Fast Stereo-based Visual Odometry for Rover Navigation

ALDO CUMANI and ANTONIO GUIDUCCI
Istituto Nazionale di Ricerca Metrologica
Str. delle Cacce, 91, I-10135 Torino
ITALY
{a.cumani,a.guiducci}@inrim.it

Abstract: The object of visual odometry is the computation of the path of a rover from onboard passive vision data only. The approach presented here relies on the accumulation of ego-motion estimates obtained by stereo vision and bundle adjustment of tracked feature points. We also propose a new feature detector/descriptor, which is a simplified and faster form of other well known descriptors (SURF). For cyclic paths, a déjà vu mechanism allows further control over the accumulated error. Tests on real-world data show that our descriptors are effective for accurate path estimation, while being fast enough for use in tasks such as autonomous planetary exploration.

Key–Words: Robot localisation, Stereo vision, SLAM, Hessian feature detectors

1 Introduction

Research in autonomous sensor-based navigation has received considerable attention in recent years, particularly in connection with planetary exploration tasks [1]. A mobile platform (rover) on, say, Mars must be capable of truly autonomous navigation in a mostly unknown environment, as its continuous teleoperation from an Earth station is clearly out of question. This requires that the rover be able to build, based on sensor measurements, a metric and topological model of its environment, while simultaneously estimating its position relative to the environment itself. This is the task of Simultaneous Localisation and Mapping (SLAM) studies [2]. SLAM algorithms have been proposed, using various kinds of sensors; among these the vision sensor, being the one that provides the largest amount of information, has been extensively studied both in single-camera [3, 4, 5] and in multi-camera setups [6, 7, 8].

Reliable navigation, however, requires accurate localization, i.e. accurate estimation of the rover path. Indeed, pure dead reckoning (wheel odometry) usually yields quite poor estimates (due e.g. to wheel slippage); also, wheel odometry alone can at most yield a 2D path estimate, so it is not even sufficient in principle when navigating on a non-planar surface, and must be complemented by other independent inputs (e.g. an absolute orientation sensor as in [1]).

On the other hand, vision-based path estimation (*Visual Odometry*) is able to yield accurate results while being intrinsically 3D. In this regard, it is important to note the following points:

- Any path estimate from onboard visual measurements is necessarily *incremental*, i.e. resulting from the sum of smaller motion estimates. It is therefore of utmost importance that each individual step be as accurate as possible, to reduce error accumulation.
- Monocular vision is subject to scale uncertainty, so, in absence of landmarks of known size (as is the case in planetary exploration), stereo or other multi-camera setups are needed.
- As error accumulation in the long run is unavoidable, it is important that the rover be able to recognize places where it has been before, and to use such information to correct its pose estimate.

In this context, our group at INRIM has developed a visual odometry algorithm [9, 10, 11, 12] which relies on the tracking of pointwise image features extracted from the images acquired by an onboard binocular stereo head. At intervals along the rover trajectory, its motion is estimated by robust bundle adjustment of the tracked features in the four images (two before and two after the motion). Several kinds of point features have been tested to this end, and a new Fast-Hessian based feature detector/descriptor, similar to SURF [13] has been developed.

A *déjà vu* mechanism for exploiting cyclic paths has also been devised, by periodically storing observed features and pose estimates, and comparing currently observed features to stored ones when near a saved position.

This work describes in some detail the above approach, and presents the results of several real-world tests performed in our laboratories. The paper is organized as follows: Sec. 2 formalizes the algorithm;

Sec.3 describes the feature detection and matching steps; Sec. 4 discusses the motion estimation algorithm and the déjà vu mechanism. Sec. 5 illustrates implementation issues, and finally Sec. 6 presents and discusses test results.

2 The visual odometry algorithm

The algorithm relies on accumulating relative motions, estimated from corresponding features in the images acquired while the rover is moving. Such estimates are computed at *key frames*, whose spacing is a compromise between larger intervals, desirable both for numerical accuracy and for reducing computations, and the need for a sufficiently large number of features, which naturally tend to be lost because going out of view.

The algorithm can be so summarized:

Feature extraction. Point features are extracted at each key frame and left-right matched.

Feature tracking. Matched features are tracked in subsequent frames, where stereo matching is performed again.

Motion estimation. The relative motion of the rover between the current frame and the last key frame is estimated by bundle adjustment of matched feature points.

Déjà vu correction. Features and pose estimates are periodically saved. When the rover believes to be near a saved position, observed features are compared to the stored ones, and a pose correction is possibly computed by bundle adjustment.

The following paragraphs describe the above steps in greater detail.

3 Features

As said in Sec. 1, several kinds of point features have been tested, namely:

- Shi-Tomasi-Kanade features [14], which however require small image displacements for reliable tracking, and a stereo disparity cue for stereo matching;
- SURF descriptors [13], which instead allow direct stereo matching and tracking with longer inter-frame steps, though still imposing a non negligible load;
- new simplified Fast-Hessian based descriptors [12], similar to SURFs but faster to compute.

The next sections describe the latter new descriptors in some detail.

3.1 Detection

Detection of point features is accomplished by a fast approximate Hessian blob detector. Feature points are sought for as the maxima of the normalized Hessian

$$H(\sigma) = \sigma^2(I_{xx}(\sigma)I_{yy}(\sigma) - D(\sigma)I_{xy}^2(\sigma)) \qquad (1)$$

where $I_{xx}(\sigma)$, $I_{yy}(\sigma)$ and $I_{xy}(\sigma)$ are the second derivatives of the image intensity I filtered with a Gaussian of scale σ (the factor $D(\sigma)$ shall be explained later). Following the approach in [13], the smoothed derivatives are approximated by box filters like those depicted in Fig. 1, which represent the derivatives at scale $\sigma = 1.2$ (filter size 9×9).

The use of box filters has the advantage that such filters can be implemented very efficiently by using the so-called *integral image* [15], requiring only four memory accesses and three add/subtract operations per box, irrespective of filter size. So, I_{xy} needs 16 accesses and 12 operations, while I_{xx} and I_{yy} are even cheaper - they can be computed as differences of two boxes, requiring only 8 accesses, 6 adds and one multiply each.

The box filter approximation is the reason for the factor $D(\sigma)$ in (1), which takes into account the different norms of the filter masks - it would be 1 in the absence of approximations, while with masks like those in Fig. 1 $D(\sigma)$ ranges from $\sim$0.913 (9×9 mask) to $\sim$0.996 (195×195 mask).

This Hessian detector is applied at various filter sizes, corresponding to a scale range of up to four octaves ($\sigma = 1.2$ to $\sigma = 26$). Feature points are then found by non-maximum suppression over pixel space and scale space. A threshold on the value of H is also applied, in order to limit the detected points to the strongest ones. Maximum locations are further refined to sub-pixel and sub-scale accuracy by fitting a quadratic function to the 27 values of H in the $3\times3\times3$ neighborhood of each maximum point. Note that we, as [13], use the same criterion, i.e. the Hessian, for both spatial localisation and scale estimation, in contrast to other mixed approaches as e.g. the Hessian-Laplace favored by [16].

3.2 Description

The image information around each detected point is encoded into a descriptor as follows. For a point (x, y, s) detected at scale $\sigma = s$, a square image area of side $10s$ centered at (x, y) is processed and reduced to a size of 8×8 pixels, by block averaging and bilinear interpolation. The resulting gray values are then reduced to zero mean, weighted by a Gaussian of scale $2s$ and the resulting vector of 64 floats is finally normalised to unit length. These 64 values, together with

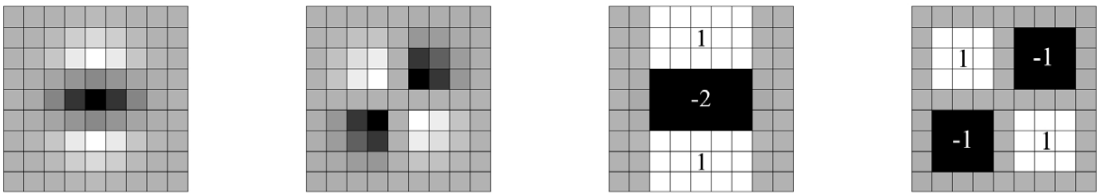

Figure 1: The discretised and cropped Gaussian ($\sigma = 1.2$) second derivative filter masks in yy and xy.(left), and their box approximations (right) (reproduced from Bay et al. [13])

x, y and s constitute the feature descriptor. Note that choosing an 8×8 size is just in order to have the same number of descriptors as standard SURFs - a different (perhaps larger) size can be easily accommodated.

This kind of descriptor is similar to the one named *cross correlation descriptor* in [16], the main difference being the Gaussian weighting and the zero-mean and unit normalisations, which tend to make the descriptor more robust against affine illumination changes. In fact, this descriptor has the drawback of being neither affine- nor even rotation-invariant (it is scale- and lighting-invariant, though). However, it is extremely fast to compute (also thanks to the use of the integral image for block averaging), and our experience has shown it to be quite adequate for moderate baseline matching tasks, like the stereo and motion matching needed by our visual odometry algorithm for accurate path estimation.

Table 1 compares our method to the SURF method. For the latter, we have used the author's implementation available on the ETH site [13]. Both the full SURF descriptors and the upright-SURF (U-SURF) variant have been tested, the latter not being rotation-invariant and therefore more directly comparable to our method. These tests have been performed on a HP DC7700, equipped with a Pentium D at 2.6 GHz and running Linux Debian 4 (Sid). Note that the DC7700 has a dual-core processor, but this is largely irrelevant as both our method and the SURF one have been implemented as single-thread tasks.

	points	time (ms)	pts/ frame	good pts
SURF	1474	1605	2451	155
U-SURF	1474	808	2451	204
New method	1448	391	2401	151

Table 1: Comparison of new method and SURF (see text for explanations).

The first two columns compare average execution times on a small subset of typical images taken from the sequences discussed in Sec. 6; the parameters (Hessian threshold) have been adjusted to have more or less the same number of points detected by the two methods. Timings include the building of the integral image, the detection of features and the computation of descriptors.

The last two columns of the table compare the average output of the two algorithms on the outdoor test of Sec. 6 in terms of number of total detected points at each frame and points "good" for path estimation, i.e. left-right and forward matched, as explained in Sections 3.3 and 4.1. As the table shows, SURF and the new method show comparable performances. Rather surprisingly, U-SURF yields a slightly higher number of good matches, yet our method is about twice faster than U-SURF and almost four times faster than full SURF.

3.3 Matching

Both left-right and temporal matching relies on a similarity measure defined in terms of the Euclidean distance between feature descriptors. The well-known Nearest-Neighbour-Ratio method [17] is used for matching: a given feature point F_1 in image 1 matches a point F_2 in image 2 if F_2 is the nearest neighbour of F_1 (in feature space), *and* the ratio of the distance of F_1 to F_2 to that of F_1 to its *second* nearest neighbour does not exceed a threshold r. In our tests, we have used $r = 0.5$, which may seem a rather restrictive value (Lowe suggests 0.8 in the cited paper), but in fact has been found able to reject most of the false matches.

Matching reliability is further improved by requiring *bidirectional* matching: correspondences are computed in both directions (i.e. from image 1 to image 2 and viceversa), and only those returning back to the same point are kept.

The matching algorithm can be improved by exploiting specific matching constraints (e.g. positive disparity for the stereo case). In our implementation, only stereo constraints are imposed, i.e. positive disparity and vertical distance under some preset thresh-

old. No constraint is imposed instead on temporal matching.

Stereo matched point pairs are then backprojected to a 3D point estimate, in the camera reference. This 3D estimate is used as a starting point for the bundle adjustment step. The features detected in the left image at a key frame are then tracked along the sequence to the next key frame.

4 Path estimation

4.1 Interframe motion estimation

This issue has been discussed elsewhere [11], but is summarized here for the sake of completeness. Given two consecutive keyframes, say 1 and 2, let $\mathbf{X}_i = [x_i, y_i, 1, t_i]^\top$ be the unknown 3D projective coordinates of some feature $\mathcal{F}_i$, $i = 1 \dots N_{12}$, in the left camera reference at keyframe 1. Let $\mathbf{u}_{iq} = [u_{iq}, v_{iq}]^\top$ be the 2D coordinates of $\mathcal{F}_i$ in image $q \in \{1L, 1R, 2L, 2R\}$, and $\mathbf{x}_{iq} = [su_{iq}, sv_{iq}, s]^\top$ its projective representation. Then

$$\begin{array}{rclcrcl} \mathbf{x}_{i,1L} & = & \mathsf{P}_L\mathbf{X}_i & & \mathbf{x}_{i,2L} & = & \mathsf{P}_L\mathsf{M}_{12}\mathbf{X}_i \\ \mathbf{x}_{i,1R} & = & \mathsf{P}_R\mathsf{M}_S\mathbf{X}_i & & \mathbf{x}_{i,2R} & = & \mathsf{P}_R\mathsf{M}_S\mathsf{M}_{12}\mathbf{X}_i \end{array} \tag{2}$$

where the intrinsic camera matrices P_L and P_R are known from calibration, as is the 4×4 stereo transform matrix M_S. The only unknown (apart from $\mathbf{X}_i$) is the motion matrix M_{12}, which can be parametrized in terms of a 3-rotation $\mathbf{r}_{12}$ and 3-translation $\mathbf{t}_{12}$.

Now, with N_{12} observed features $\mathbf{u}^*_{iq}$ tracked from keyframe 1 to 2, we can define an error measure

$$\begin{array}{rcl} J(\mathbf{p}) & = & \sum_i \sum_q f(\|\mathbf{e}_{iq}(\mathbf{p})\|^2) \\ & = & \sum_i \sum_q f(\|\mathbf{u}_{iq}(\mathbf{p}) - \mathbf{u}^*_{iq}\|^2) \end{array} \tag{3}$$

parametrized as a function of the $6 + 3N$ unknowns

$$\mathbf{p} = [\mathbf{r}_{12}, \mathbf{t}_{12}, x_1, y_1, t_1, \dots x_N, y_N, t_N]^\top \tag{4}$$

Then, $\hat{\mathbf{p}} = \arg\min J$ yields the visual odometry estimate $(\hat{\mathbf{r}}_{12}, \hat{\mathbf{t}}_{12})$ of the inter-keyframe motion. This problem can be efficiently solved by a standard sparse Levenberg-Marquardt algorithm (see e.g. [18, A4.3]). For a robust estimate, we take as $f(\cdot)$ a Lorentzian cost function $f(e^2) = \log(1 + e^2/\sigma^2)$ with σ chosen as a function of the expected image-plane error. For better accuracy, the residuals $\mathbf{e}_{iq}$ are compared against a threshold (proportional to the Lorentz σ), and those exceeding that threshold are marked as outliers. The bundle adjustment is then run again on the inliers only.

4.2 Adaptive keyframe determination

The optimal inter-keyframe step must be a compromise between longer steps (less computations and better accuracy), and the need for a sufficient number of tracked features. In our tests, which have only involved batch processing of pre-acquired image sequences, we have adopted the following strategy:

- start with a predefined maximum inter-frame step, say 20 frames;
- if the number of tracked points is enough, go on; else halve the step and retry.

This strategy can also be used in a realtime, non-batch application, provided that a sufficiently large memory buffer be available to store the incoming max number of images between two keyframes. It must be noticed, however, that small steps are only needed in the presence of large image-plane motions, i.e. when the rover turns; in fact, a rough estimate of the expected number of lost features can be easily obtained from the known programmed rover motion, so that the keyframe step can be predicted.

4.3 Déjà vu correction

To implement déjà vu correction, the rover keeps an archive of scene descriptions. Each description consists of the set of stereo matched features, together with the estimated rover pose. The description of the current scene is added to the top of the archive when:

- enough time is elapsed since the last entry in the archive (the top entry);
- the number of matched features between the frame to be added and the top entry is near zero.

This strategy is aimed to ensure that the scenes corresponding to the archive entries are all different.

When a new key frame is processed, the estimated pose is checked against the stored ones, starting from the bottom (oldest) entry. If an entry with similar pose is found, the current features and those stored in the entry are compared, and if enough matches are found a correction between the archived pose and the current one is computed by bundle adjustment. The corrections computed while the rover remains near the archived position are saved. When the rover leaves the archived position, the saved correction corresponding to the frame with the highest number of matches is picked and actually applied. The pose corrected this way is expected to be more accurate, as the error accumulated along the intervening path is exchanged with that from a single keyframe pair.

Note that the above discussion implicitly assumes that the features used for implementing the déjà vu mechanism are the same as those used for visual odometry, but this is not compulsory. In fact, this mechanism requires matching of scenes seen from much more different points of view than is the case for odometry, and in this case both SURF and the new features may not be enough reliable (too few matches). However, descriptions for the déjà vu mechanism are only needed at a much lower rate, and this may justify additional computations for other kinds of features (e.g. Maximally Stable Extremal Regions [19]).

5 Implementation issues

The proposed method has been implemented in C and MATLAB, with the feature detection/description, matching and bundle adjustment in C for reasons of speed. The algorithm has been tested on sequences of real images acquired by a stereo head mounted on a small commercial rover. Indoor and outdoor tests have been carried out, using either of the two rovers at our disposal, namely the 3-wheel ActivMedia Pioneer 3-DX, and the slightly more rugged 4-wheel 2-AT model (see Fig. 2).

Figure 2: The Pioneer 3DX (left) and 2AT (right) rovers, equipped with stereo head and controlling PC.

As shown in the photos, both rovers are equipped with a support for carrying a laptop PC. The latter is used both for sending motion commands to the rover, as well as for providing control signals to the cameras, acquiring and storing the resulting images.

Note that in these tests no image processing was done by the PC, which was used only for storing image sequences - this choice allowed to have some constant data sets for testing different algorithms and different values of algorithm parameters. Also, the rover path was not pre-programmed; in fact, the rover was actually driven "by hand", by sending appropriate commands (start, stop, turn left/right etc.) to the laptop via a standard RF remote control.

5.1 The stereo head

The stereo head consists in a pair of Basler A312f digital cameras, equipped with 6mm lenses. The cameras are mounted on an aluminum slab, as shown in Fig. 3. Each camera provides a stream of CCIR-size (720×576) 8-bit graylevel images on a IEEE1394 bus. The two cameras can be accurately synchronized by an external trigger signal, provided in our case by the controlling PC through the parallel port.

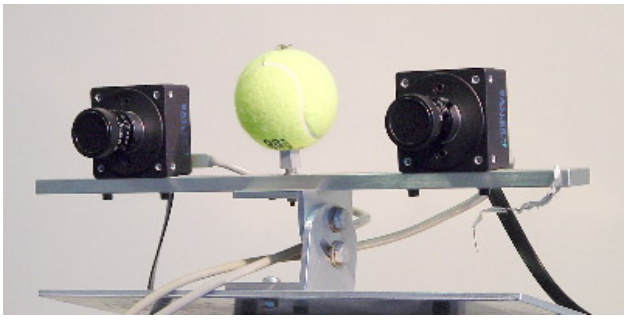

Figure 3: Detail of the stereo head.

The cameras were mounted at a nominal center-to-center distance (stereo baseline) of 236 mm, and the head was calibrated using Bouguet's method [20], with a 0.6×0.6 m checkerboard pattern.

As can be seen from the photos, the two cameras are mounted with parallel axes, but the head is slightly tilted (about 13°) towards the ground. In fact, a good estimate of robot motion requires both distant features, providing reliable hints about direction, as well as near ones, which yield more reliable hints about the distance traveled between keyframes. The arrangement used allows to increment the number of usable near features, while cutting off most of the sky area, which does not provide any useful information.

5.2 Ground truth determination

Assessing the accuracy of a path estimation algorithm requires that such path be independently measured by some means. Two problems arise in this context, namely:

a) the rover position (and, where possible, orientation) must be actually measured with respect to some fixed World reference Frame (WF), and

b) the rototranslation $(\mathsf{R}_L, \mathbf{t}_L)$ linking the above reference and the Rover reference Frame (RF) has to be determined.

As concerns item a), we have used the following setup. As seen in Fig. 3, a standard tennis ball was rigidly attached to the stereo head, to serve as visual target for defining the rover position. Four tennis balls were laid on the test field (see Fig. 4), three roughly at the corners of an equilateral triangle, and a fourth in the middle to be used as check. Their distances were measured to within ± 2 mm, so defining a metric world reference frame. Two hand-held compact digital cameras, after intrinsic calibration again by Bouguet's algorithm, were used to frame the scene at about 90° with respect to the center of the reference frame. At several times the rover was stopped and images of the scene were taken by the two cameras; the ball centers, manually extracted from each image pair, allowed then to estimate a ground-truth position of the rover in the chosen reference, to within an accuracy better than the ball diameter (order of 0.06m).

Figure 4: The Pioneer 2AT rover on the test field. Red arrows indicate the measurement base targets.

As concerns item b), note that the output of the visual odometry algorithm is a series of poses relative to the starting one, and expressed in the frame of reference RF of the stereo head at the starting position. Linking RF to WF requires an *exterior calibration* of the stereo head when the rover is at the starting position, which in turn requires arranging some targets visible by the cameras and accurately measured in WF. For the purpose of the tests here reported, however, we have not used any of the standard calibration methods available; in fact, assuming a measurement of the starting head position in WF be available, the translation part $\mathbf{t}_L$ of the linking transform can be easily obtained by identifying the origin of the path with such position (and neglecting the measurement error). There remains the rotation part R_L to be determined, and to this extent we have used different *ad hoc* methods that shall be discussed with the tests in Sec. 6.

5.3 Image preprocessing

The use of point features that can be directly matched allows to avoid some costly preprocessing such as stereo rectification at the image level, as was done e.g. in [10] for the purpose of computing a dense depth map. Indeed, except for very high values of lens distortion (which is not the case with "normal" lenses), the lens distortion and stereo rectification transform can be applied directly to the feature point coordinates.

However, in the case of a relatively featureless ground like the one observed in our outdoor tests, we have found that applying the detection algorithm to a LOG filtered version of the images significantly increases the number of usable near features. The scale of the LOG filter has been empirically set to $\sigma = 3$, while its gain was adaptively chosen among three values (30,60,90) so as to keep the number of detected features more or less constant (see Fig. 5 for an example, referring to the data of Sec. 6.3).

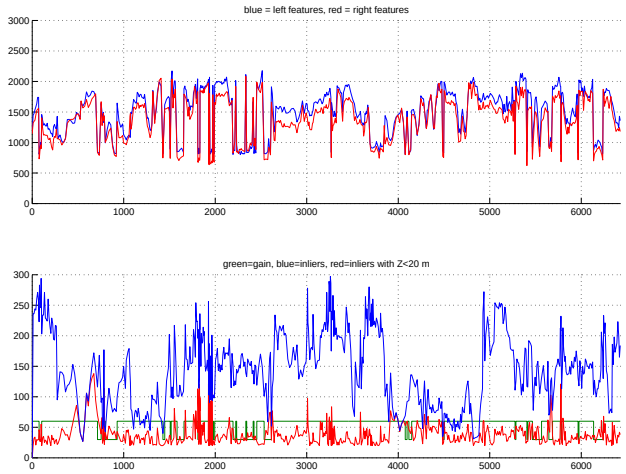

Figure 5: Top: Extracted left (blue) and right (red) features vs. frame number. Bottom: LOG gain (green), number of matched features (blue) and number of features with $z < 20$m (red).

6 Test results

6.1 Indoor test

The following figures show the outcomes of the indoor test, conducted in our laboratories (Fig. 6). This test was performed using a constant inter-keyframe step of 10. The new features, as well as SURF and U-SURF features were tested, with similar results; for clarity, the drawing only shows the path estimated with the new simplified features. To assess the estimation accuracy, the rover was stopped and its position

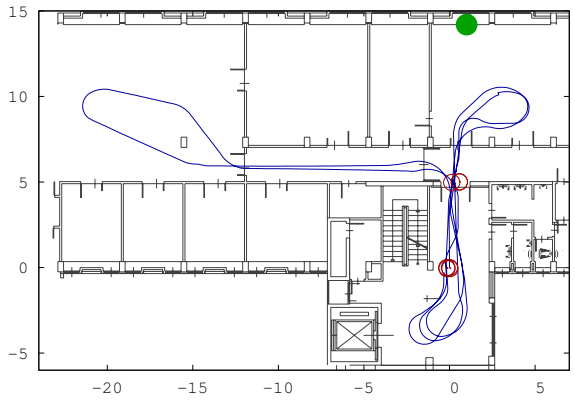

Figure 6: Indoor test. The estimated rover path, approximately superimposed to the floor plan. Coordinates in m.

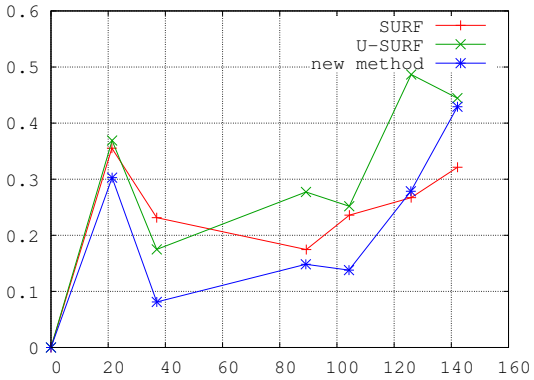

Figure 7: Indoor test. Raw estimated rover path error versus path length. Coordinates in m.

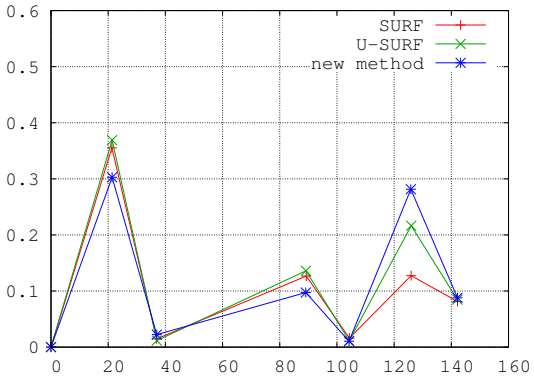

Figure 8: Indoor test. Corrected rover path error versus path length. Coordinates in m.

measured when passing near some predefined points, marked with red circles in the figure. Note that in this case we did not use the setup discussed in Sec. 5.2, which would have been impractical. The rover position was instead determined by measuring the head target distance from the walls. Two DOFs of R_L were fixed by using a single target, marked by the green dot in the Figure; the remaining DOF was arbitrarily fixed by aligning the estimated path to the ground using a graphical tool. Note that the latter arbitrarity has little effect on the error figures reported next, as the measurement positions are almost aligned vith the (measured) direction of the target.

Figs. 7 and 8 plot the position error (distance of the estimated robot position from the measured one) at the measurement stops, respectively without and with the *déjà vu* correction. As can be seen from these plots, using either SURF-based features or the new ones does not yield significantly different results as concerns path accuracy. Also, the déjà vu mechanism appears to be effective in reducing the accumulated error.

6.2 Outdoor test 1

A first outdoor test was done on a paved area within the INRIM campus. Ground-truth rover position data were collected by using the arrangement explained in Sec. 5.2. In order to have more data to compare with the outcome of our algorithm, the rover was driven to perform several turns without going too far from the reference base, as shown in Fig. 9.

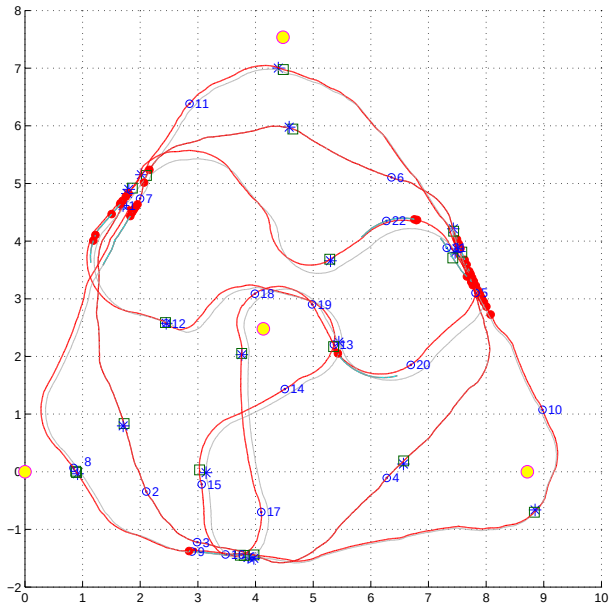

Figure 9: Outdoor test 1. Estimated rover path without (light gray) and with (red) corrections. Dots are the reference targets. Stars and squares are the measured and estimated stop positions. Coordinates in m.

The latter figure plots the estimated rover path; for the sake of graph clarity, only the estimate using the new features is drawn. Figs. 10 and 11 show

the measured position error vs. path length, respectively without and with déjà vu corrections, either using SURF features or our simplified ones.

In this case, the rotation R_L linking the rover reference RF to the base WF was estimated as follows. Let $\mathbf{x}_{Mi}$ be the head target coordinates measured, at the stop positions $i = 0 \ldots N$, in WF, and $\mathbf{x}_{Oi}$ the corresponding ones in the RF as estimated by the visual odometry algorithm. The rotation was estimated by fitting the *directions* of the vectors $\mathbf{x}_{Oi} - \mathbf{x}_{O0}$ and $\mathbf{x}_{Mi} - \mathbf{x}_{M0}$ in some initial subset $i = 1..N'$ with $2 \leq N' \leq N$ ($N' = 4$ for the reported plots). While this approach may be expected to introduce some bias in the estimated errors (since the estimated rotation depends upon the outcome of the visual algorithm itself), in fact this bias is only relevant for the first few measurements - we have verified that the plots reported in Figs. 10 and 11 do not change substantially for other values of N' (e.g. from 2 to 5).

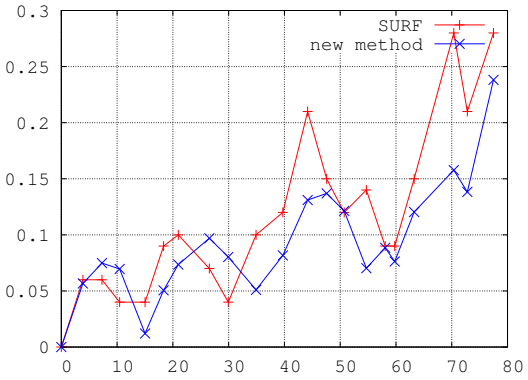

Figure 10: Outdoor test 1. Raw estimated rover path error versus path length. Coordinates in m.

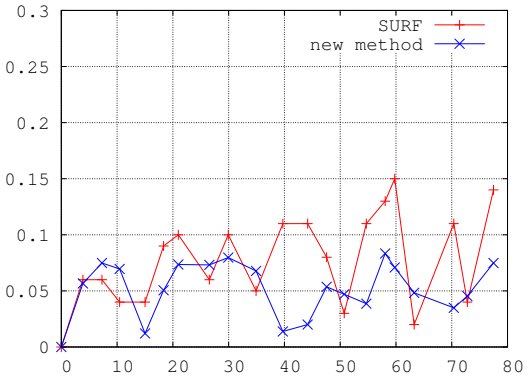

Figure 11: Outdoor test 1. Déjà vu corrected rover path error versus path length. Coordinates in m.

With regard to these plots, it is worth noting (though this may be accidental) that in this case our features perform slightly better than SURFs - and with a lower computational cost. Anyway, also in this case the déjà vu mechanism appears effective in keeping the accumulated error low.

6.3 Outdoor test 2

This other outdoor test was performed on a paved road near one of INRIM buildings (see Fig. 4). Ground-truth data of rover position were collected, as in the previous case, using the setup described in Sec. 5.2. In this test, in order to get a much longer path, the rover was driven quite far from the measurement base, which however implied having quite fewer ground-truth data.

The results from the latter test are reported in the following. Both full SURF features and the new simplified ones were tested, using the adaptive keyframe strategy of Sec. 4.2 with a maximum keyframe step of 20 frames, corresponding to a path length of about 1 m. Fig. 12 plots the estimated path (again, for clarity only the result using the new features and no cyclic correction is shown).

Figures 13 and 14 plot the position error without and with the déjà vu correction, respectively (note that the déjà vu mechanism in this test was triggered only once, after about 220m of path). Due to the small number of measurements, the method described in Sec. 6.2 for determining R_L was not used. Instead, the rotation was estimated by fitting the directions of two points on the building visible in the background of Fig. 4. A graphical tool was then used to check that the error plots reported in Figs. 13 and 14 did not change substantially for rotations as large as $\pm 5^\circ$ around either of the 3 axes of R_L.

In this case, the result using the simplified features clearly outperforms the one with SURFs. It is also worth noting that, even without corrections, the position error remains under 0.9 m over a total path length of more than 300 m.

7 Concluding remarks

In this paper we have presented a visual odometry algorithm using a new, fast Hessian-based feature detector/descriptor, which has proven effective for use in moderate baseline matching tasks like the ones required by our algorithm. The main advantage of the new features is greater speed, without sacrificing accuracy. For example, in the test of Sec. 6.3 the features were extracted from 363 pairs over a total of 6428, taken at an average rate of 4.3 pairs/s (not counting stops). As feature extraction took less than 0.5 s average per image - i.e. 1 s per pair, this implies a computing load less than 25% at a rover speed of 0.22 m/s.

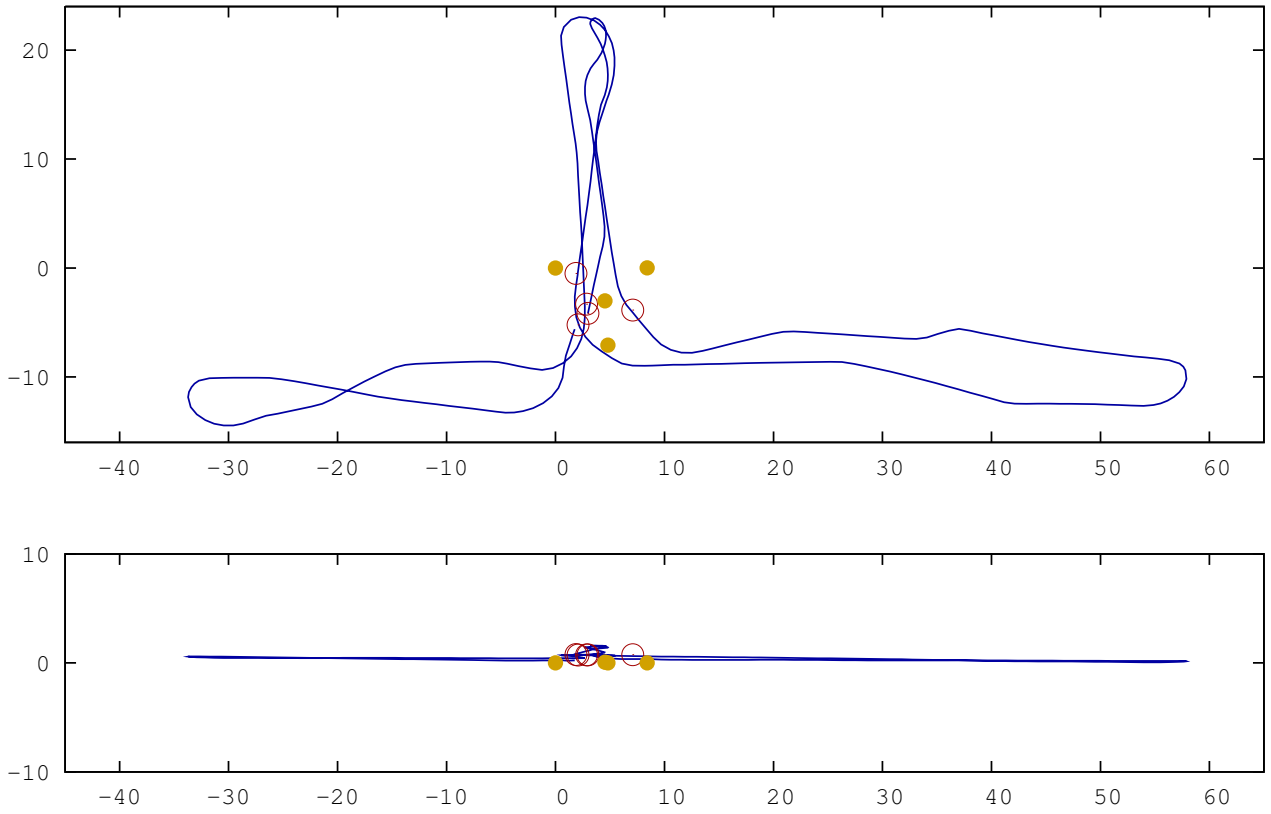

Figure 12: Outdoor test 2. The estimated rover path (new method, no correction). Dots are the reference targets, circles the measured positions. Coordinates in m.

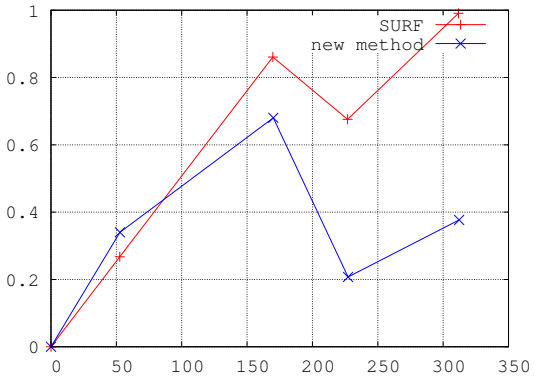

Figure 13: Outdoor test 2. Raw estimated rover path error versus path length. Coordinates in m.

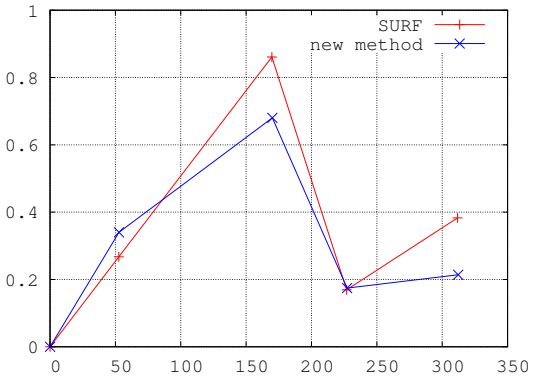

Figure 14: Outdoor test 2. Corrected rover path error versus path length. Coordinates in m.

The proposed déjà vu mechanism has also proven effective in reducing the accumulated error over cyclic paths, although there is still much space for improvement. Indeed, in our tests the use of both the original SURF features, and, to an even greater extent, of the new ones has been found not too reliable for triggering the déjà vu mechanism, as a sufficient number of matched features was only found when the rover passed quite near, and with a very similar heading, to a stored position. Therefore, in order to implement a reliable, fully autonomous landmark-based navigation system more research is needed for finding a fast and reliable representation of scene contents useful for landmark recognition.

Acknowledgment - This work has been partly supported by Regione Piemonte in the framework of the STEPS Project, lead by Thales Alenia Space Italia.

References:

[1] C. F. Olson, L. Matthies, M. Schoppers, and M. W. Maimone, "Robust stereo ego-motion for long distance navigation.," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2453–2458, 2000.

[2] S. Thrun, "Robotic mapping: A survey," Tech. Rep. CMU-CS-02-111, Carnegie Mellon University, 2002.

[3] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings of the 9th International Conference on Computer Vision, Nice*, 2003.

[4] J. Neira, M. I. Ribeiro, and J. D. Tardos, "Mobile robot localization and map building using monocular vision," in *Proc. 5th Int. Symp. Intelligent Robotic Systems '97*, pp. 275–284, 1997.

[5] A. Cumani, S. Denasi, A. Guiducci, and G. Quaglia, "Integration of visual cues for mobile robot localization and map building," in *Measurement and Control in Robotics* (M. A. Armada, P. Gonzales de Santos, and S. Tachi, eds.), Instituto de Automatica Industrial, Madrid, 2003.

[6] D. Murray and J. Little, "Using real-time stereo vision for mobile robot navigation"," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1998.

[7] H. Hirschmuller, "Real-time map building from a stereo camera under unconstrained 3d motion," Faculty Research Conference, De Montfort University, Leicester, 2003.

[8] A. Cumani, S. Denasi, A. Guiducci, and G. Quaglia, "Robot localisation and mapping with stereo vision," *WSEAS Trans. Circuits and Systems*, vol. 3, no. 10, pp. 2116–2121, 2004.

[9] A. Cumani and A. Guiducci, "Robot localisation error reduction by stereo vision," *WSEAS Trans. Circuits and Systems*, vol. 4, no. 10, pp. 1239–1245, 2005.

[10] A. Cumani and A. Guiducci, "Stereo-based visual odometry for robust rover navigation," *WSEAS Trans. Circuits and Systems*, vol. 5, no. 10, pp. 1556–1562, 2006.

[11] A. Cumani and A. Guiducci, "Visual odometry with SURFs and déjàvu correction," in *Proc. 8th Conf. Optical 3D Measurement Techniques*, 2007.

[12] A. Cumani and A. Guiducci, "Fast point features for accurate visual odometry," in *Proc. 12th WSEAS CSCC Multiconference*, 2008.

[13] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proceedings 9th European Conference on Computer Vision*, 2006. See also `http://www.vision.ee.ethz.ch/~surf/`.

[14] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[15] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 511–518, 2001.

[16] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[18] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2nd ed., 2003.

[19] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image Vision Comput.*, vol. 22, no. 10, pp. 761–767, 2004.

[20] J. Y. Bouguet, "Complete camera calibration toolbox for MATLAB," tech. rep., `http://www.vision.caltech.edu/~bouguetj/calib_doc/index.html`, 2004.