Outdoor autonomous navigation using monocular vision

Eric Royer, Jonathan Bom, Michel Dhome, Benoît Thuilot, Maxime Lhuillier, Francois Marmoiton

Outdoor autonomous navigation using monocular vision

Eric Royer, Jonathan Bom, Michel Dhome, Benoit Thuilot, Maxime Lhuillier and François Marmoiton

LASMEA UMR6602 CNRS and Blaise Pascal University
24 Avenue des Landais 63177 Aubière CEDEX
Eric.ROYER@lasmea.univ-bpclermont.fr
http://wwwlasmea.univ-bpclermont.fr/Personnel/Eric.Royer/

***Abstract*— In this paper, a complete system for outdoor robot navigation is presented. It uses only monocular vision. The robot is first guided on a path by a human. During this learning step, the robot records a video sequence. From this sequence a three dimensional map of the trajectory and the environment is built. When this map has been computed, the robot is able to follow the same trajectory by itself. Experimental results carried out with an urban electric vehicle are shown and compared to the ground truth.**

***Index Terms*— robot navigation, monocular vision, map building, urban vehicles.**

I. Introduction

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 actual 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. The only sensor we use is a camera and there is no artificial landmark. In this paper we present the whole system : the vision algorithms for learning the reference trajectory and providing real time localization, and the control law used.

For outdoor robotic vehicles, a common way of providing a localization to the robot is the use of a GPS sensor. A 1 cm accuracy, consistent with guidance applications, can be obtained with Real-Time Kinematic (RTK) GPS (Differential GPS can only provide with a 50 cm accuracy). Nevertheless this sensor has also some limitations. First, the accuracy depends on the position of the satellites. In urban environments, especially in narrow streets called urban canyons, buildings can occult the visibility of satellites. In that case the accuracy drops considerably. To overcome these occultations, the fusion with odometry must also be used, see [2]. Secondly, a kinematic GPS is very expensive. Alternatively, accurate vehicle localization can also be supplied by odometry, see [18], or by vision system. This latter device, considered in this paper, is here quite attractive since in city centers, there are a lot of visual features that can be used to compute a satisfactory localization.

The problem of navigation with reference to a human guided experience has already been studied. Two main approaches have been proposed. The first approach uses only images. Key frames are extracted from the reference sequence and the robot must go from one frame to the next. To do that, the robot can compute its relative position compared to the key frame and move accordingly, see [11]. However, computing a relative position is not necessary since moving from one key frame to the next can also be done by visual servoing as in [12].

Another way to address the problem is to first build a map of the trajectory and the environment. Building a map is time consuming but it is done offline and the localization is faster when a map has been computed beforehand. Having a localization in a global coordinates system also allows data fusion from different sensors (camera, GPS, odometry). In [8], two cameras and odometry are used to build a map of the environment, which is used to compute a global localization of the robot. We have chosen the same approach but with fewer constraints : we do not assume that the ground is planar and we use only one camera and no odometry. Simultaneous Localization And Mapping (SLAM) can be compared to this work with a big difference. In our case, map building is done off line, so a costly algorithm can be used to build a good map. Very often, SLAM is done with more complex sensors such as stereo vision, laser range finders, or with artificial landmarks. An example using trinocular vision and odometry can be found in [16]. Recently, SLAM using only monocular vision has been achieved [3]. But, in order to achieve real time performance, this approach assumes that the landmark database is kept small (under about 100 landmarks). This is well suited for computing localization in a room but not in a street where landmarks can be observed for a few meters and are replaced by new ones.

In section II we show how we process the reference video sequence to build a map and we also detail the localization algorithm. In section III we present the control law that is used. Finally, experimental results carried out with an urban electric vehicle are displayed in section IV.

II. Localization with monocular vision

A. Overview

The vision algorithm provides the localization and orientation of the camera in real time. This is done in two

steps. First, we build a 3D reconstruction of the learning video sequence. Because we use only one camera, this is a structure from motion problem. The computation is done off line. The second step is the real time localization process. The algorithms are briefly presented here. More details can be found in [13]

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 [6]. For each interest point in image 1, we select some candidate corresponding points in a search region in image 2. Then a Zero Normalized Cross Correlation score is computed between their neighborhood. And the pairs with the best scores are kept to provide a list of corresponding point pairs between the two images.

B. Map building

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 [7]. For our experiments, the camera was calibrated using a planar calibration pattern [9]. 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 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.

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 so that there are at least M common interest points between I_1 and I_2. When key frames $I_1 \dots I_n$ are chosen, we select I_{n+1} 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) Camera motion computation: We compute an initial solution for camera motion and a hierarchical bundle adjustment is used to refine this initial estimation.

For the first image triplet, the computation of the camera motion is done with the method described in [10] 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 [4] 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 [5]. 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 %.

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 $f(C_E^1, \cdots, C_E^N, P^1, \cdots, P^M)$ where C_E^i are the external parameters of camera i, and P^j are the world coordinates of point j. The cost function is the sum of the reprojection errors of all the inlier reprojections in all the images :

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

where $d^2(p_i^j, K_i P^j)$ is the squared euclidian distance between $K_i P^j$ the projection of point P^j by camera i, and p_i^j is the corresponding detected point. K_i is the 3×4 projection matrix built from the parameters values in C_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.

It is not a good idea to compute all the camera locations and use the bundle adjustment only once on the whole sequence. In that case, increasing errors could produce an initial solution too far from the optimal one for the bundle adjustment to converge. Thus it is necessary to use the bundle adjustment throughout the reconstruction of the sequence. The adjustment is done hierarchically as described in [7]. 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 sectionII-B.2 and we run a bundle adjustment over its three frames.

In order to merge two sequences S^1 and S^2, we use the last 2 cameras S^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 does not 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.

C. 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 K_0) for image I is the same as the pose computed for the preceding image. Based on K_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 IP_k reconstructed in 3D. We detect interest points in I and we match them with IP_k. To do that, for each point in IP_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 IP_k we know a 3D position, so with K_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 K_1 for I. Then the pose is refined using the iterative method proposed by Araújo et al. [1] 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 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.

At this point we have a real time algorithm that is able to compute the pose of the camera. But since it is based on vision alone the coordinate system has its center at the optical center of the first camera, with the Z 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 with a vertical axis. 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. Even if the control law works on a ground plane, we compute the camera pose with 6 degrees of freedom. It allows us to track interest points even if the ground is irregular.

III. Control law

A. Vehicle Modeling

Before designing a control law, the vehicle must be modeled. Its working environments being urban areas covered at low speed, slipping can be ignored (confirmed by extensive tests). Therefore, a classical kinematic model can be considered. More precisely, the celebrated tricycle model, where the two actual front wheels are merged as a unique virtual wheel is here used, see Fig. 1. Vehicle configuration can be

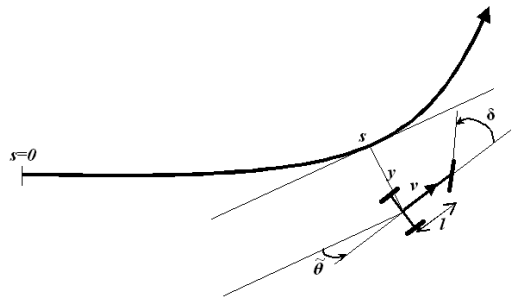

Fig. 1. Model tricycle description.

Fig. 2. Experimental Vehicle : Cycab with its camera

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 constituted in the vehicle linear velocity and the front wheel steering angle, denoted respectively v and δ. Vehicle model is then given (see e.g. [15]) by:

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

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.

B. Control Law Design

Reference path following is now addressed. More precisely, 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, nonlinear vehicle model (1) can be converted, in an exact way, into the so-called chained form, see [14]. $(a_1, a_2, a_3) = (s, y, (1-yc(s))\tan\tilde{\theta})$ 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 PD controller. The expression of the actual control variable δ can then be obtained by inverting the chained control transformation. Computations, detailed in [17], lead to ($K_p, K_d > 0$ are the PD gains):

$$\begin{aligned} \delta(y,\tilde{\theta}) = \quad & \arctan\Big(l\Big[\tfrac{\cos^3\tilde{\theta}}{(1-c(s)\,y)^2}\Big(\tfrac{d\,c(s)}{d\,s}\,y\tan\tilde{\theta} \\ & - K_d\,(1-c(s)\,y)\tan\tilde{\theta} - K_p\,y \\ & + c(s)\,(1-c(s)\,y)\tan^2\tilde{\theta}\Big) + \tfrac{c(s)\cos\tilde{\theta}}{1-c(s)\,y}\Big]\Big) \end{aligned} \tag{2}$$

IV. Experimental Results

The experimental electric vehicle, called Cycab, can transport simultaneously two passengers, see Fig. 2. Its small dimensions (length: 1.90 m, width: 1.20 m) are advantages for the urban traffic. This vehicle is entirely under computer control, and can be driven either manually with a joystick, or automatically. Its velocity can reach 18km/h. For experimentations, only the front wheels are steered, so the mobility is the same as a common car. A camera equipped with a fish eye lens (130° field of view) is fixed on the vehicle. A wide field is important because it reduces the risk of occultation by pedestrians or moving vehicles. For our experiments the image size is 640x480 pixels. Offline map building takes about one hour and realtime localization takes 60 ms per frame on a 3.4 GHz Pentium 4 processor.

To check the accuracy of our algorithm, we made the following experiment. First, a reference video sequence was recorded on a 127 m long trajectory. It was chosen so that both straight lines and tight turns compose this trajectory, and because the buildings are sometimes far (less visual features) and sometimes closer. From this video sequence, a map was computed. This map is used to localize the camera in real time in autonomous navigation. From the camera position and orientation, current value of vehicle state vector $(s, y, \tilde{\theta})^T$ is inferred and steering angle δ is computed from (2). Vehicle speed was chosen constant and equal to 2km/h. Finally, in order to record the ground truth, a kinematic GPS providing position measurements with a 2cm accuracy at 10Hz has been mounted on the Cycab.

The result of the structure from motion algorithm is displayed on Fig. 3 as seen from the top. There were 182

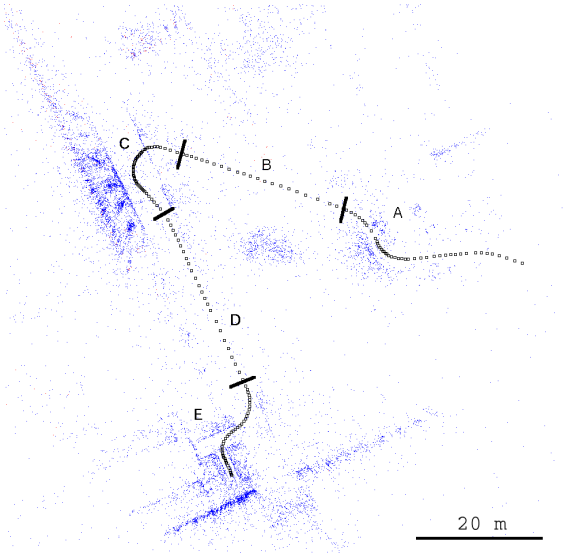

Fig. 3. 3D reconstruction computed from the reference video sequence (top view). Black squares are the position of the key frames. The landmarks appear as dots. Letters indicate different parts of the trajectory.

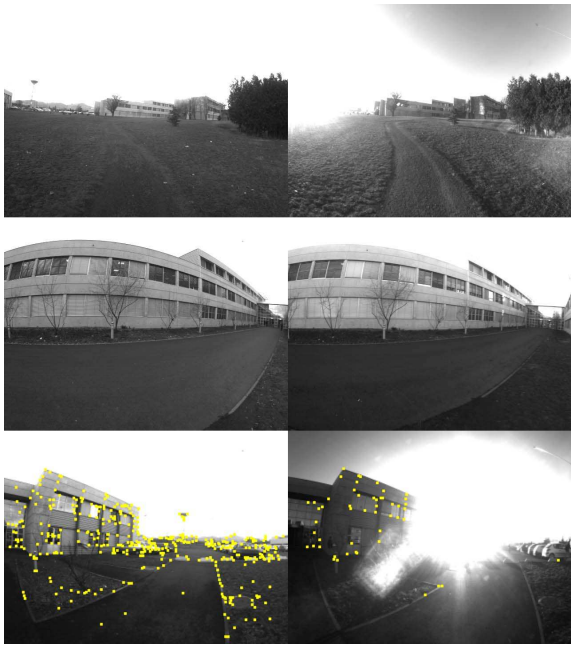

Fig. 4. 3 frames taken during the autonomous navigation on the right and the corresponding key frames on the left

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. 4. The third image outlines the necessity of having a wide field of view and local visual features all over the frame. It shows in yellow the interest points present in the database and those that are really used in the localization. The center of the image is completely overexposed because the sun is in front of the camera, but there are still some parts of the image which can be used for computing the camera pose.

In order to investigate path following performances achieved during the four above-mentioned experiments carried out with the camera sensor, vehicle lateral deviations have been recorded from the position measurements provided by the RTK GPS sensor. For comparison purposes, a fifth experiment has also been performed, relying on the RTK GPS sensor (instead of the vision algorithms) to provide vehicle state vector to control law (2). Lateral deviations recorded during 3 of these experiments are displayed with the same scale on Fig. 5. Letters enable to identify each part of the trajectory, with respect to the letters shown on Fig. 3.

It can be observed, on one hand 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. On the other hand, it can be noticed that these vision algorithms are reliable with respect

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

TABLE I

MEAN OF THE LATERAL DEVIATION IN STRAIGHT LINES

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

TABLE II

MAXIMUM AND MINIMUM DEVIATION IN CURVES

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 Table I and II. Table I reports the mean value of $|y|$ during straight lines part of the trajectory, denoted B and D. In the most favourable situation, i.e. cloudy weather, vision algorithms meet the performances obtained with the RTK GPS sensor, i.e. a very satisfactory centimeter guidance accuracy. In the worst case, i.e. sunny weather, performances are slightly damaged, but are still very satisfactory. Table II 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

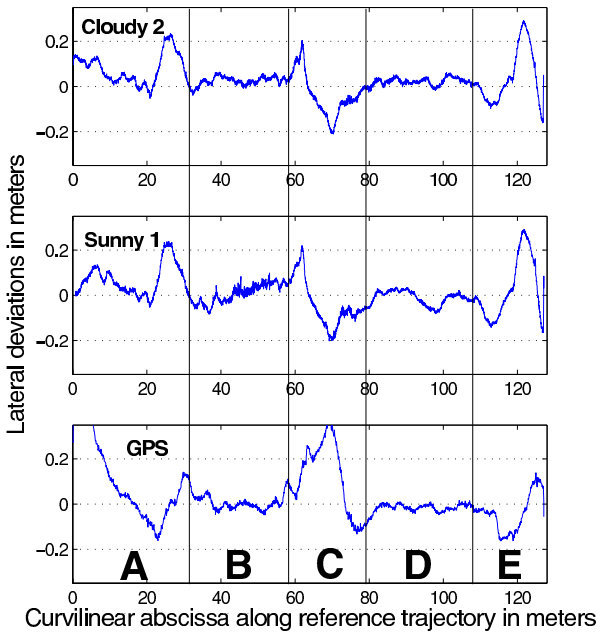

Fig. 5. Lateral deviation from the reference trajectory

are similar. During E-part, where curves do not exceed 90°, guidance obtained when relying on the RTK GPS sensor is slightly more accurate. On the contrary, during C-part, where a 150° curve is enclosed, vision algorithms enable to keep the vehicle closer to the reference path. It is a consequence of the way vehicle heading is evaluated. When relying on the RTK GPS sensor, vehicle heading is inferred from velocities measurements (obtained by differentiating successive position data) under non-slipping assumptions, smoothed via a Kalman filter. In sharp curves, non-slipping assumptions are no longer completely valid, and the delay introduced by the filter turns to be perceptible. In such situations, vehicle heading delivered by vision algorithms appears to be more accurate, and so are guidance performances.

V. Conclusion

We have presented a sensing device that enables a robotic vehicle to follow a trajectory obtained from a human guided experience, relying uniquely on monocular vision. Vision algorithms achieve a guidance accuracy similar to the one obtained when relying on a RTK GPS sensor, and are robust to large changes in illumination. 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. On the contrary, autonomous navigation on agricultural fields cannot satisfactorily be addressed by vision algorithms since they are varying environments, with very few visual features. In such applications, where no obstacle prevents from satellite receiving , RTK GPS appears as the more suitable device.

The accuracy of the localization is satisfactory as long as the robot stays on the reference trajectory. We plan to study the case where the robot needs to deviate from the learnt trajectory in order to avoid an obstacle. This could be done by using techniques developed for the SLAM problem in order to add landmarks to the database in real time.

References

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

[2] D. Bouvet, M. Froumentin, G. Garcia, "A real-time localization system for compactors.", *Automation in Construction*, 10:417–428, 2001.

[3] A. Davison, "Real-time simultaneaous localization and mapping with a single camera", *In proceeding of the Ninth International Conference on Computer Vision ICCV'03*, pp 1403-1410, Nice, France, 2003.

[4] M. Fischler and R. Bolles, "Random Sample Consensus: a Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography", *Commun. Assoc. Comp. Mach.*,24:381-395, 1981.

[5] R. Haralick, C. Lee, K. Ottenberg, M. Nolle, "Review and analysis of solutions of the three points perspective pose estimation problem", *International Journal of Computer Vision*, 1994.

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

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

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

[9] J. M. Lavest, M. Viala, M. Dhome, "Do we need an accurate calibration pattern to achieve a reliable camera calibration ?", *ECCV'98*, Friburg, Germany, 1, pp 158-174, June 1998.

[10] D. Nistér, "An efficient solution to the five-points relative pose problem", *2003 Conference on Computer Vision and Pattern Recognition*, volume II, June 2003.

[11] T. Ohno, A. Ohya, S. Yuta, "Autonomous navigation for mobile robots refering pre-recorded image sequence", *International Conference on Intelligent Robots and Systems*, 1996.

[12] A. Remazeilles, F. Chaumette, P. Gros, "Robot motion control from a visual memory", *In IEEE Int. Conf. on Robotics and Automation, ICRA'04*, Vol. 4, pp 4695-4700, April 2004

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

[14] C. Samson. "Control of chained systems. Application to path following and time-varying point stabilization of mobile robots.", *IEEE Transactions on Automatic Control*, 40(1):64–77, January 1995.

[15] The Zodiac, *Theory of robot control*, C. Canudas de Wit, B. Siciliano and G. Bastin, Eds, Springer Verlag Berlin, 1996.

[16] S. Se, D. Lowe, J. Little, "Mobile Robot Localization And Mapping with Uncertainty using Scale-Invariant Visual Landmarks", *International Journal of Robotic Research*, number 21, vol. 8, p. 735-760, August 2002.

[17] B. Thuilot, J. Bom, F. Marmoiton, P. Martinet, "Accurate automatic guidance of an urban vehicle relying on a kinematic GPS sensor", *In* 5^{th} *Symposium on Intelligent Autonomous Vehicles IAV04*, Lisboa, Portugal, July 2004

[18] R. Thrapp, C. Westbrook, D. Subramanian, "Robust localization algorithms for an autonomous campus tour guide.", *Proceeding of the 2001 IEEE International Conference on Robotics and Automation ICRA'01*, 2065–2071, Seoul, Korea, May 2001.