De-Maeztu, L., Elordi, U., Nieto, M. et al. *A temporally consistent grid-based visual odometry framework for multi-core architectures.* **J Real-Time Image Proc** 10, 759–769 (2015). This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/s11554-014-0425-y

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

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

Received: date / Revised: date

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, this framework 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 reduces the computational complexity of a visual odometry algorithm by a factor of two 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, 10, 11, 22]. 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 [6]). 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 [18], 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 [4].

Computer vision using cameras bring cheap and robust localization possibilities that are usually classified as Visual Odometry (VO) or Visual Simultaneous Localization And Mapping (V-SLAM) algorithms [19]. The main difference between both types of solutions is that, while VO methods compute motion incrementally (frame after frame), V-SLAM solutions optimize the global consistency of the path. 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 [19].

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.

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

L. De-Maeztu (✉) · U. Elordi · M. Nieto · J. Barandiaran · O. Otaegui
Vicomtech-IK4, Paseo Mikeletegi 57,
Donostia-San Sebastián, Spain
Tel.: +34 943 309230
Fax: +34 943 309393
E-mail: ldemaeztu@vicomtech.org

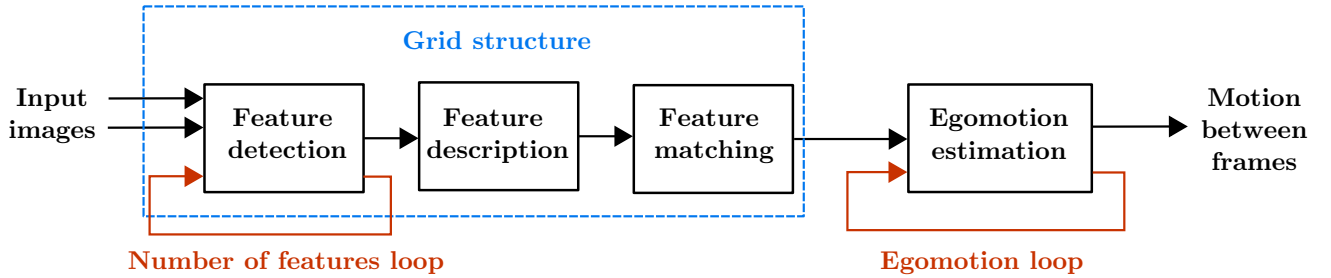


Fig. 1 Execution pipeline implemented by most VO algorithms (in black) along with the proposed spatial (in blue) and temporal (in red) modifications.

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 multi-core architectures. Moreover, the grid can be used to build a mask for feature matching in an efficient manner. In Figure 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 [15] that aims to develop a driverless car that can operate autonomously following a predefined path and stopping if obstacles are detected (see Figure 2).

Fig. 2 TAXISAT driverless car.

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

2 Related work

The problem of estimating relative camera positions and 3D structures using sequences of 2D images is referred as Structure from Motion (SfM) [7, 24]. VO is a particular case of SfM. 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. Monocular VO suffers from the scale ambiguity problem, because the absolute scale is unknown. In the stereo case, the path is estimated in a known scale, since the size of the stereo baseline is known [14]. VO algorithms compute the path by incrementally estimating motion over consecutive frames [19].

Two steps can be clearly distinguished in most stereo VO algorithms (in Figure 1 the first step was further divided in three sub-steps):

1. *Correspondence search between frames.* The objectives of correspondence search between images are twofold. First, in multi-camera setups, correspondence search between pictures taken at the same instant with different cameras is used to triangulate the positions of points in 3D [7]. Second, correspondences between frames taken with the same camera at different instants are used in the second step to compute motion between these instants.
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

Three categories of algorithms can be distinguished according to the type of correspondences used to compute motion: appearance-based, feature-based and hybrid methods. Appearance-based solutions use the intensity information of all pixels in the input images, while feature-based algorithms only use salient keypoints determined using some kind of detector. Hybrid methods use a combination of the previous two techniques.

The groups of appearance-based or hybrid approaches include the works of [12, 21]. In [12], rotation is extracted from the intensity profile of a column intensity graph. Then speeds are estimated based on the rate of image change. In [21], 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 include [9, 20]. In both cases, a quantitative evaluation of the performance of different feature detectors is proposed. After evaluating detectors on different scenarios, the authors of [9] chose CenSurE [1]. On the other hand, after comparing a different set of detectors and trackers, the authors of [20] 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, an homogeneous 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 [13].

According to the authors of [19], feature-based methods are in general preferred over appearance-based ones, because they are more accurate and are computationally less expensive. Moreover, most appearance-based methods have only been applied to monocular VO due to ease of implementation compared to the stereo case. As a consequence, most stereo VO solutions are feature-based, and we limit our discussion from this point to this class of methods.

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 matched features 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 [19], and in the binocular case because it provides more accurate results [14]. Optimization of the reprojection error is generally performed using minimization techniques such as Gauss-Newton [5] or Levenberg–Marquardt algorithm [21, 23].

To increase robustness against outliers, most methods include an implementation of the RANSAC algorithm [5, 8, 9]. Also, enchanced smoothness can be obtained using dynamic models of the vehicle along with Kalman filtering [5].

3 Proposed method

Our stereo VO framework consists of an adaptive grid-based correspondence search stage and a temporal consistent motion estimation stage composed of a Levenberg–Marquardt minimization step along with RANSAC outlier removal and a final Kalman filter.

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 Figure 1). 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.

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) [16, 17] 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 (see Figure 3). Moreover, an adaptive FAST threshold value is used to try 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). 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:

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

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 of Figure 3b, 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 Inde-

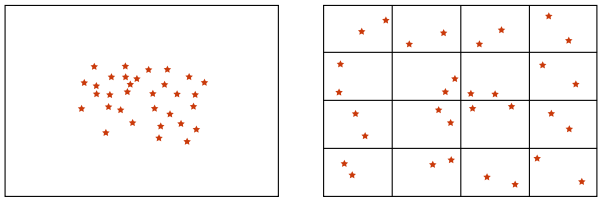


Fig. 3 Comparison of possible feature point distributions (a) without and (b) with a grid structure.

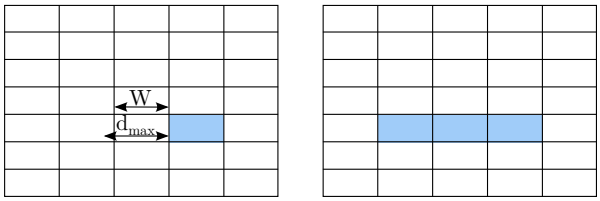


Fig. 4 Grid-based *a priori* mask computation for stereo matching. The left image keypoints belonging to the cell highlighted in (a) can only be matched to the keypoints belonging to the cells of the right image highlighted in (b).

pendent Elementary Features (BRIEF) [3]. The BRIEF descriptor computes for each keypoint a 64 bits 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. Imposing these 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 4 contains a graphical explanation. In this particular example $\lceil d_{max}/W \rceil = 2$, so the candidate cells include the cell occupying the same position as the one in the left image and its two neigbors 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 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 5 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 composed by a 5×3 rectangle of cells. Of course, potential search cells that lie outside the images are not considered.

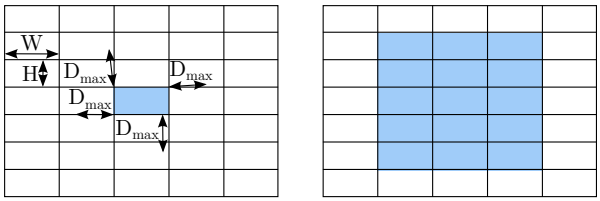


Fig. 5 Grid-based *a priori* mask computation for optical flow computation. The previous frame keypoints belonging to the cell highlighted in (a) can only be matched to the keypoints belonging to the cells of the current image highlighted in (b).

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 [5]. 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 Motion estimation and filtering

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 $\mathbf{X_{k-1}}$ and the 2D coordinates of the features detected in the current stereo pair $\mathbf{x_k}$. If we define $f^l_{\mathbf{r},\mathbf{t}}$ 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 $f^r_{\mathbf{r},\mathbf{t}}$ for the right camera, the cost function to minimize is:

$$\sum_{i=1}^{L} \left\| \mathbf{x}^l_k - f^l_{\mathbf{r},\mathbf{t}}(\mathbf{X}^l_{k-1}) \right\| + \left\| \mathbf{x}^r_k - f^r_{\mathbf{r},\mathbf{t}}(\mathbf{X}^r_{k-1}) \right\| \quad (2)$$

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 [5]. Given that the vehicle motion is relatively smooth, 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 as an initialization value for the Levenberg–Marquardt/RANSAC procedure in each iteration as shown in Figure 6 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.

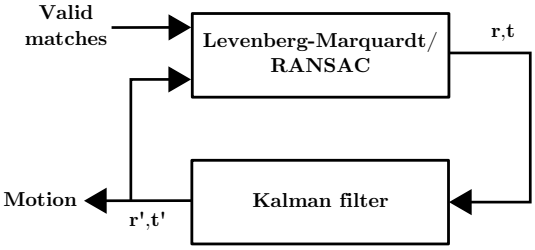


Fig. 6 Motion estimation and filtering pipeline.

4 Experimental results

To test the performance of the proposed VO algorithm, two different benchmarking scenarios were designed. First, the proposed ideas were evaluated in a laboratory computer using the KITTI odometry evaluation dataset with ground truth [4]. Then, we tested the performance of the VO solution on the TAXISAT vehicle.

4.1 Dataset with ground truth

The KITTI Vision Benchmark Suite [4] consists of five individual benchmarks designed to evaluate the performance of different algorithms that are generally integrated in ADAS or autonomous vehicles. One of the five benchmarks is designed to test VO 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. The platform used for evaluating the odometry algorithm with the KITTI benchmark consists in a standard PC with an Intel Core i5-3330 3.00 GHz CPU (four cores). 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.

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.

Table 1 Translation error (%) of the VO algorithm versus the number of rows and columns of the grid structure.

	1	2	4	8
1	2.18	2.14	2.14	2.16
2	2.17	2.13	2.10	2.12
4	2.16	2.16	2.19	2.09

Tables 1 and 2 show that a small accuracy improvement is obtained when integrating the proposed grid in the feature detection stage for certain configurations (a 10% improvement in rotation error and a 4% improvement in translation error). However, this grid-based

[1] http://qt-project.org/
[2] http://www.threadingbuildingblocks.org/
[3] http://opencv.org/

Table 2 Rotation error (deg/m) of the VO algorithm versus the number of rows and columns of the grid structure.

	1	2	4	8
1	0.0134	0.0134	0.0133	0.0134
2	0.0134	0.0128	0.0125	0.0122
4	0.0132	0.0120	0.0121	0.0122

feature detection stage brings two new possibilities to speedup the algorithm: feature detection inside each block of the grid can be implemented independently (i.e., parallel execution in multi-core architectures) and the grid structure of the detected features can be used to avoid matching pixels belonging to distant blocks in an efficient manner in the feature matching stage.

Figure 7 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 in the grid (TB). The analysis is repeated for different grid distributions. The use of two threads for separate processing of the left and right image (TI) clearly speedups the algorithm because both processes are completely independent. Moreover, the grid-based implementation (TB) enables a further speedup of the algorithm thanks to multithreading processing of the cells of the grid.

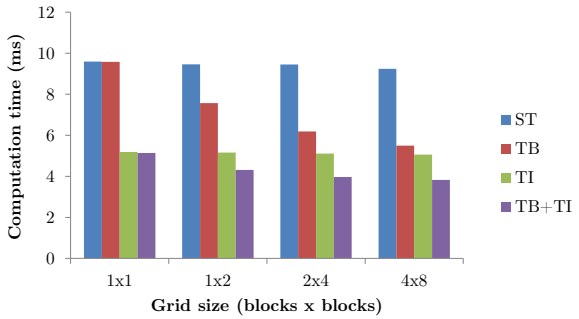


Fig. 7 Feature detection and description execution time.

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 Figure 8 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. The speedup is more important when computing a small set of feature points (systems with a limited computational power).

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

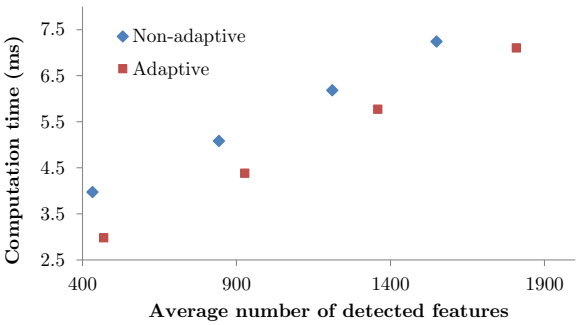


Fig. 8 Comparison of the computation time versus the average number of features when using a fixed value and a variable value for the FAST feature detector threshold.

to be compared. Because computing the Euclidean distance between all potential candidates for 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 Figure 9 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 slightly faster when using large grids and in addition it makes possible to keep a higher number of matches for motion computation. If this matches are not prohibited before matching, they will have to be discarded after, loosing potential matches for motion computation.

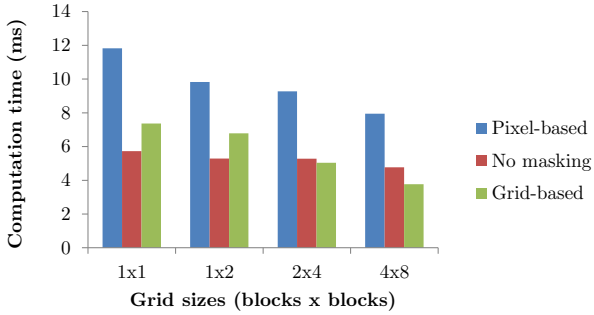
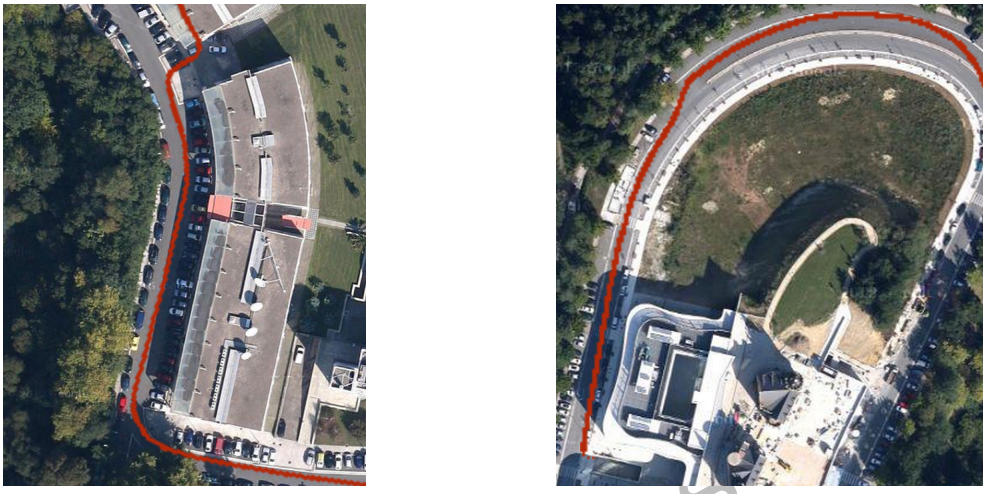


Fig. 9 Feature matching execution time.

Finally, the proposed motion estimation loop (see Figure 6) reduces the translation error from 2.24% to 2.09% and the rotation error from 0.0123 deg/m to 0.0122

Table 3 Accuracy and computation time of the proposed VO algorithm.

Method	Translation error	Rotation error	Runtime
Raw	2.33%	0.0136 [deg/m]	19.6 ms
Proposed	2.09%	0.0122 [deg/m]	9.0 ms



(a) (b)

Fig. 10 Different trajectories computed by the visual odometry algorithm embedded in the TAXISAT vehicle (image source: Google Maps).

deg/m compared to a motion estimation stage in which the last rotation and translation vectors are not used as initial hypothesis for the next pair of frames . Moreover, the convergence of the Levenberg-Marquardt / RANSAC iterative stage is accelerated from 5.2 ms to 2.3 ms.

To summarize the performance of the proposed solution and compare it to a solution without the proposed optimizations, Table 3 contains the accuracy and execution time of different solutions that include some of the modifications proposed in this paper. The parameter configuration for both algorithms is $N = 500$, $D_{max} = 200$, $d_{max} = 150$, $I = 50$, $p = 85\%$. Additionally, for the proposed algorithm, a 4×8 grid is used.

4.2 TAXISAT vehicle tests

In the case of the real scenarios tests performed using a robotic car with two cameras, no ground truth was available. Nevertheless, the performance of the algorithm was previously assessed using the KITTI evaluation benchmark. For these tests, qualitative evaluation was carried out using satellite maps scaled so that the computed path and the map can be registered.

For these tests, we used the TAXISAT vehicle already presented in Section 1. 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 front of the vehicle and connected via Gigabit Ethernet to an industrial computer with an Intel Core i7-2655LE 2.2 GHz CPU. 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). The embedded computer has a power consumption limited to 40 W, thus having a smaller computational power than the desktop computer. However, the resolution of the images used in this case is also smaller than the resolution of the KITTI stereo pairs (644×482 pixels). As a consequence, the execution time is similar to the execution time in the desktop PC (around 10 ms).

In Figure 10 the computed path is plotted on satellite images with the appropriate scale. The drift is small enough to keep the computed path inside the road for several hundreds of meters. The fusion of VO with other types of sensors would further improve the accuracy of the results enabling accurate navigation in longer paths.

5 Conclusion

In this paper, we presented a new spatiotemporal framework in which most VO algorithms can be casted. The framework enables egomotion computation at high framerates on a desktop computer thanks to a full exploitation of multi-core architectures. 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.

The results of the algorithm are also demonstrated on an embedded platform on the TAXISAT vehicle. Despite the lower consumption of this CPU, the proposed framework is executed in 10 ms, producing accurate results over several hundreds of meters. Free computational power is left for implementing other computer vision methods such as road lane tracking or obstacle detection in real time.

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.

6 Acknowledgements

The work described in this study was supported by the European Union 7$^{\text{th}}$ 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. M. Agrawal, K. Konolige, and M. R. Blas. *CenSurE: Center surround extremas for realtime feature detection and matching*, volume 5305 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2008.
2. M. Bertozzi and A. Broggi. GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, 7(1):62–81, 1998.
3. M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *European Conference on Computer Vision*, pages 778–792. Springer, 2010.
4. A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI Vision Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
5. A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, 2011.
6. R. Giffinger, C. Fertner, H. Kramar, R. Kalasek, N. Pichler-Milanovic, and E. Meijers. Smart cities-ranking of European medium-sized cities. Technical report, Centre of Regional Science, Vienna, 2007.
7. R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge University Press, 2000.
8. B. Kitt, A. Geiger, and H. Lategahn. Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 486–492. IEEE, 2010.
9. K. Konolige, M. Agrawal, and J. Sola. Large-scale visual odometry for rough terrain. In *Robotics Research*, pages 201–212. Springer, 2011.
10. J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.
11. J. C. McCall and M. M. Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):20–37, 2006.
12. M. J. Milford and G. F. Wyeth. Single camera vision-only SLAM on a suburban road network. In *IEEE International Conference on Robotics and Automation*, pages 3684–3689. IEEE, 2008.
13. V. Nannen and G. Oliver. Grid-based spatial keypoint selection for real time visual odometry. In *International Conference on Pattern Recognition Applications and Methods*, pages 586–589, 2013.
14. D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–652. IEEE, 2004.
15. O. Otaegui, O. Desenfans, L. Plault, and A. Lago. *TAXISAT project website*, http://www.taxisat.net/.
16. E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1515, 2005.
17. E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443. Springer, 2006.
18. K. Rumar. The role of perceptual and cognitive filters in observed behavior. In *Human Behavior in Traffic Safety*, pages 151–170. Plenum Press, 1985.
19. D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics & Automation Magazine*, 18(4):80–92, 2011.
20. D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In *IEEE International Conference on Robotics and Automation*, pages 4293–4299, 2009.
21. D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on*

Robotics, 24(5):1015–1026, 2008.
22. Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):694–711, 2006.
23. J.-P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz. A new approach to vision-aided inertial navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4161–4168, 2010.
24. C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.