with respect to this path. On the other hand, the control vector is defined by the vehicle linear velocity and the front wheel steering angle, denoted respectively v and δ. Vehicle model is then given (see e.g. de Wit (1996)) by:

$$\begin{cases} \dot{s} = v\dfrac{\cos\tilde{\theta}}{1 - yc(s)} \\ \dot{y} = v\sin\tilde{\theta} \\ \dot{\tilde{\theta}} = v\left(\dfrac{\tan\delta}{l} - \dfrac{c(s)\ \cos\tilde{\theta}}{1 - y\,c(s)}\right) \end{cases} \tag{11}$$

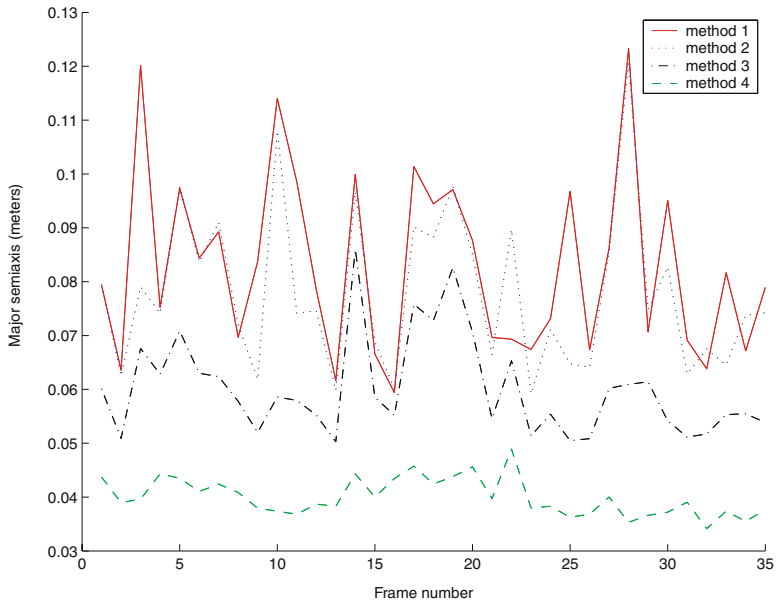

Figure 7. Major semiaxis (in meters) of the 90% confidence ellipsoid with four computation methods for a few frames localized with reference to a given key-frame. Frame to frame variations are mostly due to changes of the 3D-2D correspondences.

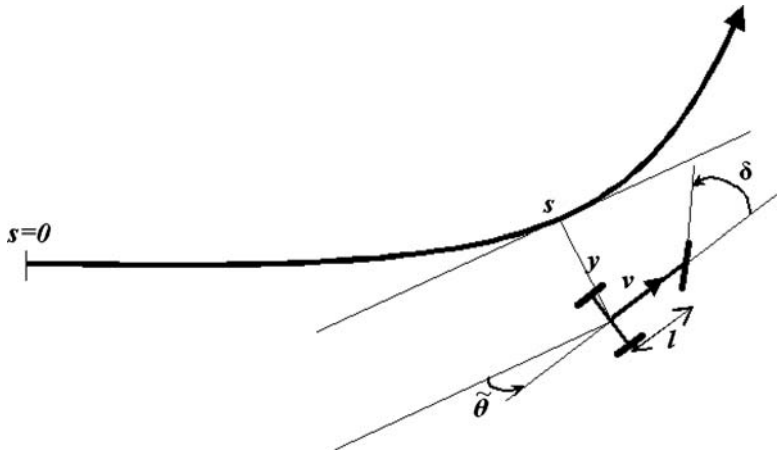

Figure 8. Tricycle model description.

where l is the vehicle wheelbase and $c(s)$ denotes the reference path curvature at coordinate s. It is assumed that: $y \neq \frac{1}{c(s)}$ (i.e. the vehicle is not on the reference path curvature center) and $\tilde{\theta} \neq \frac{\pi}{2}[\pi]$. In practical situations, if the vehicle is well initialized, such difficulties never arise.

5.2. *Control Law*

The control objective is to bring and maintain state variables y and $\tilde{\theta}$ to 0, relying uniquely on control variable δ (v is considered as a possibly varying free parameter). The whole vehicle state vector $(s, y, \tilde{\theta})$ is available with a satisfactory accuracy by comparing vehicle absolute position and heading, provided by the vision algorithm, with the reference path. Via invertible state and control transformations, the nonlinear vehicle model (11) can be converted, in an exact way, into the so-called chained form (see Samson, 1995). $(a_1, a_2, a_3) = (s, y, (1 - yc(s)) \tan \tilde{\theta})$ is the chained state vector and $M = (m_1, m_2)^T = \Upsilon(v, \delta)^T$ is the chained control vector. From this, a large part of linear systems theory can be used (but, since the transformations are exact, it is not required that the vehicle state is in a specific configuration, contrarily to tangent linearization techniques). More precisely, it can be noticed that path following (i.e. control of a_2 and a_3) can be achieved by designing only m_2 as a linear proportional derivative controller. The expression

of the actual control variable δ can then be obtained by inverting the chained control transformation. Computations, detailed in Thuilot et al. (2004), lead to:

$$\delta(y,\tilde{\theta}) = \arctan\left(l\left[\frac{\cos^3\tilde{\theta}}{(1-c(s)\,y)^2}\left(\frac{d\,c(s)}{d\,s}\,y\,\tan\tilde{\theta} - K_d\,(1-c(s)\,y)\,\tan\tilde{\theta} - K_p\,y + c(s)\,(1-c(s)\,y)\,\tan^2\tilde{\theta}\right) + \frac{c(s)\,\cos\tilde{\theta}}{1-c(s)\,y}\right]\right) \tag{12}$$

with $K_p, K_d > 0$ the proportional derivative gains.

6. Experimental Results

6.1. Methods for Performance Evaluation

6.1.1. Using GPS Data as the Ground Truth. Most of the results presented in this paper, show the accuracy of the system by comparing the results of the algorithms with the data recorded by a Real Time Kinematic (RTK) GPS mounted on the Cycab. But comparing data obtained by two different sensors is not completely straightforward. This paragraph explains how results from the vision algorithms are compared to GPS measurements.

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 receiver is located above the mid-point between the rear wheels of the car, while the camera is in front of the vehicle (1.15 meters in front of the GPS sensor). 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 mounted on the Cycab 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 global rotation, translation and scale change. The approach described by Faugeras and Herbert (1986) is used to compute this global transformation. After this transformation has been made, the camera and GPS positions are available in the same metric coordinate system. This process is done for the 3D reconstruction of the reference video sequence. After that, the localization algorithm gives directly the camera positions in the same metric coordinate system.

The GPS sensor we use is a Real Time Kinematics Differential GPS (Thalès Sagitta model). It is accurate to 1 cm (standard deviation) in an horizontal plane when enough satellites are available. The accuracy on a vertical axis is only 20 cm on our hardware platform. So we discard the vertical readings and all the localization errors reported in this article are measured in an horizontal plane only. In any case, vertical errors could be interesting for a flying robot but not for our application.

6.1.2. Reconstruction and Localization Error. We want to distinguish between the error that is attributed to the reconstruction process and the error coming from the localization algorithm. So we define two errors to measure the reconstruction and the localization accuracy. The reconstruction error is the average distance between the camera positions computed in the structure from motion algorithm and the true positions given by the RTK GPS (after the two trajectories have been expressed in the same coordinate system). 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.

The definition of the localization error is a bit more complex. To understand why, suppose the robot is exactly on the reference path and the localization algorithm indicates that the robot is on the reference path. In this case, the localization error should be zero, and controlling the robot so that this error is kept zero would result in the robot following the reference path. But if we computed directly the difference between the 3D position given by the localization algorithm and the position given by the GPS, we would get the reconstruction error which is not zero in most cases. In fact, in our application, 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. First we compute the lateral deviation between the current robot position G_1 and the closest robot position G_0 on the reference trajectory. This is illustrated on Fig. 9. 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 RTK GPS. When working with vision it must be computed from the camera position and orientation. The 3D reconstruction used as the reference has already been transformed so that it is in the same metric coordinate system as the GPS 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 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 $y_v = \overrightarrow{G_0G_1} \cdot \overrightarrow{N}$. The lateral deviation is computed from the GPS measurements as well and we get y_g (in this case we have directly G_0 and G_1). y_g and y_v are the same physical

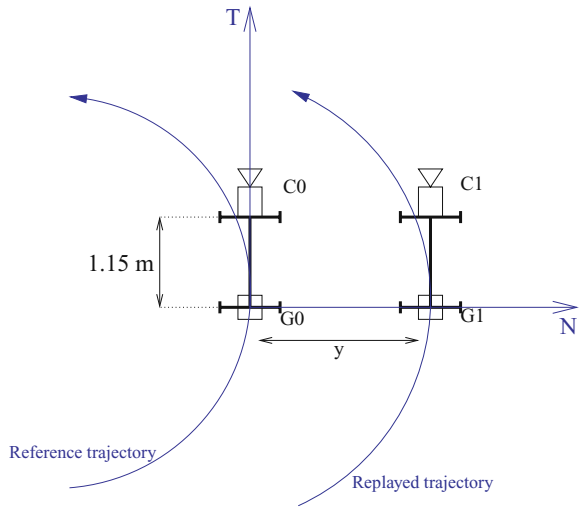

Figure 9. Computing the lateral deviation from the reference trajectory.

distance measured with two different sensors. Then the localization error is defined as $\epsilon = y_v - y_g$.

6.2. 3D Reconstruction Results

The problem we want to solve is robot localization. Our goal is not to build the best reconstruction of the environment. In fact, we are convinced that an accurate reconstruction is not absolutely necessary for robot navigation, especially if the robot stays on the learning path. The map should be accurate on a small scale (less than about 50 meters) but long term drift is acceptable. We will discuss this point in more detail when we examine the case of closed trajectories. Nevertheless, it is interesting to see how accurate our reconstruction is, and to make sure it is accurate on a small scale. The results presented in this section concern only the reconstruction of the camera motion. The accuracy of the structure was not measured.

Four sequences called $outdoor_1$ through $outdoor_4$ were recorded by driving manually the vehicle on a 80 m long trajectory. The four sequences were made approximately on the same trajectory (with at most a 1 m lateral deviation), the same day. The lens used was a standard wide angle lens with a field of view of roughly 60°. We computed a 3D reconstruction from each of the four sequences. 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. A few images extracted from $outdoor_1$ are shown on Fig. 10. The positions of the key frames computed from this sequence are shown on Fig. 11 (as seen from the top) along with the trajectory recorded by the GPS. The reconstruction error for each of the sequences was 25, 40, 34 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.

6.3. Localization Accuracy

6.3.1. Positional Accuracy. The localization accuracy was computed from the same sequences $outdoor_1$ through $outdoor_4$ we used in Section 6.2. Each sequence was used in turn as the reference sequence. We computed a localization for $outdoor_i$ using $outdoor_j$ as the reference sequence for each $j \in \{1, 2, 3, 4\}$ and $i \neq j$. So we made twelve experiments.

The localization error was computed with the method explained in Section 6.1.2. From this we can compute the standard deviation of ϵ 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. Fig. 13 shows the lateral deviation and localization error for one experiment with a 1.9 cm average localization error. The error measured in this experiment is very close to the GPS accuracy (1 cm of standard deviation in the best conditions). So the noise in the GPS measurements contributes to the localization error given in this paragraph.

Figure 10. A few images from $outdoor_1$.

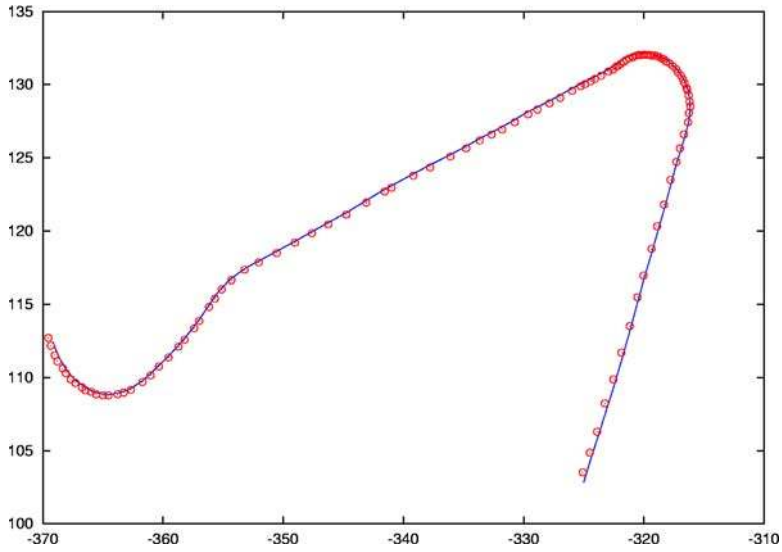

Figure 11. Position of the key frames (circles) with reference to the trajectory recorded by the RTK GPS (continuous line). Units in meters.

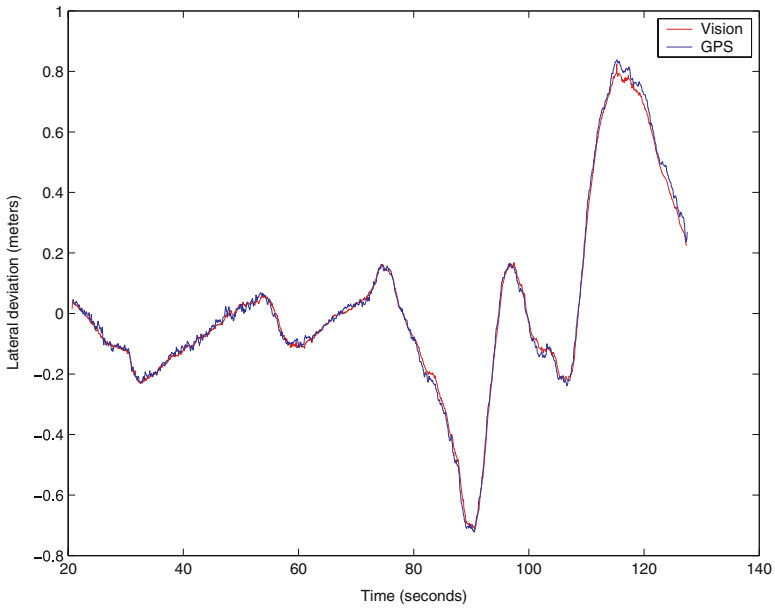

Figure 12. Lateral deviation measured with the RTK GPS y_g (blue) or with vision y_v (red). The two curves are nearly the same.

6.3.2. Rotational Accuracy. In order to evaluate the rotational accuracy, we made an indoor experiment because the GPS can't give accurate angular measurements. The camera was mounted on a rotating platform, with the optical center on the rotational axis. The angle of the platform can be read with about $\pm 0.1^\circ$ accuracy. We compared the orientation α provided by the localization part of the vision algorithm to the angle α_0 given by the platform. For this experiment (and the following ones till the end of the article), we used a fish eye lens providing a 130° field of view (in the diagonal) and we made a measurement for each angle from $\alpha_0 = -94^\circ$ to $\alpha_0 = 94^\circ$ 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 Fig. 14. The algorithm was not able to provide a reliable 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 $\pm 0.1^\circ$, 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

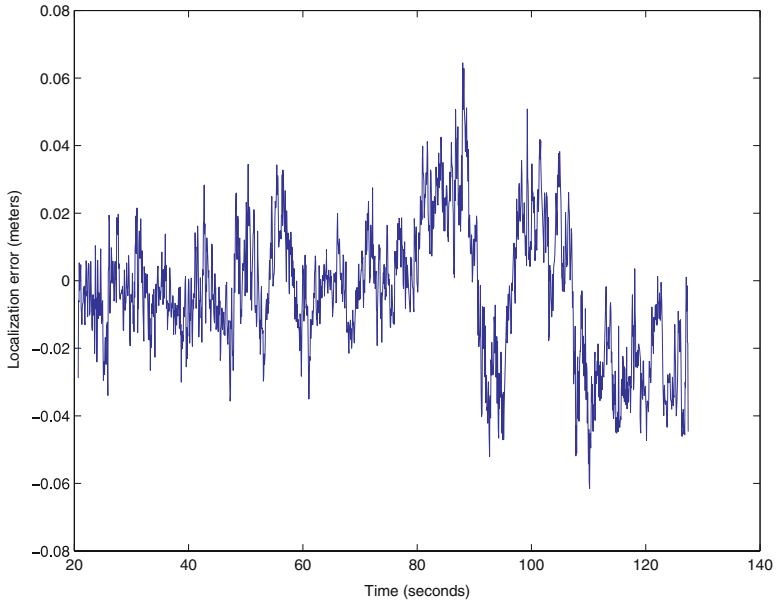

Figure 13. Localization error ϵ.

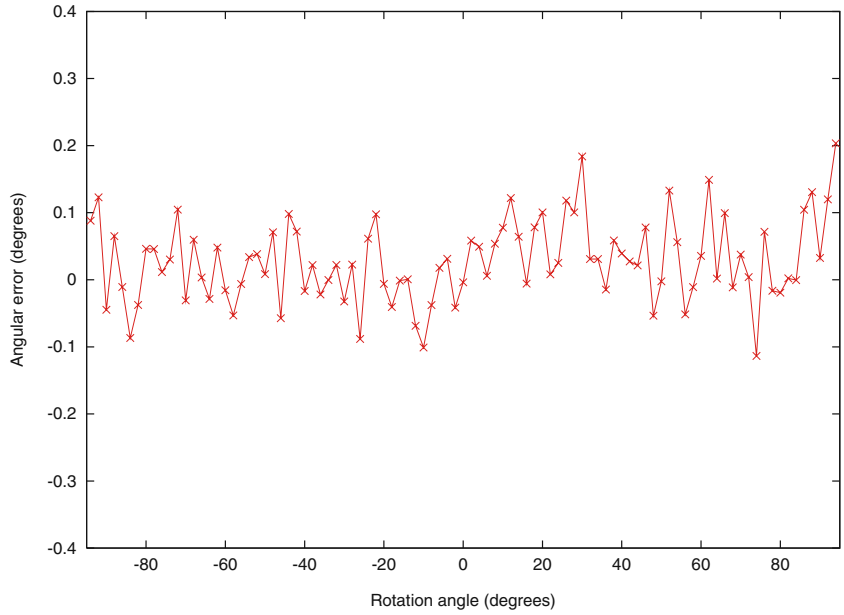

Figure 14. Angular error.

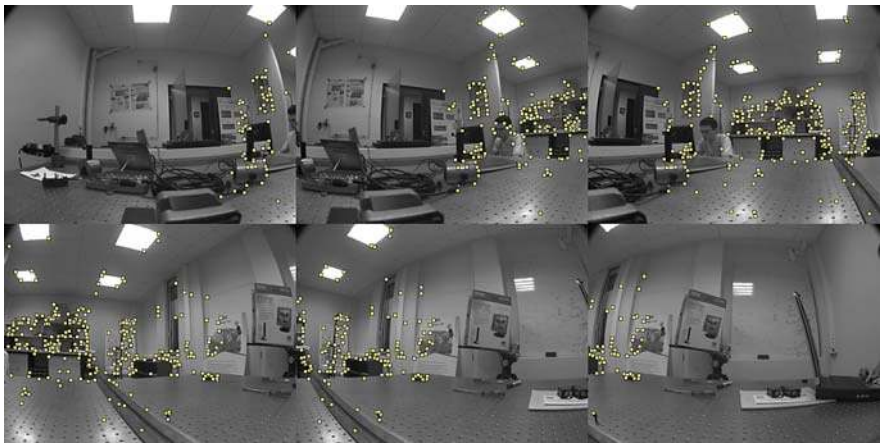

Figure 15. Images taken at -90°, -60°, -30°, 30°, 60° and 90° orientation, with interest points correctly matched.

can be used is very small, and the localization becomes impossible if there is an occultation in this area. Images captured at different orientations are shown on Fig. 15.

6.4. Autonomous Navigation

We also made some experiments where we tested the whole system. A reference video sequence was recorded and a 3D reconstruction was computed from it. Then the robot had to follow the same path in autonomous navigation. We recorded GPS measurements at the same time we recorded the learning video sequence. Then we also recorded GPS measurements in autonomous navigation.

The learning path was 127 meters long. It was chosen so that there are both straight lines and tight turns, and because the buildings are sometimes far (less visual features) and sometimes closer. Vehicle speed was chosen constant and equal to 2 km/h. The result of the structure from motion algorithm is displayed on Fig. 16 as seen from the top. There were 182 key frames and 16295 points correctly reconstructed.

The reference video sequence was recorded on a cloudy day. The first two navigation experiments were made a few days later with a cloudy weather too. But the second set of two was made on a clear day with the sun low in the sky and sometimes in the field of view of the camera. A few images from the video recorded during the last navigation experiment as well as the corresponding key frame are displayed on Fig. 17. The last image outlines the necessity of having a wide field of view and local visual features all over the frame. It shows the interest points which are used in the localization. The center of the image is completely overexposed because the sun is in front of the camera, but the building on the left can still be used for computing the camera pose.

For comparison purposes, a fifth experiment has also been performed, relying on the RTK GPS sensor (instead of the vision algorithms) to provide the vehicle state vector which is used in the control law. Lateral deviations from the reference path recorded during 3 of these experiments are displayed with the same scale on Fig. 18. Letters enable to identify each part of the trajectory, with respect to the letters shown on Fig. 16.

It can be observed that the vision algorithms detailed in this paper appear as a very attractive alternative to RTK GPS sensor, since they can provide with roughly the same guidance accuracy. It can also be noticed that these vision algorithms are reliable with respect to outdoor applications since they appear robust to weather conditions: guidance accuracy is not significantly altered in as harsh conditions as the sunny ones. More precisely, guidance performances during straight lines and curves following are investigated separately on Tables 1 and 2. Table 1 reports the mean value of the lateral deviation ($|y|$) during straight lines part of the trajectory, denoted B and D. In the most favorable situation (cloudy weather), vision algorithms meet the performances obtained with the RTK GPS sensor. In the worst case (sunny weather), performances are slightly damaged, but are still very satisfactory. Table 2 displays the extremum values of y recorded during curved parts of the trajectory, denoted C and E. Once more, it can be observed that guidance performances are roughly similar.

For these experiments, guidance accuracy seems to be limited by the ability of the control law to keep the robot on the trajectory. This explains the similarity between the two vision based lateral deviations (“Cloudy 2” and “Sunny 1” on Fig. 18). The difference between the RTK GPS and vision based control may be explained because vehicle heading is not computed in the same way with both sensors. This is especially visible in curves. When relying on the RTK GPS sensor, vehicle heading is inferred from velocities measurements (obtained by differentiating successive position data) under non-slipping

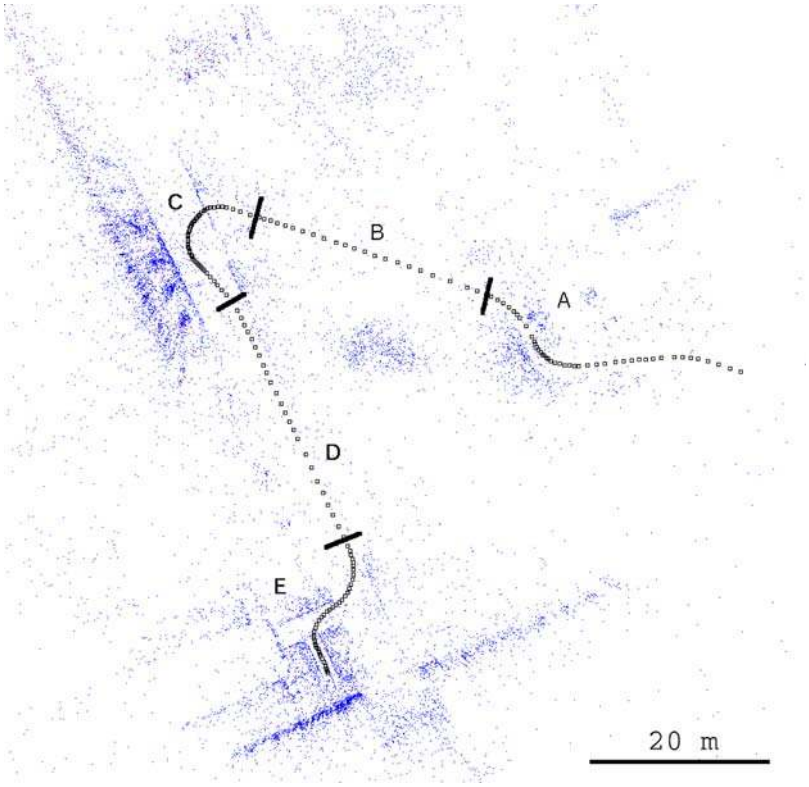

Figure 16. 3D reconstruction computed from the reference video sequence (top view). Black squares are the position of the key frames. The landmarks appear as dots. Letters indicate different parts of the trajectory.

Figure 17. Three frames taken during the autonomous navigation (bottom) and the corresponding key frames (top).

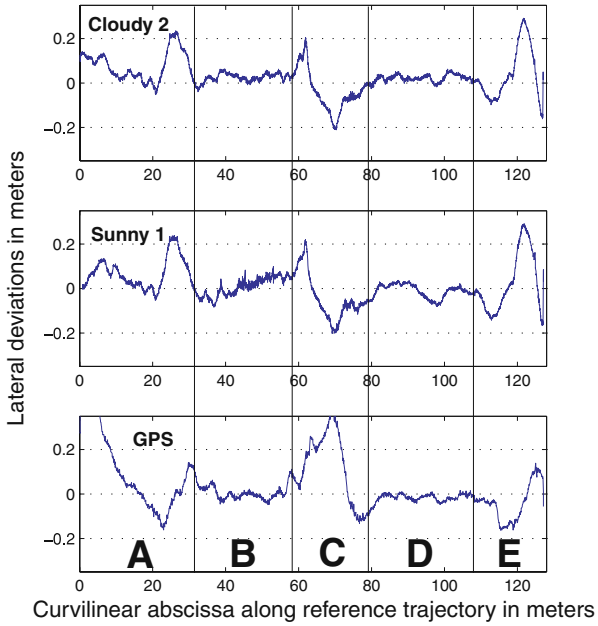

Figure 18. Lateral deviation from the reference trajectory.

assumptions, smoothed via a Kalman filter. The filter introduces a delay and the result may not be as accurate as the orientation computed with vision.

6.5. *Navigation on a Closed Loop*

The case of a closed loop is very interesting because it is a good way to visualize the influence of the reconstruction error on robot navigation. Of course, it would be possible to treat specifically the case of closed loops by searching point correspondences between the last image of the sequence and the first one in order to reduce reconstruction error. But that's not our purpose. We want to use closed loops as a way to illustrate what happens in the presence of reconstruction error. An example of a closed loop trajectory appears on Fig. 19 with a few images from the video sequence on Fig. 20. The position of the first and last cameras should be the same. But because of some drift in the reconstruction process, the reconstruction of the loop is not exactly closed (the gap measures 1.5 m). In that case, some points are reconstructed twice : once when they are seen at the beginning of the sequence and once when they are seen at the end of the loop. In spite of this reconstruction error, the robot could navigate continuously on this trajectory (for several loops), without a hint of jerky motion at the seam.

Table 1. Mean of the lateral deviation in straight lines.

	Sunny 1	Sunny 2	Cloudy 1	Cloudy 2	GPS
B	3.5 cm	4.8 cm	3.4 cm	2.8 cm	2.7 cm
D	2.4 cm	1.9 cm	1.8 cm	2.3 cm	1.8 cm

Table 2. Maximum and minimum deviation in curves.

	Sunny 1	Sunny 2	Cloudy 1	Cloudy 2	GPS
C max	22.0 cm	26.8 cm	20.1 cm	20.4 cm	37.9 cm
C min	−20.2 cm	−25.4 cm	−22.2 cm	−21.1 cm	−14.3 cm
E max	29.1 cm	35.4 cm	30.0 cm	29.2 cm	13.9 cm
E min	−16.5 cm	−19.7 cm	−16.5 cm	−16.1 cm	−16.3 cm

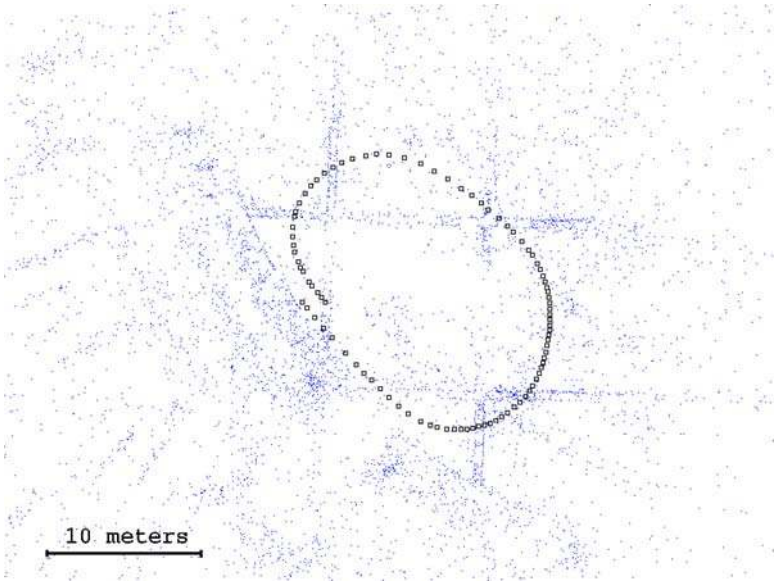

Figure 19. 3D Reconstruction for the loop sequence.

Figure 20. Some frames from the loop sequence.

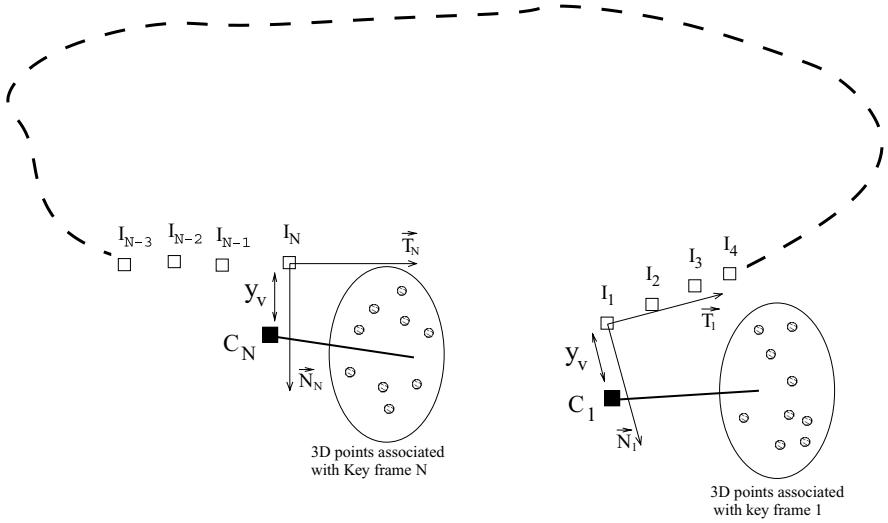

Figure 21. Lateral deviation computed at the beginning or at the end of a loop.

To understand why, we have to recall which variables are used in the control law. Along with the curvature of the path, the two other variables are the lateral deviation from the reference path y_v and the angular deviation. Figure 21 helps to understand what happens. The localization is computed with reference to the points cloud visible at one moment by the camera (the points which appear on only one key frame). The absolute localization is not used for the computation of the lateral and angular deviations. What is important is the localization in a local reference frame whose center is the position of the robot at the key frame. On Fig. 21, this local reference frame is drawn for the beginning of the loop (I_1, $\overrightarrow{T_1}$, $\overrightarrow{N_1}$) and for the end of the loop (I_N, $\overrightarrow{T_N}$, $\overrightarrow{N_N}$), where $\overrightarrow{T}$ and $\overrightarrow{N}$ are the unit vectors tangent and normal to the reference trajectory. The change from the end of a loop to the beginning of the next one corresponds to a change of the local reference frame which affects simultaneously the reference trajectory, the current localization, and the 3D points currently visible. In this case, the lateral deviation y_v has no discontinuity. The same is true for the angular deviation.

This characteristic suggests that it should be possible to link several short trajectories (of about 100 m) to provide the robot with enough information to travel for several kilometers. A global bundle adjustment would be too costly for trajectories of several kilometers, but we see here that a global optimization of this scale should not be necessary. Good accuracy on the small scale (50 m to 100 m) is enough.

6.6. *Navigation on a Path Different from the Learning Path*

6.6.1. Experiment. From a practical point of view, exactly following the path that was recorded can cause some problems. Ideally, if there is an obstacle on the trajectory, the robot should be able to go around it. That's were the expense of computing the 3D structure of the environment pays off. Knowing the 3D position of the interest points allows the robot to localize itself outside the learning path and to navigate in the neighborhood of this path. We made an experiment to see how much the robot could depart from the learning trajectory.

First, a learning video sequence was recorded on a 70 meters long trajectory. From this video sequence, a map was computed. A few images from the learning video sequence are shown on Fig. 22. This sequence was reconstructed with 102 key frames and 9579 points. For this experiment we defined a new trajectory (target trajectory), slightly different from the learning trajectory. The robot had to follow the target trajectory while using the learning trajectory and video sequence to localize itself. The experiment was conducted two weeks after the

Figure 22. Images extracted from the learning video sequence.

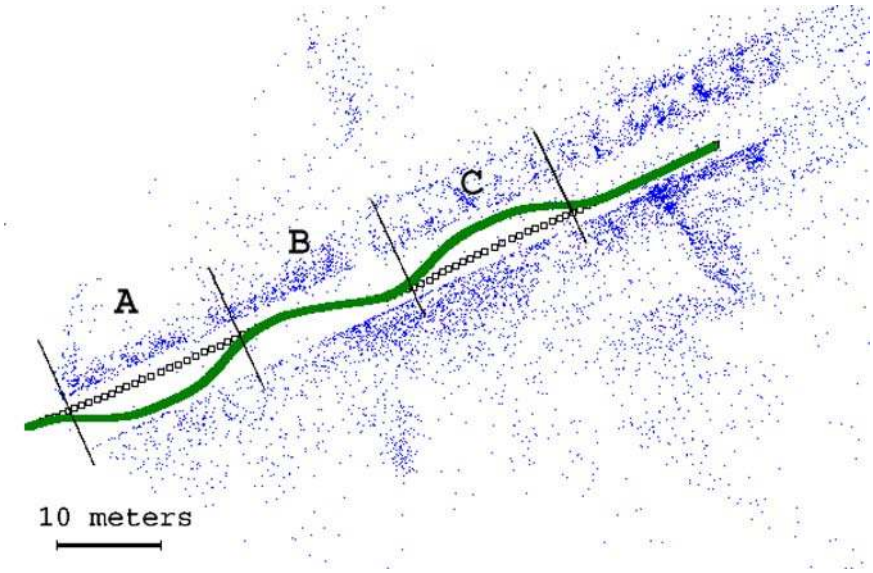

Figure 23. 3D reconstruction of the learning trajectory (black squares) and target trajectory (broad green line) viewed from the top.

learning trajectory was recorded. Figure 23 shows the 3D reconstruction of the learning sequence (black squares) and the target trajectory that was defined in a graphic editor (broad green line). The target trajectory was defined so that it reproduces what would happen if the robot had to avoid two obstacles. The target path departs from the learning path on the right at the beginning and then on the left a few meters later. The maximum lateral deviation from the learning trajectory is about 3 meters. The angular deviation is at most 20°. We parked a vehicle on the learning trajectory to add some occultations (see Fig. 24) to simulate the images the robot would use if it had to avoid a real obstacle. Some pedestrians were also passing by, occulting some parts of the image from time to time. The target trajectory was simply defined with the mouse in a graphical editor. Defining a new trajectory doesn't require expensive computations so this could be done online if necessary. For example, we could use a sensor (radar, sonar or laser range finder) to detect obstacles in front of the robot. If an obstacle was detected and localized by this sensor in the robot's coordinate system, it would be possible to modify the trajectory in 3D space in the same coordinate system without stopping the vehicle.

6.6.2. Localization and Navigation Results. Let's examine the path that was really followed by the robot during this experiment. We will call it the result path. It was accurately recorded by the RTK GPS sensor. The robot nearly followed the target path. This is visible on Fig. 25. The blue curve shows the lateral deviation of the target path with reference to the learning path. The red curve is the lateral deviation of the result path with reference to the learning path. The red curve was obtained by using the GPS measurements. The lateral deviation from the learning path directly computed from the result of the localization algorithm is displayed on the top row of Fig. 26. At this time, we don't use any filters to smooth the result of the localization algorithm. The result path deviates slightly from the target path in the turns. That is probably because we defined the target path in a graphical editor without knowing if the robot could steer fast enough to really follow the path. The lateral deviation computed with the vision algorithm matches the GPS measurements well in those parts of the trajectory.

The localization is more noisy when the robot departs from the learning trajectory but the localization is still accurate enough to drive the robot. We recorded the ground truth with the GPS so were are able to measure the error

Figure 24. Interest points correctly matched between the reference key frame and the current image. Left: when the robot is nearly on the learning path. Right: when the lateral deviation is maximal. The reference frame is on top.

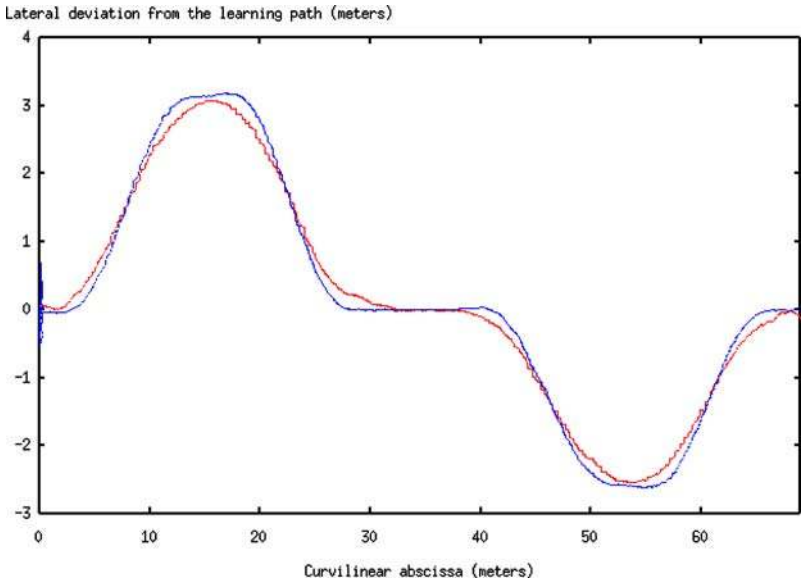

Figure 25. Lateral deviation of the target trajectory (blue), and lateral deviation measured by the GPS during the navigation experiment (red).

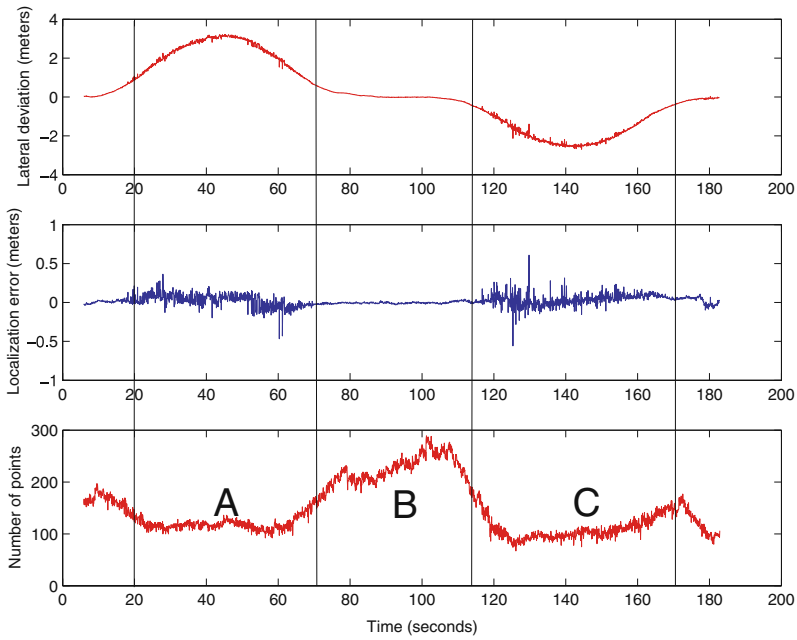

Figure 26. Lateral deviation computed by the vision algorithm (top), Lateral deviation error (middle) and number of interest points correctly matched (bottom).

that was made by the localization algorithm when computing the lateral deviation from the target path. This error appears on the middle of Fig. 26. When the robot is on the learning path, the error is less than 2 centimeters (the same accuracy as our GPS). When the robot is farther from the learning path, this error increases to more than 10 centimeters. The reason lies in the number of points matched. Figure 24 shows the interest points that were correctly matched between the current frame and the reference frame for two robot positions during the navigation experiment. We can see that the algorithm matches less points when the robot is far from the learning path. This is also visible at the bottom of Fig. 26 which displays the number of points used in the localization process for the whole experiment. When the robot is on the learning path, more than 200 points are matched. This number drops to 100 when the robot is 3 meters away. What is also important to notice is that the points close to the camera are lost first, and those points are those which provide the best positional accuracy. Points at infinity on the other hand can only give some rotational information. Table 3 shows how localization accuracy is related to the number of points matched. The graph of Fig. 26 was divided in three parts. Zones A and C correspond to parts where the robot was far from the learning trajectory and zone B correspond to a part where the robot was on the reference trajectory. For each zone, the average number of points matched was computed as well as the standard deviation of the error of the vision algorithm.

Table 3. Number of points matched and localization error.

Zone	A	B	C
Time limits	20–70 s	70–115 s	115–170 s
Average number of points	119	220	110
Localization error	8 cm	1 cm	8 cm

The robot can depart from the learning trajectory up to a limit which depends on the number of points that can be matched between the reference video sequence and the current image. Points can be lost because they may be seen from a different point of view and the correlation score is not high enough to make a valid match. This is the case if the lateral deviation is large. When angular deviation is important, points are lost because they go out of the field of view of the camera. Nothing can be done to correct that except going to an omnidirectional camera. On the other hand, we could take into account the difference in point of view by using wide baseline matching techniques or by predicting the appearance of the landmarks based on the camera position.

6.6.3. Validation of the Localization Uncertainty. The aim of this analysis is to make sure that the size of the confidence ellipsoids varies in the same way as the localization errors measured with the RTK GPS. The experiment presented in Section 6.6 gives us an opportunity to make an experimental validation of the computation of the localization uncertainty. In the experiments described in paragraph 6.3.1, the confidence ellipsoids were too small

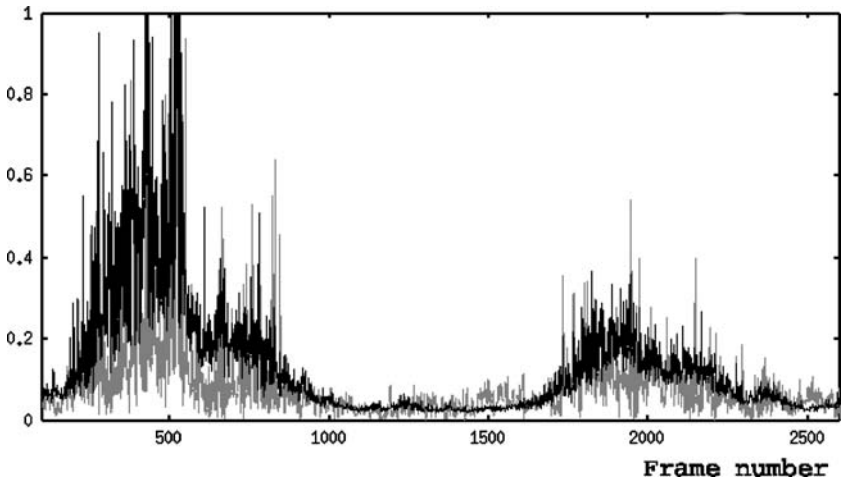

Figure 27. Comparison between the major semiaxis of the 90% confidence ellipsoid computed with the vision algorithm (black) and localization error measured with the RTK GPS (grey). Vertical scale in meters.

compared to the accuracy of the RTK GPS to have some meaningful results.

For each video frame taken during the autonomous navigation, we computed the length a of the major semiaxis of the 90% confidence ellipsoid with methods 2, 3 and 4 as explained in Section 4. Method 1 was not included in this comparison because it is not a real-time method. The localization error ϵ was also computed by using the RTK GPS measurements in a way slightly different from what is explained in Section 6.1.2. In Section 6.1.2, the measurements from both sensors were brought in the same coordinate system and the position of the cycab was computed. Then the localization error was measured along the normal to the trajectory. Here, the same computations are done to bring the data sets from both sensors in the same coordinate system, but the localization error is the distance between the position computed with vision and the position computed with the GPS. It includes the error along the normal and the error along the tangent to the trajectory. Figure 28 shows the localization error ϵ compared to the length a_2 of the major semiaxis of the 90% confidence ellipsoid computed with method 2. The major semiaxis computed with method 2 is roughly of the same size as the localization error and that both quantities vary in the same way. This experiment shows that method 2 is a valid method for computing the localization uncertainty.

A comparison can be made between the three methods used to compute the localization uncertainty. Figure 28 shows for each method the histogram of $log_2(\frac{a}{\epsilon})$ for every frame in the video sequence. If we assume zero uncertainty on the cameras in the 3D reconstruction (in method 3), the length of the major semiaxis is nearly half of the length computed with method 2. Ignoring the uncertainty on the 3D points (in method 4) conducts to an even larger underestimation of the size of the ellipsoid.

6.7. *Computation Times*

The computation times given here were measured on a 3 GHz Pentium 4 processor with an image size of 512 × 384 pixels. The code of the interest point detection and matching uses the SSE2 instruction set. Off line map building for a sequence with 100 key frames such as the one on Fig. 23 takes about 15 min. For a larger sequence such as the one on Fig. 3 with approximately 300 key frames, map building is done in 60 min. Most of the time is spent in the bundle adjustment. The localization runs at 15 frames per second (which is also the frame rate of our camera). The computation time can be roughly divided as: 35 ms for interest points detection, 15 ms for matching, 10 ms for the pose computation and 5 ms for the uncertainty computation.

6.8. *Database Management*

The memory needed to store the 3D model of the environment is very reasonable because we don't need to keep the whole key frames. We need to keep only image data around the interest points to compute the ZNCC as well as the 2D and 3D coordinates of the points. With about 150 bytes per interest point and about 500 interest point per key frame, each key frame takes less than 100 Kb of memory. Since the model is built with one key frame per meter on average, storing a 1 kilometer long trajectory needs about 100 Mb. Computers can handle this amount of data without difficulty.

From a practical point of view, keeping the model of the environment up to date is needed even if localization is possible with some modifications of the environment (see Royer et al. (2005c) or Fig. 24 for example). We have made a lot of experiments during the past two years

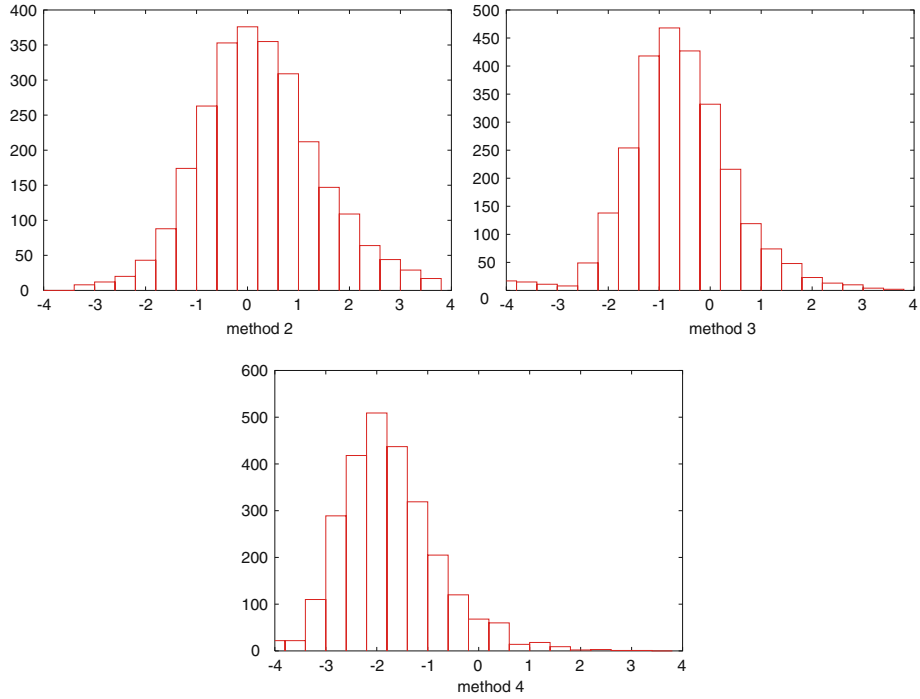

Figure 28. Histogram of $log_2(\frac{a}{\epsilon})$ for methods 2, 3 and 4.

in different places. We found that once a model is built, it can be used during a few weeks or more depending of the season and the kind of place. Buildings don't change much, but trees can change quickly in spring and fall. We have not developed a method to update the model yet, but we have some ideas about how to do that. During autonomous navigation, the pose of the camera is computed and the interest points are detected for each frame. We can store this data every time the camera is near a key frame so that at the end of a navigation experiment we have all the necessary information to update the database. Then we could match the interest points in the sequence recorded and compute the 3D position of the points. A bundle adjustment could be used to refine the structure. New points could be added to the model, and points which have not been used since a long time could be removed.

7. Conclusion

We have presented a sensing device which enables a robotic vehicle to follow a trajectory obtained from a human guided experience, relying uniquely on monocular vision. When following the same path, the localization accuracy is approximately 2 cm, nearly the same as a RTK GPS sensor. The vision system also provides with an orientation accurate to 0.1°. In practice, both sensors allow the robot to navigate autonomously with the same accuracy. When the robot departs from the learning path, the performance of the vision algorithm is somewhat degraded but still satisfactory in the case of obstacle avoidance for example. These two sensing devices appear complementary: autonomous navigation in urban environments cannot satisfactorily be addressed by RTK GPS sensors since tall buildings can disturb satellite receiving. These buildings however offer a lot of visual features which can be used to feed vision algorithms.

The main difficulty with the vision algorithm is to keep a map of the environment up to date. 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.

Acknowledgment

This work was done as part of the ROBEA CNRS Bodega project and the PREDIT MobiVip project. We are grateful to Heudiasyc at Université de Technologie de Compiègne for providing some of the video sequences used in this paper.

References

1. Araújo, H., Carceroni, R.J., and Brown, C.M. 1998. A fully projective formulation to improve the accuracy of Lowe's pose estimation algorithm. *Computer Vision and Image Understanding*, 70(2):227–238.
2. Argyros, A., Bekris, K., Orphanoudakis, S., and Kavraki, L. 2005. Robot homing by exploiting panoramic vision. *Journal of Autonomous Robots*, 19(1):7–25.
3. Blanc, G., Mezouar, Y., and Martinet, P. 2005. Indoor navigation of a wheeled mobile robot along visual routes. In *IEEE International Conference on Robotics and Automation, ICRA'05*, Barcelonne, Espagne.
4. Cobzas, D., Zhang, H., and Jagersand, M. 2003. Image-based localization with depth-enhanced image map. In *International Conference on Robotics and Automation*.
5. 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.
6. de Wit, C.C., Siciliano, B., and Bastin, G. 1996. *The Zodiac, Theory of Robot Control*. Springer Verlag.
7. Faugeras, O. and Herbert, M. 1986. The representation, recognition, and locating of 3-d objects. *International Journal of Robotic Research*, 5(3):27–52.
8. 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*, 24:381–395.
9. Georgiev, A. and Allen, P.K. 2004. Localization methods for a mobile robot in urban environments. *IEEE Transactions on Robotics*, 20(5):851–864.
10. Goedemé, T., Tuytelaars, T., Van Gool, L., Vanacker, G., and Nuttin, M. 2005. Feature based omnidirectional sparse visual following. In *International Conference on Intelligent Robots and Systems*, pp. 1003–1008.
11. 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):331–356.
12. Harris, C. and Stephens, M. 1988. A combined corner and edge detector. In *Alvey Vision Conference*, pp. 147–151.
13. Hartley, R. and Zisserman, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press.
14. 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):124–1332.
15. 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.
16. Lhuillier, M. and Perriollat, M. 2006. Uncertainty ellipsoids calculations for complex 3d reconstructions. In *International Conference on Robotic and Automation*.
17. Matsumoto, Y., Inaba, M., and Inoue, H. 1996. Visual navigation using view-sequenced route representation. In *International Conference on Robotics and Automation*, pp. 83–88.
18. Nistér, D. 2001. Frame decimation for structure and motion. In *2nd Workshop on Structure from Multiple Images of Large Environments, Springer Lecture Notes on Computer Science*, vol. 2018, pp. 17–34.
19. Nistér, D. 2003. An efficient solution to the five-point relative pose problem. In *Conference on Computer Vision and Pattern Recognition*, pp. 147–151.
20. Nistér, D., Naroditsky, O., and Bergen, J. 2004. Visual odometry. In *Conference on Computer Vision and Pattern Recognition*, pp. 652–659.
21. Remazeilles, A., Chaumette, F., and Gros, P. 2004. Robot motion control from a visual memory. In *International Conference on Robotics and Automation*, vol. 4, pp. 4695–4700.
22. Royer, E., Bom, J., Dhome, M., Thuilot, B., Lhuillier, M., and Marmoiton, F. 2005a. Outdoor autonomous navigation using monocular vision. In *International Conference on Intelligent Robots and Systems*, pp. 3395–3400.
23. Royer, E., Lhuillier, M., Dhome, M., and Chateau, T. 2005b. Localization in urban environments: Monocular vision compared to a differential gps sensor. In *International Conference on Computer Vision and Pattern Recognition, CVPR*.
24. Royer, E., Lhuillier, M., Dhome, M., and Lavest, J.-M. 2005c. Performance evaluation of a localization system relying on monocular vision and natural landmarks. In *Proceedings of the ISPRS Workshop BenCOS (Towards Benchmarking Automated Calibration, Orientation and Surface Reconstruction from Images)*.
25. Samson, C. 1995. Control of chained systems. application to path following and time-varying point stabilization of mobile robots. *IEEE Transactions on Automatic Control*, 40.
26. Se, S., Lowe, D., and Little, J. 2002. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotic Research*, 21(8):735–760.
27. Simond, N. and Rives, P. 2004. Trajectography of an uncalibrated stereo rig in urban environments. In *International Conference on Intelligent Robot and System*, pp. 3381–3386.
28. Thuilot, B., Bom, J., Marmoiton, F., and Martinet, P. 2004. Accurate automatic guidance of an urban vehicle relying on a kinematic gps sensor. In *Symposium on Intelligent Autonomous Vehicles IAV04*.
29. Torr, P., Fitzgibbon, A., and Zisserman, A. 1999. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. *International Journal of Computer Vision*, 32(1):27–44.
30. Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. 2000. Bundle adjustment—A modern synthesis. In *Vision Algorithms: Theory and Practice*, W. Triggs, A. Zisserman, and R. Szeliski (Eds.), Lecture Notes in Computer Science, pp. 298–375. Springer Verlag.
31. Vacchetti, L., Lepetit, V., and Fua, P. 2003. Stable 3-d tracking in real-time using integrated context information. In *Conference on Computer Vision and Pattern Recognition*, Madison, WI.