A temporally consistent grid-based visual odometry framework for multi-core architectures

Leonardo De-Maeztu, Unai Elordi, Marcos Nieto, Javier Barandiaran & Oihana Otaegui

Journal of Real-Time Image Processing

ISSN 1861-8200

J Real-Time Image Proc
DOI 10.1007/s11554-014-0425-y

Springer

J Real-Time Image Proc
DOI 10.1007/s11554-014-0425-y

SPECIAL ISSUE PAPER

A temporally consistent grid-based visual odometry framework for multi-core architectures

Leonardo De-Maeztu · Unai Elordi · Marcos Nieto · Javier Barandiaran · Oihana Otaegui

Received: 13 September 2013 / Accepted: 15 April 2014

Abstract Most recent visual odometry algorithms based on sparse feature matching are computationally efficient methods that can be executed in real time on desktop computers. However, further efforts are required to reduce computational complexity in order to integrate these solutions in embedded platforms with low power consumption. This paper presents a spacetime framework that can be applied to most stereo visual odometry algorithms greatly reducing their computational complexity. Moreover, it enables exploiting multi-core architectures available in most modern computing platforms. According to the tests performed on publicly available datasets and an experimental driverless car, the proposed framework significantly reduces the computational complexity of a visual odometry algorithm while improving the accuracy of the results.

Keywords Visual odometry · Grid structure · Multi-core

1 Introduction

In the past few years, there has been a growing interest in advanced driver assistance systems (ADAS) and autonomous vehicles [2, 18, 19]. In general, this interest is closely related to the sustainable, innovative and safe transport systems aspect of smart mobility inside smart cities (see the European cities report [10]). Intelligent vehicles increase fuel efficiency thanks to environmentally friendly driving and reduced traffic congestion. Additionally, considering that human factors (alone or combined with other causes) are involved in the vast majority of accidents [25], new technological contributions are expected to reduce the number of accidents thanks to more predictable behaviors of vehicles and faster response times.

A key component of an autonomous car is the positioning module, used to compute the localization of the vehicle and eventually to plan future motion. Different types of sensors can be used to compute the position of the car in real time, such as global navigation satellite systems (GNSS), inertial navigation systems (INS), laser scanners or video cameras [8].

Computer vision using cameras bring cheap and robust localization possibilities that are usually classified as visual odometry (VO) [26] or visual simultaneous localization and mapping (V-SLAM) algorithms [13]. The main difference between both types of solutions is that, while VO methods compute motion incrementally (frame after frame) [9], V-SLAM and other similar solutions optimize the global consistency of the path [5, 15]. For this purpose, a reconstruction of the path is needed to detect the particular situation where the car visits the same place twice (loop closure condition, used to enforce global consistency). From this description, it is evident that V-SLAM solutions are in general more complex but also more accurate than VO methods. However, global reasoning techniques such as loop closure detection can severely affect results in case of failure [26].

Given the relatively low computational complexity of most VO methods and the implementation on desktop computers, little attention has been paid to further reducing complexity by exploiting available architectural resources or redundant information. However, the integration of VO algorithms in embedded platforms that may execute many driving assistance algorithms requires efficient solutions.

L. De-Maeztu (✉) · U. Elordi · M. Nieto · J. Barandiaran · O. Otaegui
Vicomtech-IK4, Paseo Mikeletegi 57, Donostia-San Sebastián, Spain
e-mail: leonardo.demaeztu@gmail.com

Published online: 01 May 2014

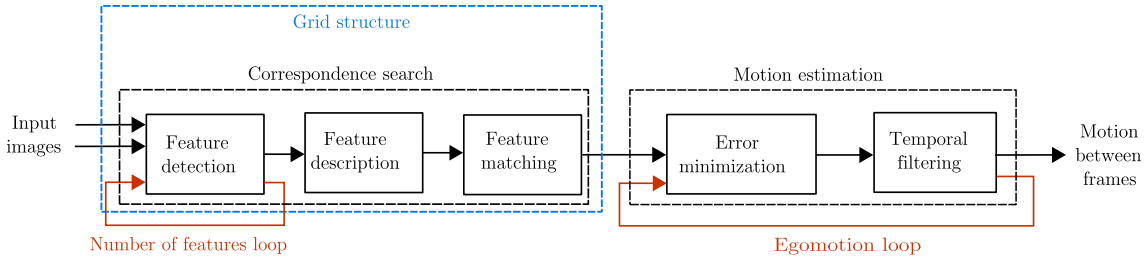

Fig. 1 Execution pipeline implemented by most VO algorithms (in *black*) along with the proposed spatial (in *blue*) and temporal (in *red*) modifications

Only some algorithms, for which parallel architectures can be used to notably increase speed, have been proposed along with parallel implementations [31]. In this particular case, for example, VO computation is based on dense information. GPU architectures are very well suited for this type of dense, pixel-based processing, because they contain many processing cores (up to several thousands). As a consequence, each thread can be allocated for the processing of each pixel.

This paper proposes a novel framework in which most stereo (i.e., using two cameras) VO solutions can be casted. This framework exploits the temporal redundancy of a video sequence and the spatial distribution of features or keypoints over images. Temporal redundancy is used to stabilize the number of detected features over time and to initialize the egomotion computation for the current frame using the motion computed for the previous frame. The spatial distribution of features over the input images is used to partition them into a regular grid so that features can be detected and described independently for each cell of the grid, enabling a parallel implementation in multicore architectures. Moreover, the grid can be used to build a mask for feature matching in an efficient manner. In Fig. 1 the common pipeline of most VO algorithms is summarized (most VO solutions compute egomotion after a stage in which keypoints are detected, described and matched between frames) along with the proposed modifications. The effectiveness of the framework is demonstrated by applying it to a relatively standard VO algorithm implemented in a robotic car in the framework of the Taxisat FP7 project [23] that aims to develop a driverless car that can operate autonomously following a predefined path and stopping if obstacles are detected (see Fig. 2).

The remainder of this paper is organized as follows. Section 2 presents the state of the art of VO algorithms. In Sect. 3 the proposed approach is introduced. Experimental results are analyzed in Sect. 4. We close the paper with a short conclusion and an outlook on future work in Sect. 5.

Fig. 2 Taxisat driverless car

2 Related work

VO is a particular case of structure from motion (SfM) [12, 30]. In general, VO assumes that the sequence of images was acquired with a single array of cameras capturing the images while moving though space. In most cases, the camera array is composed of one (monocular) or two (stereo) cameras. VO algorithms compute the path by incrementally estimating motion over temporally consecutive frames [26].

Two steps can be clearly distinguished in most stereo VO algorithms (see Fig. 1):

1. *Correspondence search between frames*. Identify corresponding points between consecutive frames.
2. *Motion estimation*. Given the computed correspondences between points in two frames taken by the same camera in different instants, motion between both positions of cameras can be computed.

2.1 Correspondence search

According to the authors of [26], feature-based methods are in general preferred over other solutions that use the intensity information of all pixels in the input images (appearance-based methods) [20, 28]. In [20], rotation is extracted from the intensity profile of a column intensity graph. Then speeds are estimated based on the rate of image change. In [28], feature-based tracking is used to obtain a first estimation of the motion of the car. Then, an appearance-based approach is deployed that improves the accuracy of the computed rotation of the vehicle.

Feature-based solutions select a set of keypoints to compute motion. A wide variety of methods have been used to select reliable keypoints (feature detection) and to compare and associate keypoints corresponding to different images (feature description and matching). Harris corner detector and SURF are implemented in [14] with similar results. In [9], two types of custom masks designed to detect blobs and corners, respectively, are used.

Some publications such as [16, 27] implement different feature detection techniques to compare their results. After evaluating detectors on different scenarios, the authors of [16] chose CenSurE [1]. On the other hand, after comparing a different set of detectors and trackers, the authors of [27] concluded that the results were not due to the accuracy of the detector, but rather to the distribution of features in the images. According to their results, a uniform distribution of features results in better motion estimation. Different techniques can be employed to force a homogeneous distribution of the computed keypoints. One of the simplest methods is to partition the input images into a regular grid and force a uniform distribution of the total number of features over the cells of the grid [21]. In [14], a similar grid is used to discard keypoints in cells of the grid that contain too many candidates compared to other regions. The use of grid or pyramid-based structures led to successful implementations in areas such as image categorization [11, 17] and inspired the extension of this type of techniques to other areas.

2.2 Motion estimation

Once the feature correspondences between frames have been established, motion between these frames can be computed. Most VO algorithms include an optimization strategy (to minimize the reprojection error of the matching results according to the cameras parameters and the computed motion) and a method to increase robustness against outliers.

Minimization can be implemented as a 2D-to-2D, 3D-to-2D or 3D-to-3D point registration method. 3D-to-2D registration is the most implemented method. In the monocular case because it enables faster data association [26], and in the binocular case because it provides more accurate results [22]. Despite these advantages, some authors have proposed algorithms that avoid explicit 3D point computation using triangulation even in multicamera setups. This is the case of [4], where quadrifocal tensors are used for motion estimation. Nevertheless, 3D-to-2D registration is the most used technique in systems with multiple cameras. Optimization of the reprojection error is generally performed using minimization techniques such as Gauss–Newton [9] or Levenberg–Marquardt algorithm [28, 29].

To increase robustness against outliers, most methods include an implementation of the RANSAC algorithm [9, 14, 16]. Also, enhanced smoothness can be obtained using dynamic models of the vehicle along with Kalman filtering [9, 14].

3 Proposed method

Our stereo VO framework consists of an adaptive grid-based correspondence search stage and a temporal consistent motion estimation stage.

3.1 Adaptive grid-based correspondence search

The correspondence search step implements the proposed adaptive grid-based strategy and it includes feature detection, description and matching (see Fig. 1). The input images are divided in grids of H row and W column cells. Unlike previous grid-based solutions that only use the grid structure in the feature detection step in order to obtain a homogeneous distribution of feature keypoints [14, 21], our proposal consistently uses this grid through feature detection, description and matching steps. Thus, not only the positive effects over motion accuracy of a homogeneous distribution of keypoints can be demonstrated; also, feature detection and description can be parallelized thanks to the grid structure. Moreover, this structure can also be used to avoid comparing keypoints that belong to distant cells during the feature-matching step, as it will be shown later. Efficient methods were selected for feature detection and description, because in VO for ground vehicles corresponding points are similar in consecutive frames and thus complex requirements such as rotation or scale invariance are not necessary [9]. According to our experiments, the implementation of robust methods against rotation or scale changes does not result in better motion estimation accuracy.

First, features are detected in each image of the input stereo pair. In this paper, we decided to use features from accelerated segment test (FAST) [24] due to its

computational simplicity and easy parametrization. To obtain a homogeneous distribution of features, FAST is applied individually to the cells of a grid-based representation of the input stereo pair. As opposed to an implementation with a fixed threshold value t, we propose the implementation of an adaptive FAST threshold to generate the desired number of features in each cell (i.e., the desired number of features N for the image divided by the number of cells). Using a fixed value for the FAST threshold implies obtaining a variable number of features for each frame that may be insufficient to accurately compute motion or excessive (in this case the excess keypoints are simply removed; however, computing these unused keypoints increases computational complexity). Moreover, an adaptive FAST threshold favors that the same points are detected in consecutive frames under illumination or camera parameters changes. For this reason, and considering the high similarity of consecutive frames of a video sequence, we propose to update the FAST threshold for each frame t_k using the threshold of the previous frame t_{k-1} and the number of detected points M in this same image according to the following rule:

$$t_k = \begin{cases} t_{k-1} - \Delta t, & \text{if} \quad \mathrm{M} < \mathrm{N}. \\ t_{k-1} + \Delta t, & \text{otherwise}. \end{cases} \tag{1}$$

Given that the computation of the keypoints for the left and the right images of the stereo pair is independent, two threads can be used to compute the keypoints of each image. Moreover, thanks to the proposed grid structure, different threads can be used to compute the features inside each cell.

Next, features are described. Feature description can also be easily parallelized by using different threads for each image of the input stereo pair and for each cell of the grid. In this particular implementation, also for simplicity purposes, we decided to use binary robust independent elementary features (BRIEF) [3]. The BRIEF descriptor computes for each keypoint a n_b bit string.

Finally, the keypoints are compared and matched. In this case, we use the Hamming distance of the BRIEF descriptors (a very efficient operation in CPUs). Two types of matching are performed in this step:

1. *Spatial matching or stereo matching*. Features captured by both cameras at the same instant are matched to enable 3D triangulation from the 2D coordinates of these features.
2. *Temporal matching or optical flow computation*. Features captured by the same camera in consecutive frames are matched to enable motion computation.

In order to limit the computational complexity, a priori constraints can be imposed to limit the number of potential matches. For stereo matching, we know that in a rectified stereo pair, correspondences have to lie on the same epipolar horizontal line of the image. Moreover, an interval of interest disparities can be established limiting the search range between a minimum and a maximum disparity. For optical flow computation, correspondence search has to be performed in all directions, but a maximum distance between candidates can be established if we know the maximum motion of the camera between frames. To compute the minimum and maximum disparity and the maximum distance between candidates, camera geometry concepts have to be used. For stereo matching it is relatively straightforward, because the disparity is only related to the focal length of the cameras, the baseline and the distance of points [6]. However, for optical flow computation the maximum rotation and translation of the cameras between consecutive frames has to be considered, along with the projection matrices of the cameras [12].

Imposing these disparity and distance constraints before matching would mean that the Euclidean distance of every pair of features has to be computed. Unfortunately, computing Euclidean distances in order to avoid computing unnecessary Hamming distances (which is a less complex operation on today CPUs) is a nonsense. To overcome this limitation, we propose an efficient cell-based solution using the grid deployed for feature detection and description. In this way, we can mask keypoints that belong to distant cells using a block-based fast strategy.

For stereo matching, given the maximum disparity $d_{\max}$ and the cell width W, the cells of the right image that contain candidate matches for a certain cell of the left image range from the same cell of the left image to $\lceil d_{\max}/W \rceil$ cells to the left. Figure 3 contains a graphical explanation. In this particular example, $\lceil d_{\max}/W = 2 \rceil$, so the candidate cells include the cell occupying the same position as the one in the left image and its two neighbors to the left.

A similar procedure is applied for selecting the candidate cells for optical flow matching. In this case, the vector that joins two correspondences is two dimensional with a maximum length of $D_{\max}$ pixels. Cells dimensions are $W \times H$ pixels. In general, the search cells in the right image

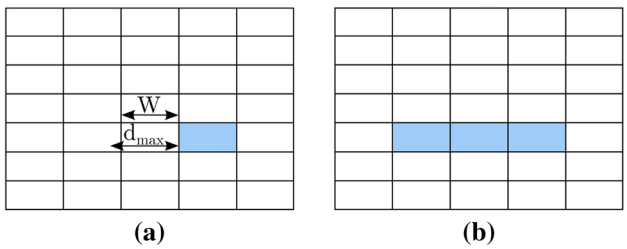

Fig. 3 Grid-based a priori mask computation for stereo matching. The left image keypoints belonging to the cell highlighted in **a** can only be matched to the keypoints belonging to the cells of the right image highlighted in **b**

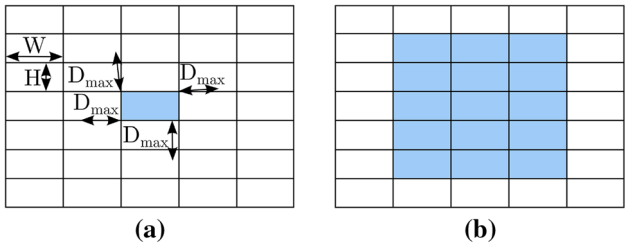

Fig. 4 Grid-based a priori mask computation for optical flow. The previous frame keypoints belonging to the cell highlighted in **a** can only be matched to the keypoints belonging to the cells of the current image highlighted in **b**

include a rectangle of $2 \times \lceil D_{\max}/H \rceil + 1$ row cells and $2 \times \lceil D_{\max}/W \rceil + 1$ column cells centered in the cell in the same position to the one of the reference image. Figure 4 contains a graphical representation of a situation where $\lceil D_{\max}/W \rceil = 1$ and $\lceil D_{\max}/H \rceil = 2$, so the search region is a 5×3 rectangle of cells. Of course, potential search cells that lie outside the images are not considered.

After matching, false matches can be eliminated checking that stereo matches lie on the same horizontal line, that are separated by a distance smaller than $d_{\max}$ and that optical flow matches are separated by a distance smaller than $D_{\max}$. Grid-based masking avoids most of these erroneous correspondences but not all. Then, remaining false matches can be removed using the circular match strategy [9]. This consistency check strategy verifies that a complete loop of temporal/spatial matches finishes in the departure point. Only matches that verify the circular match condition are used for motion computation.

3.2 Temporal consistent motion estimation

Using the valid matches previously obtained (lets suppose L valid circular matches), we compute the camera motion by minimizing the sum of reprojection errors. In particular, we use Levenberg–Marquardt minimization to find the rotation and translation vectors that best adapt to the corresponding 3D points in space of the previous stereo pair of frames $\mathbf{X_{k-1,i}}$ and the 2D coordinates of the features detected in the current stereo pair $\mathbf{x_{k,i}}$. If we define $f^{l}_{\mathbf{r,t}}$ to be the function that projects 3D points to 2D points in left camera considering that is has moved according to a rotation vector $\mathbf{r}$ and a translation vector $\mathbf{t}$, and similarly $f^{r}_{\mathbf{r,t}}$ for the right camera, the cost function to minimize is:

$$\sum_{i=1}^{L} \| \mathbf{x^{l}_{k,i}} - \mathbf{f^{l}_{r,t}}(\mathbf{X^{l}_{k-1,i}}) \| + \| \mathbf{x^{r}_{k,i}} - \mathbf{f^{r}_{r,t}}(\mathbf{X^{r}_{k-1,i}}) \| . \quad (2)$$

To increase robustness against outliers, this minimization procedure is integrated in a RANSAC scheme performing I iterations (less iterations are needed if the number of RANSAC inliers exceeds a certain percentage p of the total number of pixels used to compute motion). Finally, Kalman filtering is applied to the translation and rotation vectors to produce a statistically optimal estimate of egomotion [9]. Our main contribution to motion estimation is related to the observation that the vehicle motion is relatively smooth. As a consequence, we expect the translation and rotation vector computed for consecutive frames to be similar. For this reason, we propose to use the Kalman filter prediction (temporal filtering) as an initialization value for the Levenberg–Marquardt/RANSAC procedure (error minimization) in each iteration as shown in Fig. 1 to reduce the number of iterations needed for convergence and to improve the accuracy of the result thanks to a better departure point for optimization.

To improve the reproducibility of the proposed algorithm, we summarize in Table 1 all the parameters involved.

4 Experimental results

To test the performance of the proposed VO algorithm, two different benchmarking scenarios were designed. First, the

Table 1 List of parameters of the proposed VO algorithm

Correspondence search	Grid size: $W \times H$	
	Feature detection (FAST)	Threshold: t (fixed value) or Δt step (adaptive)
		Target number of detected points: N
	Feature description (BRIEF)	Number of bits: n_b
	Feature matching	Maximum distance: $D_{\max}$
		Maximum disparity: $d_{\max}$
Egomotion estimation	Levenberg–Marquardt/RANSAC	Number of iterations: I
		Minimum percentage of inliers: p
	Kalman filter	Covariance 6×6 matrix of the translation process noise: R_t
		Covariance 3×3 matrix of the translation observation noise: B_t
		Covariance 6×6 matrix of the rotation process noise: R_r
		Covariance 3×3 matrix of the rotation observation noise: B_r

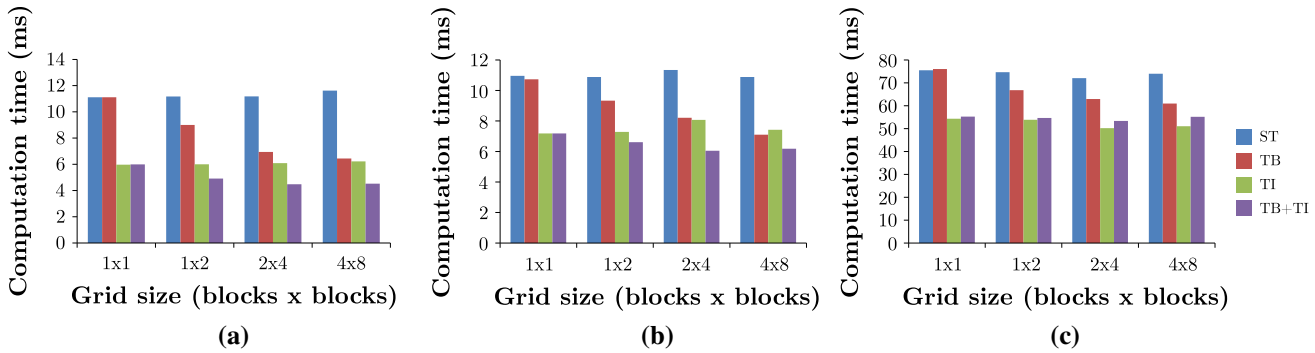

Fig. 5 Feature detection and description execution time: **a** Intel Core i5-3330, **b** Intel Core i7-2640M and **c** Intel Atom N270

improvement due to each of the proposed ideas was evaluated using three different laboratory CPUs and the KITTI odometry evaluation dataset with ground truth [8]. Then, we tested the performance of the final VO solution on the Taxisat vehicle.

4.1 Dataset with ground truth

The KITTI Vision Benchmark Suite [8] consists of six individual benchmarks designed to evaluate the performance of different algorithms that are generally integrated in ADAS or autonomous vehicles. One of the six benchmarks is designed to test VO and V-SLAM algorithms. It contains 22 sequences of images recorded with a stereo pair of cameras embedded in a car. A ground truth and an evaluation methodology are provided for the first 11 sequences. The other 11 sequences are provided without this type of information and the results have to be uploaded to the KITTI server to obtain an evaluation of the accuracy of the results along with the position in a ranking of VO/V-SLAM algorithms. As a consequence, the first 11 sequences are useful for optimal parameter setting and intensive testing of different configurations as done in this paper. In order to evaluate the scalability of the proposed algorithm over different computing architectures, three different CPUs were used for the laboratory tests: Intel Core i5-3330 (four cores, four threads, TDP 77 W), Intel Core i7-2640M (two cores, four threads, TDP 35 W) and Intel Atom N270 (one core, two threads, TDP 2.5 W). TDP stands for thermal design power. When the number of threads exceeds the number of cores, it means that Intel Hyper-Threading Technology is implemented in this CPU to enable that each core handles more than one thread. Qt[1] and TBB[2] were integrated in the implementation of the algorithm to enable parallel computing using multiple threads along with OpenCV[3] that already implements some of the methods described in this paper.

In this section, not only the performance of the proposed spatiotemporal solution is tested. On the contrary, the effect of both spatial and temporal modifications on accuracy and computational complexity is analyzed separately to distinguish their contribution to the performance of the final solution. With this object, first the performance of a raw VO algorithm without the proposed modifications is summarized in Table 4 ('Raw'). The parameter configuration used is $W = 1 \times H = 1$ (no grid), $t = 10$, $N = 500$, $n_b = 256$, $D_{\max} = 200$, $d_{\max} = 150$, $I = 50$, $p = 85\,\%$, $R_t = 10^{-4} \times I$, $B_t = 10^{-3} \times I$, $R_r = 10^{-3} \times I$ and $B_r = 10^{-4} \times I$.

4.1.1 *Grid-based correspondence search*

As it was previously explained and as it is depicted in Fig. 1, the grid structure is integrated in feature detection, description and matching stages. The first benefit is the computational complexity reduction of the tasks that can be parallelized. Figure 5 depicts the execution time of feature detection and description for different parallelism configurations of the algorithm: without any level of parallelization (ST), with two threads performing in parallel the same task for the left and right images (TI), and finally with a different thread processing each block of the grid (TB). The analysis is repeated for different grid distributions and for the three tested CPUs. The use of two threads for separate processing of the left and right image (TI) clearly speeds up the algorithm because both processes are completely independent. The gain is independent of the grid size. The grid-based implementation (TB) also enables a speedup of the algorithm thanks to multithreading processing of the cells of the grid. In this case, the gain depends on the grid size. For large grid sizes the speedup

[1] http://qt-project.org/.

[2] http://www.threadingbuildingblocks.org/.

[3] http://opencv.org/.

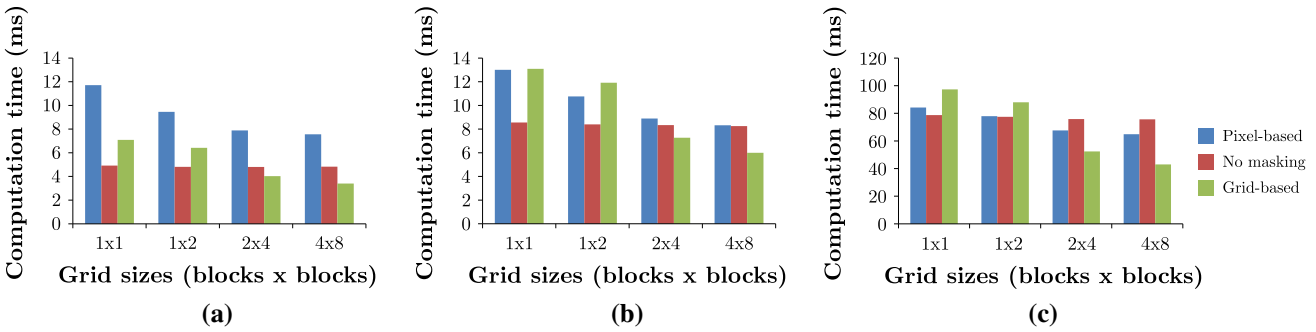

Fig. 6 Feature matching execution time: **a** Intel Core i5-3330, **b** Intel Core i7-2640M and **c** Intel Atom N270

can be similar to the TI speedup. When both types of parallelization are combined (TB+TI) computation time is further reduced, except in the case of the Intel Atom CPU. This processor only contains one core and two threads can be executed simultaneously. As a consequence, the combination of two parallelization techniques (TB+TI) that results in at least four threads is not effective on the Intel Atom processor. To give an idea of the parallelization efficiency, when using a 8×4 grid, the single thread 11.62 ms execution time on the Intel Core i5-3330 processor can be reduced to 6.22 ms using TI (46 % speedup) and to 4.52 ms (61 % speedup) using TB+TI. In the case of the Intel Core i7-2640M processing unit, the single thread execution time of 10.88 ms is reduced to 7.42 ms when integrating TI (32 % speedup) and to 6.18 ms (43 % speedup) when combining TB and TI. Finally, when testing the Intel Atom N270 platform, the 73.97 ms single thread execution time can be reduced to 51.06 ms (31 % speedup) if TI is used. However, computation time cannot be further reduced using TB due to the limited number of threads.

The second benefit is better motion estimation accuracy, thanks to a more homogeneous distribution of feature points. To verify the influence of the distribution of the feature points in the odometry results, we tested the average error on the KITTI first 11 sequences using different grid configurations (the number of columns being always equal or larger than the number of rows because the horizontal resolution is larger than the vertical resolution).

Unlike the results for underwater navigation [21], grid-based feature detection and description seems to improve the accuracy of the estimated path for a subset of grid configurations according to Tables 2 and 3. Large grid configurations such as 4×2 or 8×4 that force a homogeneous distribution of features in both horizontal and vertical directions are beneficial for motion estimation accuracy. These results agree with those presented in [27], where it is concluded that for on-road vehicles (in the mentioned case integrating only one camera) a uniform distribution of feature keypoints favors egomotion estimation accuracy.

Even if the improvement is small in most cases (a 7 % improvement in rotation error and a 11 % improvement in translation error), it is always associated with faster execution times (see Fig. 5).

Finally, the third benefit of the grid structure is a faster execution even in non-parallel architectures, thanks to the speedup of feature matching. As previously explained, the use of grid-based masking was also expected to reduce the computation time of feature matching. Grid-based masking is a computationally efficient way of limiting the number of feature points to be compared. Because computing the Euclidean distance between all potential candidates for

Table 2 Translation error (%) of the VO algorithm versus the number of rows (left) and columns (up) of the grid structure

	1	2	4	8
1	2.36	2.36	2.43	2.40
2	×	2.46	2.19	2.64
4	×	×	2.54	2.20

Table 3 Rotation error (°/m) of the VO algorithm versus the number of rows (left) and columns (up) of the grid structure

	1	2	4	8
1	0.0142	0.0132	0.0136	0.0127
2	×	0.0135	0.0128	0.0131
4	×	×	0.0132	0.0126

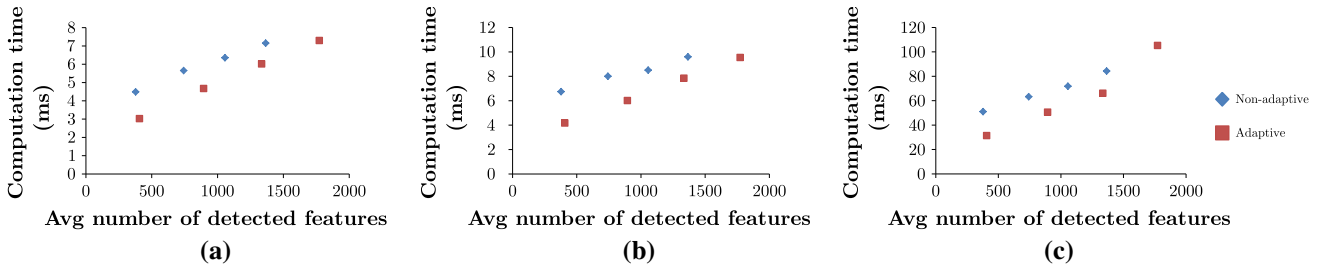

Fig. 7 Comparison of the computation time versus the average number of features when using a fixed value and a variable value for the FAST feature detector threshold: **a** Intel Core i5-3330, **b** Intel Core i7-2640M and **c** Intel Atom N270

Table 4 Accuracy and computation time of the proposed modifications to the raw VO algorithm

Method	CPU	Runtime				Error	
		Detection and description (ms)	Matching (ms)	Egomotion (ms)	Total (ms)	Translation (%)	Rotation (°/m)
Raw	i5	11.32	5.09	1.95	18.36		
	i7	10.96	8.72	2.74	22.42	2.36	0.0142
	Atom	74.68	77.85	24.40	176.93		
Raw+grid	i5	4.52	3.40	2.91	10.83		
	i7	6.18	6.00	5.38	17.56	2.20	0.0126
	Atom	51.06	43.04	46.73	140.83		
Raw+grid+stable	i5	3.43	4.34	2.71	10.48		
	i7	4.26	7.72	4.72	16.70	2.29	0.0124
	Atom	33.26	52.95	38.70	124.91		
Proposed	i5	3.48	4.35	1.02	8.85		
	i7	4.46	8.06	1.56	14.08	2.09	0.0122
	Atom	33.04	52.21	10.09	95.34		

matching has a high computational cost, we can take advantage of the grid partitioning of the images in our algorithm to avoid comparing pixels that are located in distant cells.

If Fig. 6, the computation time of feature point matching is plotted when comparing all possible candidates, or when masking the possible matches using pixel-based Euclidean distance computation or the proposed grid-based masking. Euclidean distance masking increases the computation time because the distance between all the possible candidates is computed. Compared to pixel-based masking, grid-based masking is much more computationally efficient because it compares buckets of pixels. Compared to the no masking case, a priori masking of impossible matches (very distant pixels in optical flow or pixels on different horizontal lines in stereo matching) is faster when using large grids; and in addition, it makes possible to keep a higher number of matches for motion computation. If these matches are not prohibited before matching, they will have to be discarded after, loosing potential matches for motion computation. According to the results presented in Fig. 6, for a 8 × 4 grid configuration, grid-based masking produces a 30, a 27 and a 43 % speedup in the Intel Core i5-3330, Intel Core i7-2640M and Intel Atom N270 implementations, respectively. As expected, the speedup is not related to the parallel architecture of the processors. On the contrary, the speedup is higher for the single-core processor.

According to the previous experiments, we selected the 8 × 4 grid configuration due to the balance between accuracy improvement and execution speedup. In Table 4, the results of the raw algorithm incorporating grid-based feature detection, description and matching are presented ('Raw+grid'). The same parameter values than in the raw solution without grid are used except for the 8 × 4 grid instead of the 1 × 1 configuration.

4.1.2 Temporal consistency

The other major proposed modification is the integration of a temporal mechanism to obtain a stable number of feature points over time and to use the computed motion of the last

pair of frames as a departure point for minimization in the next pair of frames.

In order to stabilize the number of detected features over time and to further speedup the feature detection process, an adaptive threshold for the grid-based FAST detector was proposed. In Fig. 7, the average number of detected features over all sequences is represented versus the computation time for feature detection and description when using an adaptive and a fixed FAST threshold. $t = 10$ is used in the fixed threshold configuration and $\Delta t = 1$ is used in the adaptive threshold configuration. According to our experiments, larger Δt values do not speedup the algorithm and result in reduced accuracy. According to Fig. 7, the speedup is more important when computing a small set of feature points (systems with a limited computational power). For example, for the selected configuration ($N = 500$), feature detection and description is 24 % faster for the adaptive solution on the Intel Core i5 CPU, 31 % on the Intel Core i7 processor and 35 % on the Intel Atom platform.

Finally, the proposed motion estimation loop reduces the convergence time of the egomotion estimation stage (essentially Levenberg–Marquardt/RANSAC, since the Kalman filter computation time is negligible) as can be observed in Fig. 8. 62, 67 and 74 % speedups are obtained for the Intel Core i5, Intel Core i7 and Intel Atom implementations, respectively.

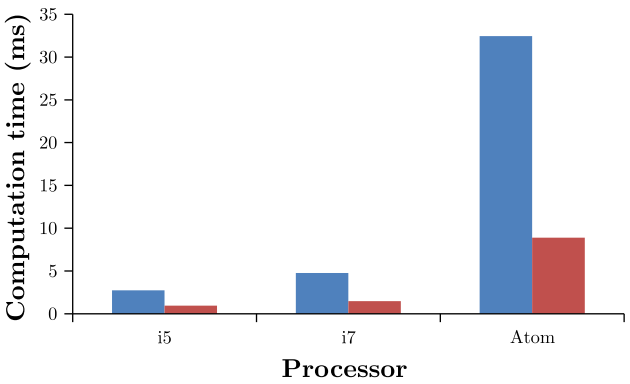

Fig. 8 Egomotion loop speedup. In *blue*, computation time of the egomotion estimation stage without the proposed egomotion loop. In *red*, the same computation time with the proposed egomotion loop

To summarize the performance of the proposed solution and compare it to a solution without the proposed optimizations, Table 4 shows its accuracy and execution time. The 'Raw+grid+stable' entry adds to the 'Raw+grid' configuration the stabilization of the number of detected features over time. Finally, the 'Proposed' entry refers to the proposed algorithm containing all the mentioned modifications. The parameter configuration used for both entries of the table is $W = 8 \times H = 4$, $\Delta t = 1$, $N = 500$, $n_b = 256$, $D_{\max} = 200$, $d_{\max} = 150$, $I = 50$, $p = 85\,\%$, $R_t = 10^{-4} \times I$, $B_t = 10^{-3} \times I$, $R_r = 10^{-3} \times I$ and $B_r = 10^{-4} \times I$. According to this table, not only a 52 % (Intel Core i5-3330), 37 % (Intel Core i7-2640M) and 46 % (Intel Atom N270) speedup is obtained considering the total computation time. Moreover, better egomotion estimation accuracy is attained due to a more uniform distribution of features over the input stereo pair, a stable number of detected features over time and a temporally consistent initialization of error minimization for motion estimation. As a reference, the accuracy and complexity of other VO solutions is listed in [7].

4.2 Taxisat vehicle tests

In the case of the real scenarios tests performed using the robotic car with two cameras, a Trimble 7400 DGPS was installed in the vehicle. With a positioning error of a few centimeters, it can be used as soft groundtruth reference to evaluate the VO results.

For these tests, we used the Taxisat vehicle already presented in the Introduction. The VO system in the Taxisat vehicle is composed of two Point Grey Flea3 cameras FL3-GE-13S2C-C (baseline 50 cm) fixed to the

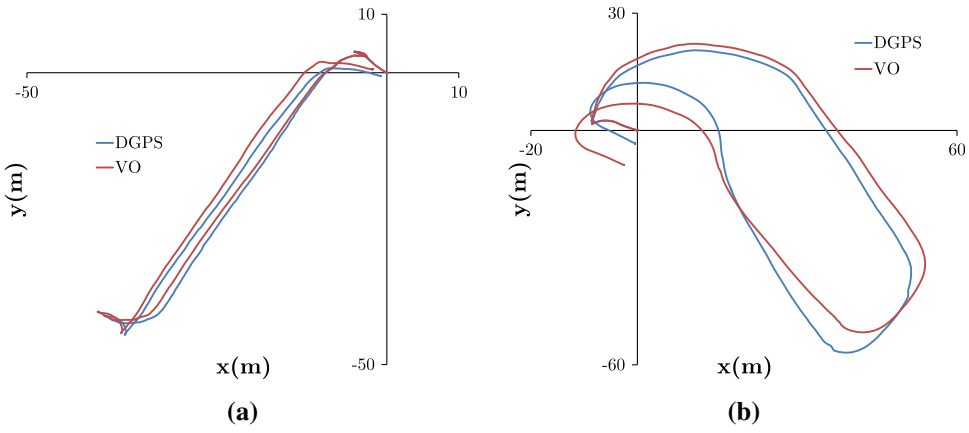

Fig. 9 Comparison of the estimated path for two trajectories

front of the vehicle and connected via Gigabit Ethernet to an industrial computer with an Intel Core i7-2655LE 2.2 GHz CPU. The Intel Core i7-2655LE 2.2 GHz CPU TDP is 25 W, it contains two processing cores and can handle up to four threads. After the system installation, both cameras were calibrated using a checkerboard pattern. Finally, a master/slave configuration was selected for synchronization purposes (hardware synchronization using a trigger wire linking both cameras).

After several days of field tests, 15 min of VO data logs including DGPS information have been recorded and more than an hour of VO-only data logs are available. Motion was computed under sunny, cloudy and even indoor conditions if there is sufficient illumination (exposure times varying from <1–10 ms). The Taxisat vehicle is a low-speed robot with a maximum speed close to 4 m/s, so speed is not a limitation for the accuracy of the system. Finally, the presence of dynamic objects such as cars or pedestrians does not affect egomotion estimation. This type of dynamic objects would have to cover most of the images to discard the background as a reference for motion estimation during Levenberg–Marquardt/RANSAC optimization and erroneously compute motion with respect to dynamic objects.

In Fig. 9, the computed path is compared to the path obtained from the DGPS positioning device for two different trajectories. The results obtained on the Taxisat vehicle cannot be directly compared to the results obtained on the KITTI dataset for several reasons. First, the sampling rate of the DGPS was configured to 2 Hz during the tests. Second, the Taxisat vehicle reaches a maximum speed of 4 m/s. As a consequence, the rotation error, which is divided by the translation to be expressed in rad/m, is much higher than in the KITTI dataset. Despite not being comparable to KITTI data, the results obtained for the two presented trajectories reflect the performance of the proposed VO algorithm in the Taxisat vehicle. For the two trajectories plotted in Fig. 9, the average measured translation error is 3.48 % and the average rotation error 0.1149°/m. These errors are measured each time that the translation of the vehicle exceeds 20 m. In this manner, the low rate of DGPS information is compensated with long enough measurement intervals.

The parameter configuration used is $W = 6 \times H = 4$, $\Delta t = 1$, $N = 1{,}000$, $n_b = 256$, $D_{\max} = 75$, $d_{\max} = 75$, $I = 70$, $p = 85 * \cong$, $R_t = 10^{-4} \times I$, $B_t = 10^{-2} \times I$, $R_r = 10^{-4} \times I$ and $B_r = 10^{-3} \times I$. Compared to the laboratory tests, the new parameter values are adapted to the resolution of the cameras installed in the Taxisat vehicle (644 × 482). Given the cameras refresh rate of 10 frames per second, the number of detected features is increased to 1,000, as well as the minimization iterations to 70, to take full advantage of the 100 ms between frames.

5 Conclusion

In this paper, we presented a new spatiotemporal framework in which most VO algorithms can be casted. Temporal improvements are related to using redundant information from the previous pair of frames. Spatial improvements homogeneously distribute detected features over the images. Both types of optimizations improve the accuracy and the computational complexity of VO algorithms that can be integrated in the proposed framework.

Results are demonstrated on three different CPUs, ranging from a four core and 77 W platform to a one core and 2.5 W processor. Computation time is almost halved for the three architectures, demonstrating the scalability of the proposed approach. Experiments were also conducted on an embedded platform on the Taxisat vehicle, validating the laboratory results.

The code developed in the framework of the Taxisat project is currently being integrated in the Viulib® computer vision library, developed by Vicomtech-IK4. A demo sample will be available in future releases of Viulib®.

Future work includes the integration of new types of odometry sensors in the car such as GNSS or INS with two purposes. First, to compare the results of VO with other types of sensors. And second, to integrate the information of many types of systems to obtain a more accurate and robust estimation of the vehicle odometry.

Acknowledgments The work described in this study was supported by the European Union 7th Framework Programme, contract number 277626-2, and by the NET and ETORTEK programs of the Basque Government under the projects VITEM-F and MOVITIC, respectively.

References

1. Agrawal, M., Konolige, K., Blas, M.R.: CenSurE: Center surround extremas for realtime feature detection and matching, Vol. 5305 LNCS of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2008)
2. Bertozzi, M., Broggi, A.: GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. IEEE Trans. Image Process. **7**(1), 62–81 (1998)
3. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: binary robust independent elementary features. In: European Conference on Computer Vision, pp. 778–792. Springer, New York (2010)
4. Comport, A.I., Malis, E., Rives, P.: Accurate quadrifocal tracking for robust 3D visual odometry. In: IEEE International Conference on Robotics and Automation, pp. 40–45 (2007)
5. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: MonoSLAM: real-time single camera SLAM. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1052–1067 (2007)
6. De-Maeztu, L.: Towards accurate and real-time local stereo correspondence algorithms: computational efficiency and massively parallel architectures. Ph.D thesis, Public University of Navarre (2013)

7. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: The KITTI Vision Benchmark Suite (2014). http://www.cvlibs.net/datasets/kitti/index.php
8. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI Vision Benchmark Suite. In: IEEE Conference on Computer Vision and Pattern Recognition (2012)
9. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: dense 3D reconstruction in real-time. In: IEEE Intelligent Vehicles Symposium, pp. 963–968 (2011)
10. Giffinger, R., Fertner, C., Kramar, H., Kalasek, R., Pichler-Milanovic, N., Meijers, E.: Smart cities-ranking of European medium-sized cities. Technical report, Centre of Regional Science, Vienna (2007)
11. Grauman, K., Darrell, T.: The pyramid match kernel: discriminative classification with sets of image features. IEEE Int. Conf. Comput. Vision **2**, 1458–1465 (2005)
12. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision, 2nd edn. Cambridge University Press, Cambridge (2004)
13. Karlsson, N., Di Bernardo, E., Ostrowski, J., Goncalves, L., Pirjanian, P., Munich, M.E.: The vSLAM algorithm for robust localization and mapping. In: IEEE International Conference on Robotics and Automation, pp. 24–29 (2005)
14. Kitt, B., Geiger, A., Lategahn, H.: Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. In: IEEE Intelligent Vehicles Symposium, pp. 486–492. IEEE (2010)
15. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 225–234 (2007)
16. Konolige, K., Agrawal, M., Sola, J.: Large-scale visual odometry for rough terrain. In: Robotics Research, pages 201–212. Springer, New York (2011)
17. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recogn. **2**, 2169–2178 (2006)
18. Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., et al.: A perception-driven autonomous urban vehicle. J. Field Rob. **25**(10), 727–774 (2008)
19. McCall, J.C., Trivedi, M.M.: Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. IEEE Trans. Intell. Transp. Syst. **7**(1), 20–37 (2006)
20. Milford, M.J., Wyeth, G.F.: Single camera vision-only SLAM on a suburban road network. In: IEEE International Conference on Robotics and Automation, pp. 3684–3689 (2008)
21. Nannen, V., Oliver, G.: Grid-based spatial keypoint selection for real time visual odometry. In: International Conference on Pattern Recognition Applications and Methods, pp. 586–589 (2013)
22. Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recogn. **1**, 652–659 (2004)
23. Otaegui, O., Desenfans, O., Plault, L., Lago, A.: Taxisat project website (2014). http://www.taxisat.net/
24. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European Conference on Computer Vision, pp. 430–443. Springer, New York (2006)
25. Rumar, K.: The role of perceptual and cognitive filters in observed behavior. In: Human Behavior in Traffic Safety, pp. 151–170. Plenum Press, New York (1985)
26. Scaramuzza, D., Fraundorfer, F.: Visual odometry [tutorial]. IEEE Robot. Autom. Mag. **18**(4), 80–92 (2011)
27. Scaramuzza, D., Fraundorfer, F., Siegwart, R.: Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In: IEEE International Conference on Robotics and Automation, pp. 4293–4299 (2009)
28. Scaramuzza, D., Siegwart, R.: Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. IEEE Trans. Rob. **24**(5), 1015–1026 (2008)
29. Tardif, J.-P., George, M., Laverne, M., Kelly, A., Stentz, A.: A new approach to vision-aided inertial navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4161–4168 (2010)
30. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. Int. J. Comput. Vision **9**(2), 137–154 (1992)
31. Whelan, T., Johannsson, H., Kaess, M., Leonard, J., McDonald, J.: Robust real-time visual odometry for dense RGB-D mapping. In: IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May (2013)

Leonardo De-Maeztu received his Ingeniero de Telecomunicación degree in 2007, M.S. degree in 2008 and Ph.D. degree in 2013, all from the ETSIIT, Universidad Pública de Navarra (UPNA), Pamplona, Spain. He also received his M.S. degree in 2006 from the ENSERG, Institut National Polytechnique de Grenoble (INPG), Grenoble, France. He currently works as a researcher of the Intelligent Transportation Systems and Engineering area of Vicomtech-IK4, Donostia-San Sebastián, Spain. His actual research interests include motion estimation and depth computation using multiple cameras.

Unai Elordi received the Ingeniero en Informática degree from the Mondragon Unibertsitatea, Mondragon, Spain, in 2009. He currently works as a researcher of the Intelligent Transportation Systems and Engineering area of Vicomtech-IK4, Donostia-San Sebastián, Spain. His research interests include pattern recognition and motion estimation.

Marcos Nieto received the Ingeniero de Telecomunicación degree in 2005 and the Ph.D. degree in 2010 both from the E.T.S.Ing. Telecomunicación (ETSIT) of the Universidad Politécnica de Madrid (UPM), Spain. He currently works in the Vicomtech-IK4 research center, in the area of Intelligent Transportation Systems and Engineering. His actual research interests include the use of optimization methods for probabilistic models in computer vision as well as the design of H.264/AVC video codecs.

Javier Barandiaran is a researcher at the Intelligent Transportation Systems and Engineering area of Vicomtech-IK4 since 2007. He received a degree in Computer Science (2004) from the University of the Basque Country, Spain. He previously worked as an intern researcher at Ceit-IK4 from 2004 to 2007. He has been involved in several local and international projects focusing mainly on computer vision. His research interests are in the field of computer vision, augmented reality, tracking and 3D reconstruction.

Oihana Otaegui received her M.S. and Ph.D. degrees in mechanical engineering from the Tecnun, University of Navarra, Donostia-San Sebastián, Spain in 1999 and 2005 respectively. She is currently the head of the Intelligent Transportation Systems and Engineering area of Vicomtech, Donostia-San Sebastián, Spain. Her research interests include satellite navigation and transport fields.