J Intell Robot Syst (2008) 51:113–134
DOI 10.1007/s10846-007-9182-5

UNMANNED SYSTEMS PAPER

3D Visual Odometry for Road Vehicles

R. García-García · M. A. Sotelo · I. Parra · D. Fernández · J. E. Naranjo · M. Gavilán

Received: 11 May 2007 / Accepted: 29 August 2007 /
Published online: 4 October 2007

Abstract This paper describes a method for estimating the vehicle global position in a network of roads by means of visual odometry. To do so, the ego-motion of the vehicle relative to the road is computed using a stereo-vision system mounted next to the rear view mirror of the car. Feature points are matched between pairs of frames and linked into 3D trajectories. Vehicle motion is estimated using the non-linear, photogrametric approach based on RANSAC. This iterative technique enables the formulation of a robust method that can ignore large numbers of outliers as encountered in real traffic scenes. The resulting method is defined as visual odometry and can be used in conjunction with other sensors, such as GPS, to produce accurate estimates of the vehicle global position. The obvious application of the method is to provide on-board driver assistance in navigation tasks, or to provide a means for autonomously navigating a vehicle. The method has been tested in real traffic conditions without using prior knowledge about the scene nor the vehicle motion. We provide examples of estimated vehicle trajectories using the proposed method and discuss the key issues for further improvement.

Keywords 3D visual odometry · Ego-motion estimation · Navigation assistance · RANSAC · Non-linear least squares

R. García-García · M. A. Sotelo (✉) · I. Parra · D. Fernández · M. Gavilán
Department of Electronics, Escuela Politécnica Superior, University of Alcalá,
Alcalá de Henares, Madrid, Spain
e-mail: miguel.sotelo@uah.es

J. E. Naranjo
Department of Informatics, Industrial Automation Institute,
CSIC, Arganda del Rey, Madrid, Spain

1 Introduction

The use of video sensors for vehicle navigation has become a research goal in the field of Intelligent Transportation Systems and Intelligent Vehicles in the last years. Accurate estimation of the vehicle global position is a key issue, not only for developing useful driver assistance systems, but also for achieving autonomous driving. Using stereo-vision for computing the position of obstacles or estimating road lane markers is a usual technique in intelligent vehicle applications. The challenge now is to extend stereo-vision capabilities to also provide accurate estimation of the vehicle ego-motion with regard to the road, and thus to compute the vehicle global position. This is becoming more and more tractable to implement on standard PC-based systems nowadays. However, there are still open issues that constitute a challenge in achieving highly robust ego-motion estimation in real traffic conditions. These are discussed in the following lines.

1) There must exist stationary reference objects that can be seen from the cameras position. Besides, the reference objects must have clearly distinguishable features that make possible to unambiguously perform matching between two frames. Accordingly, the selection of features becomes a critical issue.
2) Information contained on road scenes can be divided into road feature points and background feature points. On the one hand, roads have very few feature points, most of then corresponding to lane markings, or even no points in the case of unmarked roads. On the other hand, information corresponding to the background of road scenes may contain too many feature points. Robust matching techniques are then needed to avoid false matching.
3) Typical road scenes may contain a large amount of outlier information. This includes non-stationary objects such as moving vehicles, pedestrians, and car wipers. All these artifacts contribute to false measurements for ego-motion estimation. Possible solutions to overcome this problem are two fold: to deploy some outlier rejection strategy; to estimate feature points motion using probabilistic models in order to compensate for it in the estimation process.

In this paper, we propose a method for ego-motion computing based on stereo-vision. The use of stereo-vision has the advantage of disambiguating the 3D position of detected features in the scene at a given frame. Based on that, feature points are matched between pairs of frames and linked into 3D trajectories. The idea of estimating displacements from two 3-D frames using stereo vision has been previously used in [1, 2] and [3]. A common factor of these works is the use of robust estimation and outliers rejection using RANSAC [4]. In [2] a so-called firewall mechanism is implemented in order to reset the system to remove cumulative error. Both monocular and stereo-based versions of visual odometry were developed in [2], although the monocular version needs additional improvements to run in real time, and the stereo version is limited to a frame rate of 13 images per second. In [5] a stereo system composed of two wide Field of View cameras was installed on a mobile robot together with a GPS receiver and classical encoders. The system was tested in outdoor scenarios on different runs under 150 m. In [6], trajectory estimation is carried out using visual cues for the sake of autonomously driving a car in inner-city conditions.

In the present work, the solution of the non-linear system equations describing the vehicle motion at each frame is computed under the non-linear, photogrametric

approach using RANSAC. The use of RANSAC allows for outliers rejection in 2D images corresponding to real traffic scenes, providing a method for carrying out visual odometry onboard a road vehicle. A clear contribution of this work is the optimization of the RANSAC parameters. Exhaustive experimentation has been conducted in this aspect in order to yield the really optimal RANSAC parameters. Indeed, a genetic algorithm was off-line run to set a comparison between the optimized RANSAC parameters achieved on-line by our method and the same parameters obtained off-line by an evolutionary algorithm performing exhaustive global search. The results were extremely similar. The optimization of RANSAC parameters allows the use of very few feature points, thus reducing the total computation time of the visual odometry method. The possible applications of this method are twofold: on the one hand, the visual odometry system can serve to complement a GPS-based global navigation system. On the other hand, visual odometry can be used for simultaneous localization and mapping tasks (SLAM). Several examples are provided in order to show the trajectories estimated in real traffic conditions using the described method. A general layout of the system is depicted in Fig. 1.

The rest of the paper is organized as follows: in Section 2 the feature detection and matching technique is presented; Section 3 provides a description of the proposed

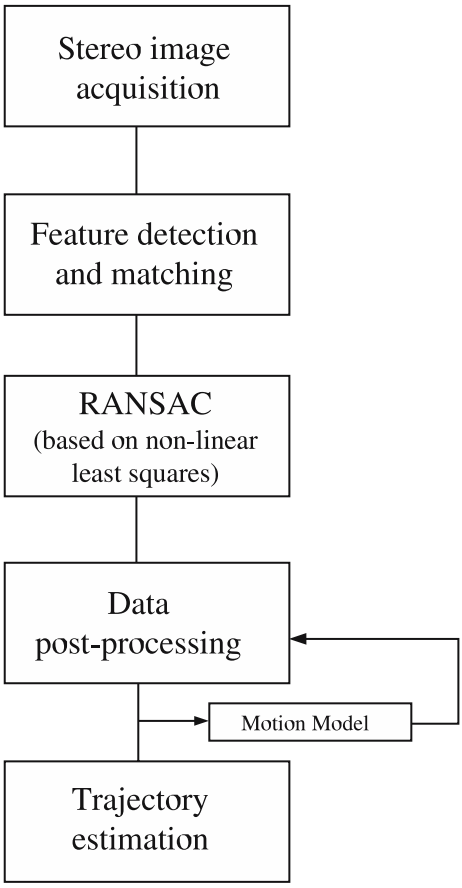

Fig. 1 General layout of the visual odometry method based on RANSAC

non-linear method for estimating vehicle ego-motion and the 3D vehicle trajectory; implementation and results are provided in Section 4; finally, Section 5 is devoted to conclusions and discussion about how to improve the current system performance in the future.

2 Features Detection and Matching

In each frame, Harris corners [7] are detected, since this type of point feature has been found to yield detections that are relatively stable under small to moderate image distortions [8]. As stated in [2], distortions between consecutive frames can be regarded as fairly small when using video input [2]. In order to reduce the computation time and to remove irrelevant features that move in the central part of the image, the method is only run in the lower left and right parts of the image, where significant features are most frequently located. The feature points are matched at each frame, using the left and right images of the stereo-vision arrangement, and between pairs of frames. Features are detected in all frames and matches are allowed only between features. A feature in one image is matched to every feature within a fixed distance from it in the next frame, called disparity limit. For the sake of real-time performance, matching is computed over a 7×7 window.

Among the wide spectrum of matching techniques that can be used to solve the correspondence problem we implemented the *zero mean normalized cross correlation* [9] because of its robustness. The normalized cross correlation between two image windows can be computed as follows.

$$\mathrm{ZMNCC}(p, p') = \frac{\sum_{i=-n}^{n}\sum_{j=-n}^{n} A \cdot B}{\sqrt{\sum_{i=-n}^{n}\sum_{j=-n}^{n} A^2 \sum_{i=-n}^{n}\sum_{j=-n}^{n} B^2}} \tag{1}$$

where A and B are defined by

$$A = \left(I\left(x+i, y+j\right) - \overline{I(x, y)}\right) \tag{2}$$

$$B = \left(I'\left(x'+i, y'+j\right) - \overline{I'(x', y')}\right) \tag{3}$$

where $I(x, y)$ is the intensity level of pixel with coordinates (x, y), and $\overline{I(x, y)}$ is the average intensity of a $(2n+1) \times (2n+1)$ window centered around that point. As the window size decreases, the discriminatory power of the area-based criterion gets decreased and some local maxima appear in the searching regions. On the contrary, an increase in the window size causes the performance to degrade due to occlusion regions and smoothing of disparity values across boundaries. In consequence, the correspondences yield some outliers. According to the previous statements, a filtering criteria is needed in order to provide outliers rejection. In order

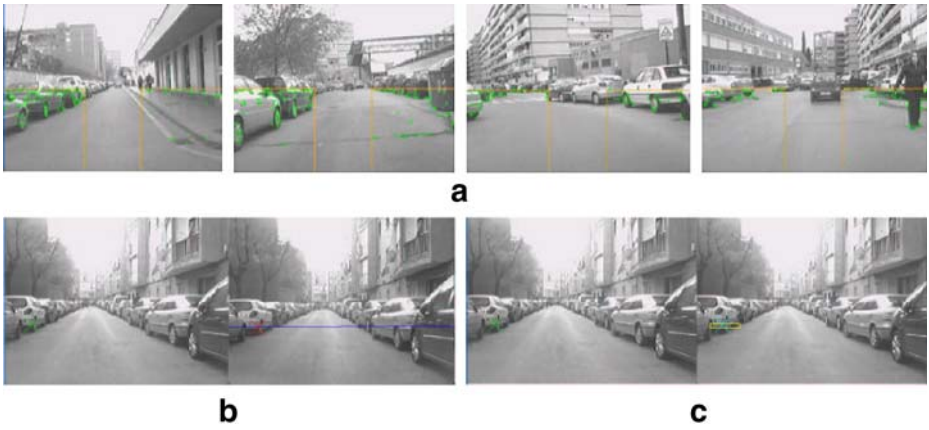

Fig. 2 **a** The *upper row* depicts feature detection results using Harris detector in several images in urban environments. Detection is constrained to a couple of regions of interest located in the *lateral areas of the image below the horizon line*. **b** The *bottom left image* shows an example of features matching in a stereo image. **c** The *bottom right image* depicts an example of feature tracking in two consecutive frames. ZMNCC and mutual consistency check is used both for feature detection and feature tracking

to minimize the number of outliers, mutual consistency check is used, as described in [2]. Accordingly, only pairs of features that yield mutual matching are accepted as a valid match. The accepted matches are used both in 3D feature detection (based on stereo images) and in feature tracking (between consecutive frames). Figure 2 depicts an example of features detection and tracking using Harris detector and ZMNCC matching technique.

The complete description of the feature detection and matching method can be found in [10], where a similar method was used for pedestrian detection purpose.

3 Visual Odometry Using Non-linear Estimation

The problem of estimating the trajectory followed by a moving vehicle can be defined as that of determining at frame i the rotation matrix $R_{i-1,i}$ and the translational vector $T_{i-1,i}$ that characterize the relative vehicle movement between two consecutive frames. The use of non-linear methods becomes necessary since the nine elements of the rotation matrix can not be considered individually (the rotation matrix has to be orthonormal). Indeed, there are only three unconstrained, independent parameters, i.e., the three rotation angles θ_x, θ_y and θ_z, respectively. The system's rotation can be expressed by means of the rotation matrix R given by Eq. 4.

$$R = \begin{pmatrix} cycz & sxsycz + cxsz & -cxsycz + sxsz \\ -cysz & -sxsysz + cxcz & cxsysz + sxcz \\ sy & -sxcy & cxcy \end{pmatrix} \tag{4}$$

where $ci = cos\theta_i$ and $si = sin\theta_i$ for $i = x, y, z$. The estimation of the rotation angles must be undertaken by using an iterative, least squares-based algorithm [4] that

yields the solution of the non-linear equations system that must compulsorily be solved in this motion estimation application. Otherwise, the linear approach can lead to a non-realistic solution where the rotation matrix is not orthonormal.

3.1 RANSAC

RANSAC (random sample consensus) [11, 12] is an alternative to modifying the generative model to have heavier tails to search the collection of data points S for good points that reject points containing large errors, namely "outliers". The algorithm can be summarized in the following steps:

1) Draw a sample s of n points from the data S uniformly and at random.
2) Fit to that set of n points.
3) Determine the subset of points S_i for whom the distance to the model s is bellow the threshold Dt. Subset S_i (defined as consensus subset) defines the inliers of S.
4) If the size of subset S_i is larger than threshold T the model is estimated again using all points belonging to S_i. The algorithm ends at this point.
5) Otherwise, if the size of subset S_i is below T, a new random sample is selected and steps 2, 3, and 4 are repeated.
6) After N iterations (maximum number of trials), draw subset S_{ic} yielding the largest consensus (greatest number of "inliers"). The model is finally estimated using all points belonging to S_{ic}.

RANSAC is used in this work to estimate the rotation matrix R and the translational vector T that characterize the relative movement of a vehicle between two consecutive frames. The input data to the algorithm are the 3D coordinates of the selected points at times t and $t+1$. Notation t_0 and $t_1 = t_0 + 1$ is used to define the previous and current frames, respectively, as in the next equation.

$$\begin{pmatrix} {}^1x_i \\ {}^1y_i \\ {}^1z_i \end{pmatrix} = \mathbf{R}_{0,1} \begin{pmatrix} {}^0x_i \\ {}^0y_i \\ {}^0z_i \end{pmatrix} + \mathbf{T}_{0,1}; \quad i = 1, \ldots, n \tag{5}$$

After drawing samples from three points, in step 1 models $\widetilde{R}_{0,1}$ and $\widetilde{T}_{0,1}$ that best fit to the input data are estimated using non-linear least squares. Then, a distance function is defined to classify the rest of points as inliers or outliers depending on threshold Dt.

$$\begin{cases} \text{inlier} & e < t \\ \text{outlier} & e \geqslant t \end{cases} \tag{6}$$

In this case, the distance function is the square error between the sample and the predicted model. The 3D coordinates of the selected point at time t_1 according to the predicted model are computed as:

$$\begin{pmatrix} {}^1\widetilde{x}_i \\ {}^1\widetilde{y}_i \\ {}^1\widetilde{z}_i \end{pmatrix} = \widetilde{\mathbf{R}}_{0,1} \begin{pmatrix} {}^0x_i \\ {}^0y_i \\ {}^0z_i \end{pmatrix} + \widetilde{\mathbf{T}}_{0,1}; \quad i = 1, \ldots, n \tag{7}$$

The error vector is computed as the difference between the estimated vector and the original vector containing the 3D coordinates of the selected points (input to the algorithm):

$$\mathbf{e} = \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} = \begin{pmatrix} {}^1\widetilde{x}_i \\ {}^1\widetilde{y}_i \\ {}^1\widetilde{z}_i \end{pmatrix} - \begin{pmatrix} {}^1x_i \\ {}^1y_i \\ {}^1z_i \end{pmatrix} \tag{8}$$

The mean square error or distance function for sample i is given by:

$$e = |\mathbf{e}|^2 = \mathbf{e}^t \cdot \mathbf{e} \tag{9}$$

In the following subsections, justification is provided for the choice of the different parameters used by the robust estimator.

3.1.1 Distance Threshold Dt

According to this threshold samples are classified as "inliers" or "outliers". Prior knowledge about the probability density function of the distance between "inliers" and model d_i^2 is required. If the measurement noise can be modelled as a zero-mean Gaussian function with standard deviation σ, d_i^2 can then be modelled as a chi-square distribution. In spite of that, distance threshold is empirically chosen in most practical applications. In this work, a threshold of $Dt = 0.005$ m^2 was chosen.

3.1.2 Number of Iterations N

Normally, it is non viable or unnecessary to test all the possible combinations. In reality, a sufficiently large value of N is selected in order to assure that at least one of the randomly selected s samples is outlier-free with a probability p. Let ω be the probability of any sample to be an inlier. Consequently, $\epsilon = 1 - \omega$ represents the probability of any sample to be an outlier. At least, N samples of s points are required to assure that $(1 - \omega^s)^N = 1 - p$. Solving for N yields:

$$N = \frac{\log(1 - p)}{\log(1 - (1 - \epsilon)^s)} \tag{10}$$

In this case, using samples of three points, assuming $p = 0.99$ and a proportion of outliers $\epsilon = 0.25$ (25%), at least nine iterations are needed. In practice, the final selected value is $N = 10$.

3.1.3 Consensus Threshold T

The iterative algorithm ends whenever the size of the consensus set (composed of inliers) is larger than the number of expected inliers T given by ϵ and n:

$$T = (1 - \epsilon)n \tag{11}$$

3.2 Non-linear Least Squares

Given a system of n non-linear equations containing p variables:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_p) = b_1 \\ f_2(x_1, x_2, \ldots, x_p) = b_2 \\ \qquad\qquad \vdots \\ f_n(x_1, x_2, \ldots, x_p) = b_n \end{cases} \tag{12}$$

where f_i, for $i = 1, \ldots, n$, is a differentiable function from $\Re^p$ to $\Re$. In general, it can be stated that:

1) If $n < p$, the system solution is a $(p - n)$-dimensional subspace of $\Re^p$.
2) If $n = p$, there exists a finite set of solutions.
3) If $n > p$, there exists no solution.

As can be observed, there are several differences with regard to the linear case: the solution for $n < p$ does not form a vectorial subspace in general. Its structure depends on the nature of the f_i functions. For $n = p$ a finite set of solutions exists instead of a unique solution as in the linear case. To solve this problem, an underdetermined system is built ($n > p$) in which the error function $E(x)$ must be minimized.

$$E(\mathbf{x}) \triangleq \sum_{i=1}^{N} (f_i(\mathbf{x}) - b_i)^2 \tag{13}$$

The error function $E : \Re^p \to \Re$ can exhibit several local minima, although in general there is a single global minimum. Unfortunately, there is no numerical method that can assure the obtaining of such global minimum, except for the case of polynomial functions. Iterative methods based on the gradient descent can find a global minimum whenever the starting point meets certain conditions. By using non-linear least squares the process is in reality linearized following the tangent linearization approach. Formally, function $f_i(x)$ can be approximated using the first term of Taylor's series expansion, as given by Eq. 14.

$$\begin{aligned} f_i(\mathbf{x} + \delta\mathbf{x}) = \; & f_i(\mathbf{x}) + \delta x_1 \frac{\partial f_i}{\partial x_1}(\mathbf{x}) + \ldots + \\ & + \delta x_p \frac{\partial f_i}{\partial x_p}(\mathbf{x}) + O(|\delta\mathbf{x}|)^2 \approx f_i(\mathbf{x}) + \nabla f_i(\mathbf{x}) \cdot \delta\mathbf{x} \end{aligned} \tag{14}$$

where $\nabla f_i(\mathbf{x}) = \left(\frac{\partial f_i}{\partial x_1}, \ldots, \frac{\partial f_i}{\partial x_p}\right)^t$ is the gradient of f_i calculated at point $\mathbf{x}$, neglecting high order terms $O(|\delta\mathbf{x}|)^2$. The error function $E(\mathbf{x} + \delta\mathbf{x})$ is minimized with regard to $\delta\mathbf{x}$ given a value of $\mathbf{x}$, by means of a iterative process. Substituting Eq. 14 in Eq. 12 yields:

$$\begin{aligned} E(\mathbf{x} + \delta\mathbf{x}) &= \sum_{i=1}^{N} (f_i(\mathbf{x} + \delta\mathbf{x}) - b_i)^2 \approx \\ &\approx \sum_{i=1}^{N} (f_i(\mathbf{x}) + \nabla f_i(\mathbf{x}) \cdot \delta\mathbf{x} - b_i)^2 = |\mathbf{J}\delta\mathbf{x} - \mathbf{C}|^2 \end{aligned} \tag{15}$$

where

$$\mathbf{J} = \begin{pmatrix} \nabla f_1(\mathbf{x})^t \\ \dots \\ \nabla f_n(\mathbf{x})^t \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial f_1}{\partial x_p}(\mathbf{x}) \\ \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial f_n}{\partial x_p}(\mathbf{x}) \end{pmatrix} \tag{16}$$

and

$$\mathbf{C} = \begin{pmatrix} b_1 \\ \dots \\ b_n \end{pmatrix} - \begin{pmatrix} f_1(\mathbf{x}) \\ \dots \\ f_n(\mathbf{x}) \end{pmatrix} \tag{17}$$

After linearization, an overdetermined linear system of n equations and p variables has been constructed ($n < p$):

$$\mathbf{J}\delta\mathbf{x} = \mathbf{C}, \tag{18}$$

The system given by Eq. 18 can be solved using least squares, yielding:

$$\delta\mathbf{x} = (\mathbf{J}^t\mathbf{J})^{-1}\mathbf{J}^t\mathbf{C} = \mathbf{J}^\dagger\mathbf{C}. \tag{19}$$

where $\mathbf{J}^\dagger$ stands for the pseudoinverse matrix of $\mathbf{J}$. In practice, the system is solved in an iterative process, as described in the following lines:

1) An initial solution $\mathbf{x_0}$ is chosen
2) While ($E(\mathbf{x_i}) > e_{\min}$ and $i < i_{\max}$)
   - $\delta\mathbf{x_i} = \mathbf{J}(\mathbf{x_i})^\dagger\mathbf{C}(\mathbf{x_i})$
   - $\mathbf{x_{i+1}} = \mathbf{x_i} + \delta\mathbf{x_i}$
   - $E(\mathbf{x_{i+1}}) = E(\mathbf{x_i} + \delta\mathbf{x_i}) = |\mathbf{J}(\mathbf{x_i})\delta\mathbf{x_i} - \mathbf{C}(\mathbf{x_i})|^2$

where the termination condition is given by a minimum value of error or a maximum number of iterations.

3.3 3D Trajectory Estimation

Between instants t_0 and t_1 we have:

$$\begin{pmatrix} {}^1x_i \\ {}^1y_i \\ {}^1z_i \end{pmatrix} = \mathbf{R}_{0,1} \begin{pmatrix} {}^0x_i \\ {}^0y_i \\ {}^0z_i \end{pmatrix} + \mathbf{T}_{0,1}; \quad i = 1, \dots, N \tag{20}$$

Considering Eq. 4 it yields a linear six-equations system at point i, with six variables $\mathbf{w} = [\theta_x, \theta_y, \theta_z, t_x, t_y, t_z]^t$:

$$\begin{cases} {}^1x_i = cycz \cdot {}^0x_i + (sxsycz + cxsz) \cdot {}^0y_i + \\ \qquad + (-cxsycz + sxsz) \cdot {}^0z_i + t_x \\ {}^1y_i = -cysz \cdot {}^0x_i + (-sxsysz + cxcz) \cdot {}^0y_i + \\ \qquad + (cxsysz + sxcz) \cdot {}^0z_i + t_y \\ {}^1z_i = sy \cdot {}^0x_i - sxcy \cdot {}^0y_i + cxcy \cdot {}^0z_i + t_z \end{cases}$$

At each iteration k of the regression method the following linear equations system is solved (given the 3D coordinates of N points in two consecutive frames):

$$\mathbf{J}(\mathbf{w_k})\delta\mathbf{x_k} = \mathbf{C}(\mathbf{x_k}) \tag{21}$$

with:

$$\mathbf{J}(\omega) = \begin{pmatrix} J_{1,11} & J_{1,12} & J_{1,13} & J_{1,14} & J_{1,15} & J_{1,16} \\ J_{1,21} & J_{1,22} & J_{1,23} & J_{1,24} & J_{1,25} & J_{1,26} \\ J_{1,31} & J_{1,32} & J_{1,33} & J_{1,34} & J_{1,35} & J_{1,36} \\ J_{2,11} & J_{2,12} & J_{2,13} & J_{2,14} & J_{2,15} & J_{2,16} \\ J_{2,21} & J_{2,22} & J_{2,23} & J_{2,24} & J_{2,25} & J_{2,26} \\ J_{2,31} & J_{2,32} & J_{2,33} & J_{2,34} & J_{2,35} & J_{2,36} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ J_{N,11} & J_{N,12} & J_{N,13} & J_{N,14} & J_{N,15} & J_{N,16} \\ J_{N,21} & J_{N,22} & J_{N,23} & J_{N,24} & J_{N,25} & J_{N,26} \\ J_{N,31} & J_{N,32} & J_{N,33} & J_{N,34} & J_{N,35} & J_{N,36} \end{pmatrix}$$

$$\delta\mathbf{x_k} = [\delta\theta_{x,k}, \delta\theta_{y,k}, \delta\theta_{z,k}, \delta t_{x,k}, \delta t_{y,k}, \delta t_{z,k}]^t$$

$$\mathbf{C}(\mathbf{x_k}) = [c_{1,1}, c_{1,2}, c_{1,3}, \ldots, c_{N,1}, c_{N,2}, c_{N,3}]^t$$

Let us remark that the first index of each Jacobian matrix element represents the point with regard to whom the function is derived, while the other two indexes represent the position in the 3×6 sub-matrix associated to such point. Considering Eq. 16 the elements of the Jacobian matrix that form sub-matrix $\mathbf{J_i}$ for point i at iteration k are:

$$\begin{aligned}
J_{i,11} &= (cx_k sy_k cz_k - sx_k sz_k) \cdot {}^0Y_i + (sx_k sy_k cz_k + cx_k sz_k) \cdot {}^0Z_i \\
J_{i,12} &= -sy_k cz_k \cdot {}^0X_i + sx_k cy_k cz_k \cdot {}^0Y_i - cx_k cy_k cz_k \cdot {}^0Z_i \\
J_{i,13} &= -cy_k sz_k \cdot {}^0X_i + (-sx_k sy_k sz_k + cx_k cz_k) \cdot {}^0Y_i + (cx_k sy_k sz_k + sx_k cz_k) \cdot {}^0Z_i \\
J_{i,14} &= 1 \\
J_{i,15} &= 0 \\
J_{i,16} &= 0 \\
J_{i,21} &= -(cx_k sy_k sz_k + sx_k cz_k) \cdot {}^0Y_i + (-sx_k sy_k sz_k + cx_k cz_k) \cdot {}^0Z_i \\
J_{i,22} &= sy_k sz_k \cdot {}^0X_i - sx_k cy_k sz_k \cdot {}^0Y_i + cx_k cy_k sz_k \cdot {}^0Z_i \\
J_{i,23} &= -cy_k cz_k \cdot {}^0X_i - (sx_k sy_k cz_k + cx_k sz_k) \cdot {}^0Y_i + (cx_k sy_k cz_k - sx_k sz_k) \cdot {}^0Z_i \\
J_{i,24} &= 0 \\
J_{i,25} &= 1 \\
J_{i,26} &= 0 \\
J_{i,31} &= -cx_k cy_k \cdot {}^0Y_i - sx_k cy_k \cdot {}^0Z_i \\
J_{i,32} &= cy_k \cdot {}^0X_i + sx_k sy_k \cdot {}^0Y_i - cx_k sy_k \cdot {}^0Z_i \\
J_{i,33} &= 0 \\
J_{i,34} &= 0 \\
J_{i,35} &= 0 \\
J_{i,36} &= 1
\end{aligned}$$

Independent terms of the coefficients vector **c** are computed at iteration k as follows:

$$c_{i,1} = {}^1X_i - cy_k cz_k {}^0X_i - (sx_k sy_k cz_k + cx_k sz_k)^0Y_i + (cx_k sy_k cz_k - sx_k sz_k)^0Z_i - t_{x,k},$$
$$c_{i,2} = {}^1Y_i + cy_k sz_k {}^0X_i + (sx_k sy_k sz_k - cx_k cz_k)^0Y_i - (cx_k sy_k sz_k + sx_k cz_k)^0Z_i - t_{y,k},$$
$$c_{i,3} = {}^1Z_i - sy_k {}^0X_i + sx_k cy_k {}^0Y_i - cx_k cy_k {}^0Y_i - t_{z,k}.$$

Once the jacobian matrix **J** and vector **c** have been computed at iteration k system (21) is solved, solution $\mathbf{w}_{k+1} = \mathbf{w}_k + \delta\mathbf{w}_k$ is updated, and the previously explained iterative process is resumed in the next iteration ($k = k + 1$). On completion of the process (after $i_{\max}$ iterations as maximum) the algorithm yields the final solution $\mathbf{w} = [\theta_x, \theta_y, \theta_z, t_x, t_y, t_z]^t$ that describes the relative vehicle movement between two consecutive iterations at t_0 and t_1, respectively.

3.4 2D Approximation

Under the assumption that only 2D representations of the global trajectory are needed, like in a bird-eye view, the system can be dramatically simplified by considering that the vehicle can only turn around the y axis (strictly true for planar roads). It implies that angles θ_x and θ_z are set to 0, being θ_y estimated at each iteration.

Solving for the rotation matrix in Eq. 4 yields:

$$\mathbf{R} = \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{pmatrix}. \tag{22}$$

In the following, the rotation matrix obtained in Eq. 4 is used as the approximate rotation matrix in the mathematical development explained in the previous section for 3D motion estimation. Given Eqs. 20 and 22, the new equations system is:

$$\begin{cases} {}^1X_i = \cos\theta_y \cdot {}^0X_i - \sin\theta_y \cdot {}^0Z_i + t_x \\ {}^1Y_i = {}^0Y_i + t_y \\ {}^1Z_i = \sin\theta_y \cdot {}^0X_i + \cos\theta_y \cdot {}^0Z_i + t_z \end{cases} ; \quad i = 1, 2, \ldots, N$$

As observed, a non-linear equation with four unknown variables $\mathbf{w} = [\theta_y, t_x, t_y, t_z]^t$ is obtained. Thus, at least two points are needed to solve the system (or more than 2 points to solve the system using non-linear least squares). For each iteration k of the regression method, the following linear system has to be solved:

$$\mathbf{J}(\mathbf{w}_k)\delta\mathbf{w}_k = \mathbf{C}(\mathbf{w}_k). \tag{23}$$

where:

$$\mathbf{J}(\mathbf{w_k}) = \begin{pmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \vdots \\ \mathbf{J}_N \end{pmatrix} = \begin{pmatrix} J_{1,11} & J_{1,12} & J_{1,13} & J_{1,14} \\ J_{1,21} & J_{1,22} & J_{1,23} & J_{1,24} \\ J_{1,31} & J_{1,32} & J_{1,33} & J_{1,34} \\ J_{2,11} & J_{2,12} & J_{2,13} & J_{2,14} \\ J_{2,21} & J_{2,22} & J_{2,23} & J_{2,24} \\ J_{2,31} & J_{2,32} & J_{2,33} & J_{2,34} \\ \vdots & \vdots & \vdots & \vdots \\ J_{N,11} & J_{N,12} & J_{N,13} & J_{N,14} \\ J_{N,21} & J_{N,22} & J_{N,23} & J_{N,24} \\ J_{N,31} & J_{N,32} & J_{N,33} & J_{N,34} \end{pmatrix},$$

$$\delta\mathbf{w}_k = [\delta\theta_{y,k}, \delta t_{x,k}, \delta t_{y,k}, \delta t_{z,k}]^t,$$

$$\mathbf{C}(\mathbf{w}_k) = [c_{1,1}, c_{1,2}, c_{1,3}, \ldots, c_{N,1}, c_{N,2}, c_{N,3}]^t.$$

In this case, the Jacobian submatrix $\mathbf{J}_i$, associated to point i, has a dimension of $3x4$. The coefficients of $\mathbf{J}_i$ can be computed as:

$$\begin{array}{llll} J_{i,11} = -\sin\theta_{y,k} \cdot {}^0X_i - \cos\theta_{y,k} \cdot {}^0Z_i, & J_{i,12} = 1, & J_{i,13} = 0, & J_{i,14} = 0, \\ J_{i,21} = 0, & J_{i,22} = 0, & J_{i,23} = 1, & J_{i,24} = 0, \\ J_{i,31} = \cos\theta_{y,k} \cdot {}^0X_i - \sin\theta_{y,k} \cdot {}^0Z_i, & J_{i,32} = 0 & J_{i,33} = 0, & J_{i,34} = 1. \end{array}$$

The vector of independent terms $\mathbf{c}(\mathbf{w}_k)$ yields:

$$\begin{array}{l} c_{i,1} = {}^1X_i - \cos\theta_{y,k} \cdot {}^0X_i + \sin\theta_{y,k} \cdot {}^0Z_i - t_{x,k}, \\ c_{i,2} = {}^1Y_i - {}^0Y_i - t_{y,k}, \\ c_{i,3} = {}^1Z_i - \sin\theta_{y,k} \cdot {}^0X_i - \cos\theta_{y,k} \cdot {}^0Z_i - t_{z,k}. \end{array}$$

After computing the coefficients of $\mathbf{J}_i$ at iteration k, system (23) is solved and the iterative process is resumed. On completion of the process (after $i_{\max}$ iterations as maximum) the algorithm yields the final solution $\mathbf{w} = [\theta_y, t_x, t_y, t_z]^t$ that describes the relative vehicle movement between two consecutive iterations.

3.5 Data Post-processing

This is the last stage of the algorithm. Some partial estimations are discarded, in an attempt to remove as many outliers as possible, using the following criteria.

1) High root mean square error e estimations are removed.
2) Meaningless rotation angles estimations (non physically feasible) are discarded.

Accordingly, a maximum value of e has been set to 0.5. Similarly, a maximum rotation angle threshold is used to discard meaningless rotation estimations. In such

cases, the estimated vehicle motion is maintained according to motion estimated in the previous frame. Removing false rotation estimations is a key aspect in visual odometry systems since false rotation estimations lead to high cumulative errors. This is a remarkable contribution of this work.

3.6 Trajectory Representation

The *trajectory* can be defined as the set of points $\{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_N\}$ that represent the position of the camera with respect to the a stationary inertial reference system whose origin is the position of the camera at the starting time: $\mathbf{O}_c(t_0)$. In summary, given the global position of the vehicle at the starting time (that can be provided by a GPS receiver), the vehicle position and orientation are updated at each iteration of the algorithm. By integrating the successive estimations the complete trajectory can be retrieved in real time.

Accordingly, point $\mathbf{P}_k$ in the trajectory is the origin of the reference frame for one of the cameras (left camera, in this case) denoted in coordinates of the reference system (F_0) (starting time t_0):

$$\mathbf{P}_k = {}^0\mathbf{O}_c(t_k) = \left({}^0X_{t_k}, {}^0Y_{t_k}, {}^0Z_{t_k}\right)^t \tag{24}$$

for the case of 3D representation. For the simplified 2D representation case it yields ("bird-eye view"):

$$\mathbf{P}_k = {}^0\mathbf{O}_c(t_k) = \left({}^0X_{t_k}, {}^0Z_{t_k}\right)^t \tag{25}$$

After observing Fig. 3, it can be deduced that the camera position at a given time (or point in the trajectory) is computed by integrating the translational vector:

$$\mathbf{P}_k = {}^0\mathbf{T}_{0\to k} = {}^0\mathbf{T}_{0\to(k-1)} + {}^0\mathbf{T}_{(k-1)\to k} = \sum_{j=0}^{k-1} {}^0\mathbf{T}_{j\to(j+1)} = \mathbf{P}_{k-1} + {}^0\mathbf{T}_{(k-1)\to k} \tag{26}$$

where it must be remarked that all translational vectors are denoted in coordinates of the reference system (F_0). The visual odometry system estimates the rotation matrix ${}^k_{k-1}\mathbf{R}$ and the translational vector ${}^k\mathbf{T}_{k\to(k-1)}$ that describe the relative vehicle movement between two consecutive iterations $(k-1$ and $k)$.

$${}^k\mathbf{P} = {}^k_{k-1}\mathbf{R}\ {}^{k-1}\mathbf{P} + {}^k\mathbf{T}_{k\to(k-1)}. \tag{27}$$

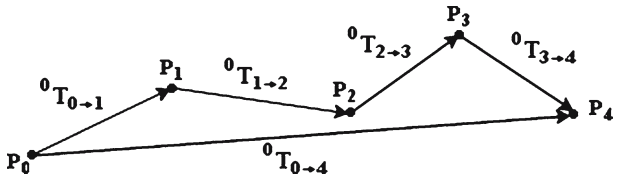

Fig. 3 Theoretical trajectory

Points $\mathbf{P}_k$ in the trajectory corresponding to the vehicle position at time t_k must be represented as a function of motion parameters estimated at that time ($^{k}_{k-1}\mathbf{R}$ and $^{k}\mathbf{T}_{k\to(k-1)}$) and those corresponding to the previous $(k-1)$ estimations. For this purpose, translational vectors are expressed as a function of the parameters that are known:

$$
\begin{aligned}
&{}^{0}\mathbf{T}_{0\to1} = {}^{0}_{1}\mathbf{R}\ {}^{1}\mathbf{T}_{0\to1} = -{}^{1}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{1}\mathbf{T}_{1\to0}\\
&{}^{0}\mathbf{T}_{1\to2} = {}^{0}_{2}\mathbf{R}\ {}^{2}\mathbf{T}_{1\to2} = -{}^{0}_{1}\mathbf{R}\ {}^{1}_{2}\mathbf{R}\ {}^{2}\mathbf{T}_{2\to1} = -{}^{1}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{2}_{1}\mathbf{R}^{\mathbf{t}}\ {}^{2}\mathbf{T}_{2\to1} = -{}^{2}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{2}\mathbf{T}_{2\to1}\\
&{}^{0}\mathbf{T}_{2\to3} = {}^{0}_{3}\mathbf{R}\ {}^{3}\mathbf{T}_{2\to3} = -{}^{1}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{2}_{1}\mathbf{R}^{\mathbf{t}}\ {}^{3}_{2}\mathbf{R}^{\mathbf{t}}\ {}^{3}\mathbf{T}_{3\to2} = -{}^{2}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{3}_{2}\mathbf{R}^{\mathbf{t}}\ {}^{3}\mathbf{T}_{3\to2} = -{}^{3}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{3}\mathbf{T}_{3\to2}\\
&\qquad\vdots\\
&{}^{0}\mathbf{T}_{(k-1)\to k} = {}^{0}_{k}\mathbf{R}\ {}^{k}\mathbf{T}_{(k-1)\to k} = -\left(\prod_{j=0}^{k-1}{}^{j+1}_{j}\mathbf{R}^{\mathbf{t}}\right){}^{k}\mathbf{T}_{k\to(k-1)} =\\
&\qquad = -{}^{k-1}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{k}_{k-1}\mathbf{R}^{\mathbf{t}}\ {}^{k}\mathbf{T}_{k\to(k-1)} = -{}^{k}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{k}\mathbf{T}_{k\to(k-1)}
\end{aligned}
\tag{28}
$$

where basic transformation properties are applied, yielding the following value for the k^{th} point in the trajectory:

$$
\begin{aligned}
\mathbf{P}_k &= \mathbf{P}_{k-1} + {}^{0}\mathbf{T}_{(k-1)\to k} =\\
&= \mathbf{P}_{k-1} - {}^{k}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{k}\mathbf{T}_{k\to(k-1)} =\\
&= \mathbf{P}_{k-1} - {}^{k-1}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{k}_{k-1}\mathbf{R}^{\mathbf{t}}\ {}^{k}\mathbf{T}_{k\to(k-1)}
\end{aligned}
\tag{29}
$$

Coordinates of point $\mathbf{P}_k$ are computed based on the coordinates of the previous point $\mathbf{P}_{k-1}$ by subtracting a term that contains the cumulative value of the variation of orientation and the variation of position between two consecutive samples $k-1$ y k, considering that the term $^{k}_{0}\mathbf{R}^{\mathbf{t}} = {}^{k-1}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{k}_{k-1}\mathbf{R}^{\mathbf{t}}$ represents the orientation at time k and $^{k}\mathbf{T}_{k\to(k-1)}$ represents motion in that direction between times $k-1$ and k.

The orientation term $^{k}_{0}\mathbf{R}^{\mathbf{t}}$ is given as function of vehicle orientation $^{k-1}_{0}\mathbf{R}^{\mathbf{t}}$ at time $k-1$ (cumulative rotation matrix) multiplied by the variation of orientation $^{k}_{k-1}\mathbf{R}^{\mathbf{t}}$ between times $k-1$ and k [rotation matrix between systems (F_{k-1}) and (F_k)]. As can be observed, the cumulation of orientation is carried out by multiplying the different rotation matrices.

Thus, trajectory $\{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_N\}$ is computed by means of the following recursive process (as a function of the rotation matrix and the translational vector relative to each interval):

1) $^{k}_{0}\mathbf{R}^{\mathbf{t}} = {}^{k-1}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{k}_{k-1}\mathbf{R}^{\mathbf{t}}$
2) $\mathbf{P}_k = \mathbf{P}_{k-1} - {}^{k}_{0}\mathbf{R}^{\mathbf{t}}\ {}^{k}\mathbf{T}_{k\to(k-1)}$; $\qquad k = 1, 2, \ldots, N$

being:

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{0} \\ {}^{0}_{0}\mathbf{R}^{\mathbf{t}} &= \mathbf{I} \end{aligned} \tag{30}$$

4 Implementation and Results

The visual odometry system described in this paper has been implemented on a Pentium IV at 1.7 GHz running Linux Knoppix 3.7 with a 2.4.18-6mdf kernel version. The algorithm is programmed in C using OpenCV libraries (version 0.9.7). A stereo vision platform based on Fire-i cameras (IEEE1394) was installed on a prototype vehicle. After calibrating the stereo vision system, several sequences were recorded in different locations including Alcalá de Henares and Arganda del Rey in Madrid (Spain). The stereo sequences were recorded using no compression algorithm at 30 frames/s with a resolution of 320 × 240 pixels. All sequences correspond to real traffic conditions in urban environments. In the experiments, the vehicle was driven bellow the maximum allowed velocity in cities, i.e., 50 km/h.

4.1 2D Visual Odometry Results

The general 3D visual odometry method described in this paper can be simplified in order to yield a 2D visual odometry system in which only the yaw angle is estimated, under the flat terrain assumption. The simplified 2D method is very useful in practice for trajectory estimation in short runs.

The results of a first experiment are depicted in Fig. 4. The vehicle starts a trajectory in which it first turns slightly to the left. Then, the vehicle runs along a straight street and, finally, it turns right at a strong curve with some 90° of variation in yaw. The upper part of Fig. 4 shows an aerial view of the area of the city (Alcalá de Henares) were the experiment was conducted (source: http://maps.google.com). The bottom part of the figure illustrates the 2D trajectory estimated by the visual odometry algorithm presented in this paper.

As can be observed, the system provides reliable estimations of the path run by the vehicle in almost straight sections. As a matter of fact, the estimated length of the straight section in Fig. 4b is 162.37 m, which is very similar to the ground truth (165.86 m). The estimated vehicle trajectory along the straight street is almost straight, similar to the real trajectory described by the vehicle in the experiment. Nonetheless, there are still some problems to estimate accurate rotation angles in sharp bends (90° or more). Rotation angles estimated by the system at strong curves tend to be higher than the real rotation experimented by the vehicle. This problem does not arise in the first left curve conducted by the vehicle, where the estimated rotation and the real rotation are very similar, as can be observed in Fig. 4. Figure 5 depicts the values of intermediate variables during the whole experiment. Figure 5a represents the cumulative estimated vehicle's yaw rate in a sequence of 1,200 frames acquired at a frame rate of 30 frames/s (thus, a duration of 40 s). Figure 5b shows the number of outliers, i.e., the number of feature points rejected at each frame. In

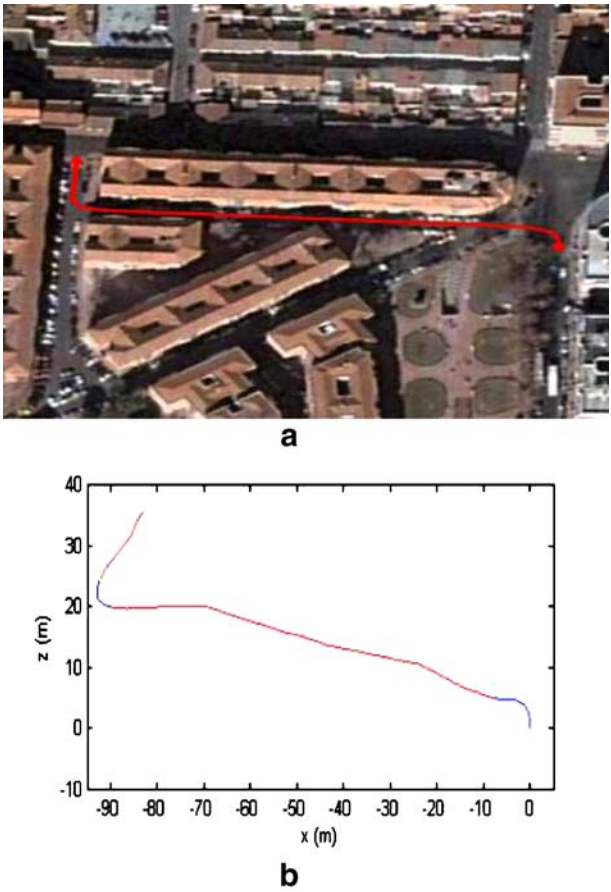

Fig. 4 **a** Aerial view of the area of the city were the experiment was conducted. **b** Estimated trajectory using visual odometry

Fig. 5c the mean-square-error of the estimation is illustrated, while Fig. 5d shows all discarded frames throughout the analysis of the sequence (1 stands for discarded frame, 0 stands for non-discarded frame). Sharp bends can be easily identified from observation of Fig. 5a, since they correspond to high values of the estimated yaw angle. Thus, from frame 0 to frame 75, approximately, the vehicle is turning to the left (negative yaw values), while from frame 800 to frame 1,000 the vehicle is turning sharply to the right. Similarly, the observation of Fig. 5b reveals that most of the times the number of outliers per frame remains low (below 6). Frames containing a high number of outliers (up to 13) are sporadic and isolated. This means that the feature extraction method is quite effective. The number of discarded frames in this experiment was 147, i.e., 12.25% of the total number of frames in the sequence. This can be considered a reasonable figure since the remaining frames still provide sufficient information for reliable position estimation. A remarkable point is the fact that discarded frames are rarely consecutive in time, allowing for robust interpolation using prediction from previous frames. System performance allows for algorithm execution at frame rate since the whole sequence (40 s of duration) was analyzed by the system in 37.56 s, including acquisition time.

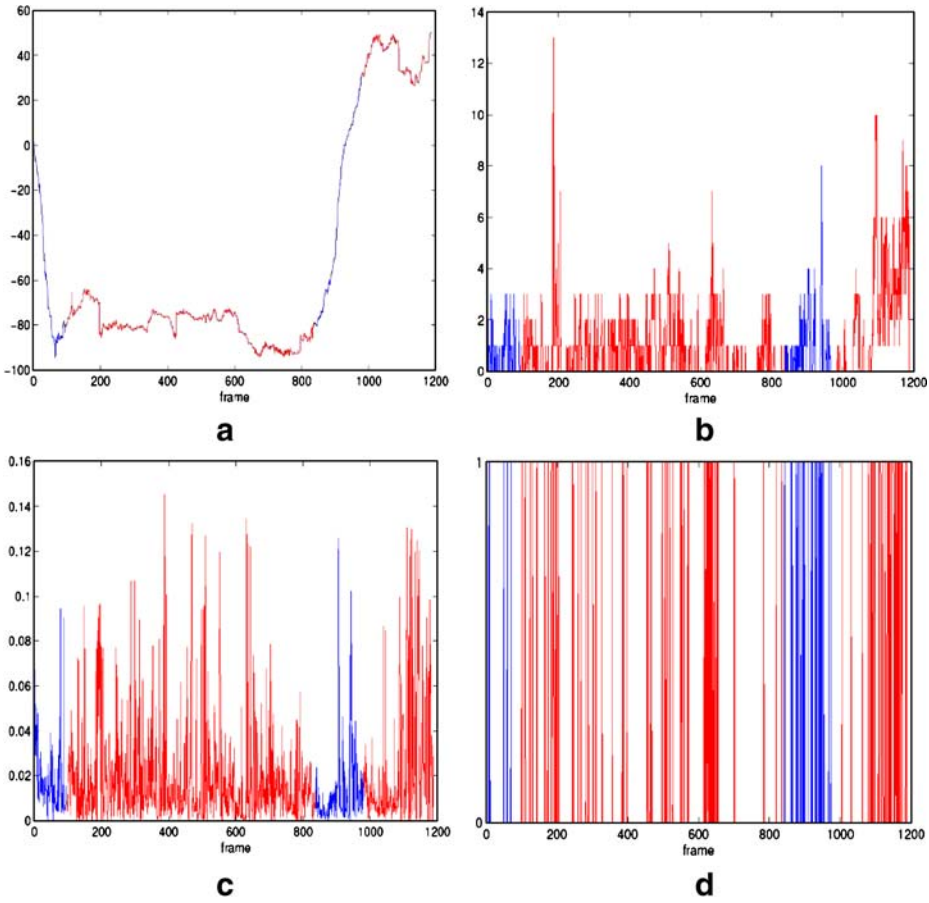

Fig. 5 **a** Cumulative estimated yaw rate. **b** Number of outliers per frame. **c** Mean square error. **d** Discarded frames after postprocessing

In a second experiment, the car started turning slightly right and then left to run along an almost straight path for a while. After that, a sharp right turn is executed. Then the vehicle moves straight for some metres until the end of the street. Figure 6 illustrates the real trajectory described by the vehicle (a) and the trajectory estimated by the visual odometry algorithm (b). In this case, the estimated trajectory reflects quite well the exact shape and length of the real trajectory executed by the vehicle. The system estimated a distance of 189.89 m in a real run of 194.33 m. As in the first experiment, Fig. 7 depicts the values of intermediate variables during the whole experiment. Figure 7a represents the cumulative estimated vehicle's yaw rate in a sequence of some 1,200 frames acquired at a frame rate of 30 frames/s (with a duration of 35.33 s). Figure 7b shows the number of outliers per frame. In Fig. 7c the mean-square-error of the estimation is illustrated, while Fig. 7d shows all discarded frames throughout the analysis of the sequence. Again, the observation of Fig. 7b reveals that most of the times the number of outliers per frame remains low (below 4). Frames containing a high number of outliers (up to 17) are again

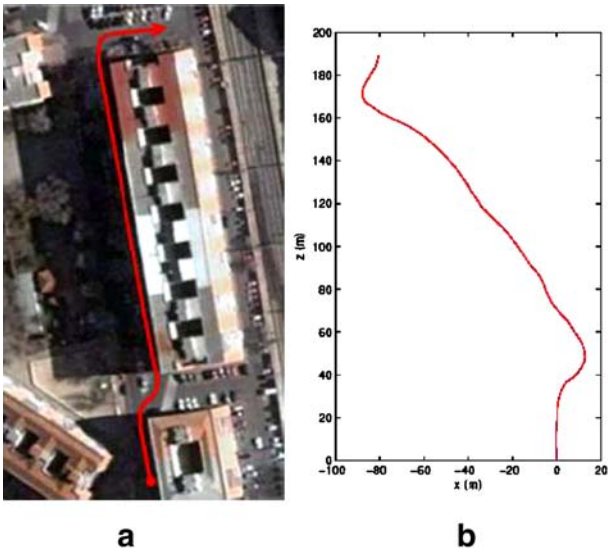

Fig. 6 **a** Aerial view of the area of the city were the experiment was conducted. **b** Estimated trajectory using visual odometry

sporadic and isolated. The number of discarded frames in this experiment was 18.14% of the total number of frames in the sequence. Once more, discarded frames are rarely consecutive in time, allowing for robust interpolation using prediction from previous frames. The whole sequence was analyzed by the system in 35.04 s including acquisition time. Algorithm execution at frame rate is then preserved.

4.2 3D Visual Odometry Results

The general 3D visual odometry method described in this paper was implemented and tested using the same road sequences recorded for 2D trajectory estimation. Most of the sequences were recorded during car runs on almost-planar roads in urban environments. A real experiment is graphically documented in this section as illustrated in Fig. 8. In the first part of the experiment, the car performs an almost straight trajectory along the street. Then, the driver rounds a corner right and resumes straight for a while. At a given moment, the trajectory turns slightly right in order to follow the street curvature. Finally, the car comes to a stop.

Figure 8a depicts the aerial view of the area of the city were the experiment was conducted. In Fig. 8b the simplified 2D estimated trajectory is shown. As can be observed, there exists great similarity in terms of shape between the estimated trajectory and the real one. Figure 8c illustrates the estimated 3D trajectory using the general method. The shape of the estimated 3D trajectory reflects quite well the real shape of the trajectory followed by the car in the experiment. The car ran a real distance of 276.79 m while the system estimated a distance of 272.52 m, which can considered as quite an accurate visual estimation. Nonetheless, the 3D visual odometry method yields an altitude change of 2 m during the car's run, which is not a realistic figure since the trajectory described by the vehicle is almost planar. The number of discarded frames in this experiment was 27.01% of the total number of frames in the sequence. Once more, discarded frames are rarely consecutive in

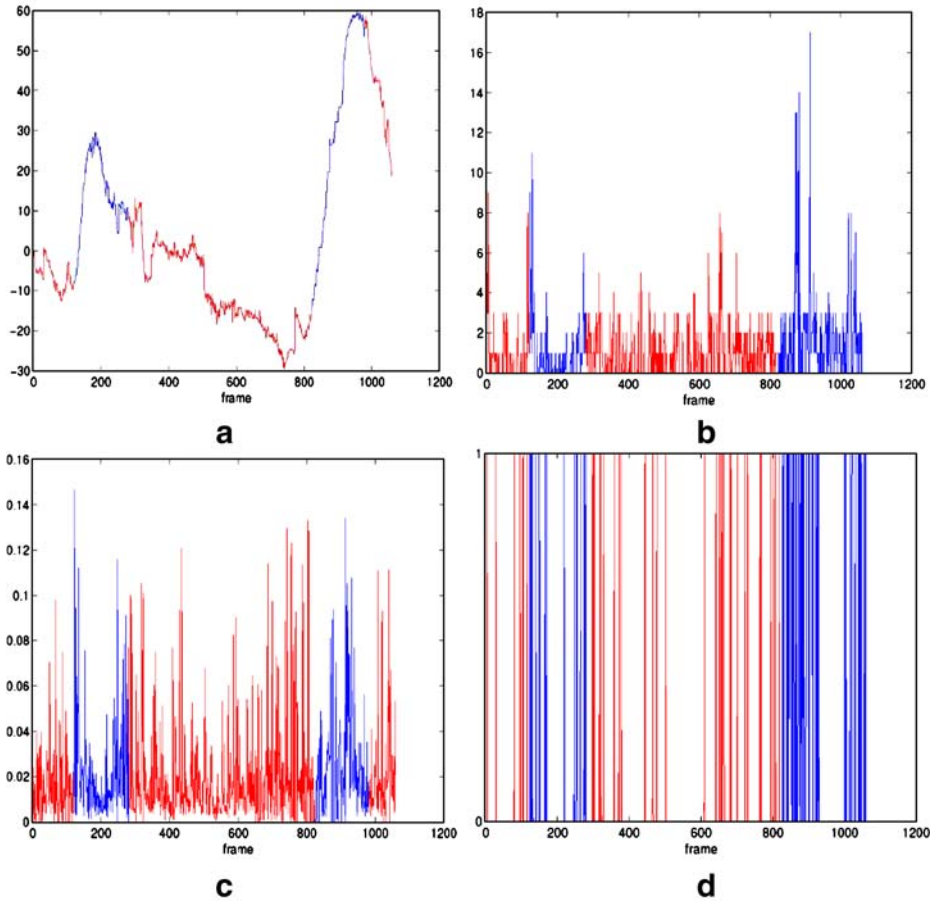

Fig. 7 **a** Cumulative estimated yaw rate. **b** Number of outliers per frame. **c** Mean square error. **d** Discarded frames after postprocessing

time, allowing for robust interpolation using prediction from previous frames. The whole sequence lasted 50.83 s and was analyzed by the system in 53.19 s including acquisition time. It can then be stated that algorithm execution at frame rate is practicely preserved also for the general 3D estimation case.

4.3 Discussion

After observation of the results provided in the previous section, it can be stated that the 3D visual odometry described in this paper provides approximate trajectory estimations that can be useful for enhancing GPS accuracy, or even for substituting GPS in short outage periods. Nonetheless, the system provides estimations that exhibit cumulative errors. Thus, it can not be realistically expected that a 3D visual odometry system be used as a standalone method for global positioning applications. Apart from this obvious fact, other problems arise especially in altitude estimation. The reason for this stems from the fact that estimations of pitch and roll angles

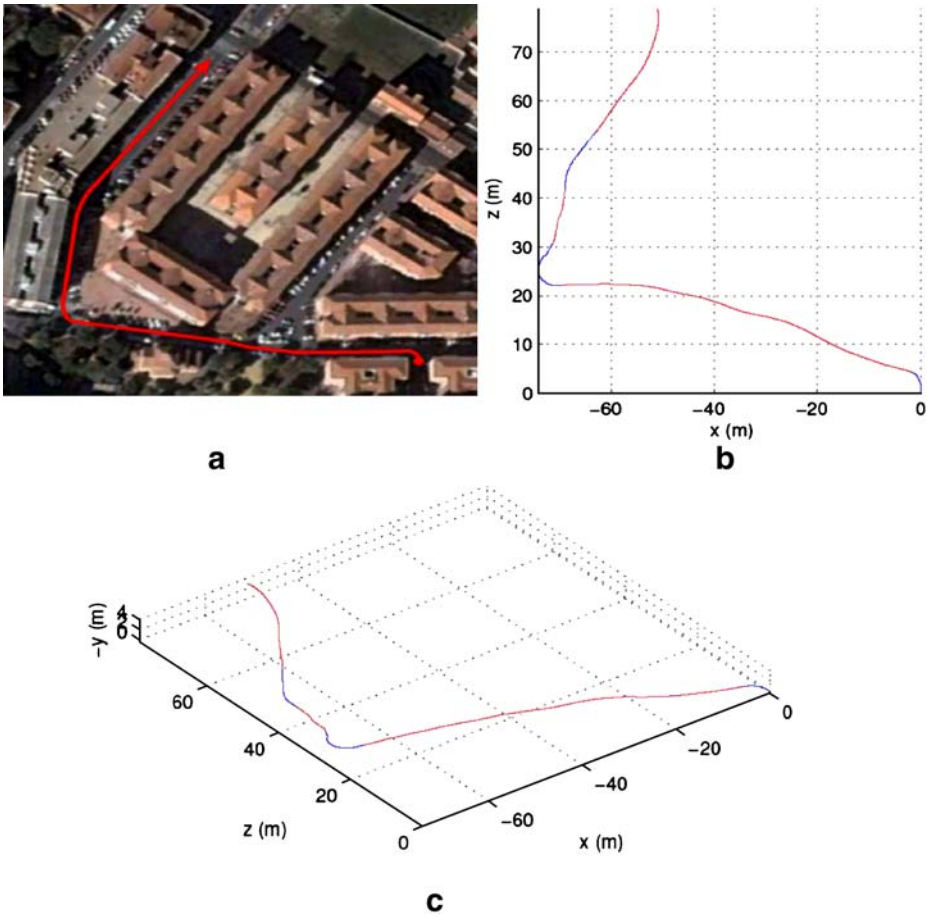

Fig. 8 **a** Aerial view of the area of the city were the experiment was conducted. **b** Estimated 2D trajectory. **c** Estimated 3D trajectory

become complex using visual means, since variations of these angles in usual car displacements are really small and difficult to measure in the 2D image plane. These difficulties produce a non-real altitude change in estimated 3D trajectories. Besides, the estimation of pitch and roll angles leads to a decrease in the accuracy of yaw angle estimation with regard to the 2D simplified method. As a consequence of that a greater error in estimated distance occurs. In addition, the 3D visual odometry method needs higher computational requirements to maintain performance at frame rate. Another problem arises when features corresponding to non-stationary objects are detected and used by the system. Non-stationary features lead to unrealistic motion estimation. This effect is observed with greater magnitude when the car is not moving. So, for instance, if the car is stopped at an intersection or a traffic signal, and other cars or pedestrians appear in the scene, the visual odometry method tends to produce unreal motion estimation in a direction that is contrary to the objects' movements. Though small, this is an upsetting effect that must be removed in future developments.

Finally, considering the possibility of a future commercial implementation of a visual odometry system for GPS enhancement, the simplified 2D estimation method described in this paper is a realistic, viable option that can help increase conventional GPS accuracy or even support GPS in short outage periods. Video sequences showing the results obtained in several experiments in urban environments can be anonymously retrieved from ftp://www.depeca.uah.es/pub/vision/visualodometry. The videos show a compound image in which the original input image and the estimated car trajectory image are synchronized and depicted together for illustrative purpose.

5 Conclusions and Future Work

We have described a method for estimating the vehicle global position in a network of roads by means of visual odometry. To do so, the ego-motion of the vehicle relative to the road is computed using a stereo-vision system mounted next to the rear view mirror of the car. Feature points are matched between pairs of frames and linked into 3D trajectories. The resolution of the equations of the system at each frame is carried out under the non-linear, photogrametric approach using least squares and RANSAC. This iterative technique enables the formulation of a robust method that can ignore large numbers of outliers as encountered in real traffic scenes. Fine grain outliers rejection methods have been experimented based on the root mean square error of the estimation and the vehicle dynamics. The resulting method is defined as visual odometry and can be used in conjunction with other sensors, such as GPS, to produce accurate estimates of the vehicle global position.

A key aspect of the system is the features selection method and tracking stage. For that purpose, a set of points has been extracted using Harris detector. The searching windows have been optimized in order to achieve a trade-off between robustness and execution time. Real experiments have been conducted in urban environments in real traffic conditions with no a priori knowledge of the vehicle movement nor the environment structure. We provide examples of estimated vehicle trajectories using the proposed method. Although preliminary, first results are encouraging since it has been demonstrated that the system is capable of providing approximate vehicle motion estimation in non-sharply bended trajectories. Nonetheless, further improvements need to be accomplished in order to accurately cope with 90° curves, which are very usual in urban environments.

As part of our future work we envision to develop a method for discriminating stationary points from those which are moving in the scene. Moving points can correspond to pedestrians or other vehicles circulating in the same area. Vehicle motion estimation will mainly rely on stationary points. The system can benefit from other vision-based applications currently under development and refinement in our lab, such as pedestrian detection [10] and ACC (based on vehicle detection). The output of these systems can guide the search for really stationary points in the 3D scene. The obvious application of the method is to provide on-board driver assistance in navigation tasks, or to provide a means for autonomously navigating a vehicle. For this purpose, fusion of GPS and vision data will be accomplished.

Acknowledgements This work has been supported by the Spanish Ministry of Education and Science by means of Research Grant DPI2005-07980-C03-02 and the Regional Government of Madrid by means of Research Grant CCG06-UAH/DPI-0411.

References

1. Zhang, Z., Faugeras, O.D.: Estimation of displacements from two 3-D frames obtained from stereo. IEEE Trans. Pattern Anal. Mach. Intell. **14**(12), 1141–1156, December (1992)
2. Nister, D., Narodistsky, O., Beren, J.: Visual odometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June (2004)
3. Hagnelius, A.: Visual odometry. In: Masters Thesis in Computing Science, Umea University, April (2005)
4. Forsyth, D.A., Ponce, J.: Computer Vision. A Modern Approach (international edn.). Pearson Education International. Prentice Hall (2003)
5. Agrawal, M., Konolige, K.: Real-time localization in outdoor environments using stereo vision and inexpensive gps. In: 18th International Conference on Pattern Recognition (ICPR06), pp. 1063–1068 (2006)
6. Simond, N., Parent, M.: Free space in front of an autonomous guided vehicle in inner-city conditions, In: European Computer Aided Systems Theory Conference (Eurocast 2007), pp. 362–363 (2007)
7. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the Fourth Alvey Vision Conference, pp. 147–151 (1988)
8. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. Int. J. Comput. Vis. **37**(2), 151–172 (2000)
9. Boufama, B.: Reconstruction tridimensionnelle en vision par ordinateur: cas des cameras non etalonnees. Ph.D. thesis, INP de Grenoble, France (1994)
10. Parra, I., Fernández, D., Sotelo, M.A., Bergasa, L.M., Revenga, P., Nuevo, J., Ocana, M., García, M.A.: Combination of feature extraction methods for SVM pedestrian detection. IEEE Trans. Intell. Transp. Syst. **8**(2), 292–307, June (2007)
11. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM. **24**(6), 381–395, June (1981)
12. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2004)