Proceedings of the 2000 IEEE
International Conference on Robotics & Automation
San Francisco, CA • April 2000

Position Estimation in Outdoor Environments using Pixel Tracking and Stereovision

Anthony Mallet, Simon Lacroix
LAAS-CNRS
7 Av. du Colonel Roche
31077 Toulouse Cedex 4 - France
Anthony.Mallet, Simon.Lacroix@laas.fr

Laurent Gallo
Aerospatiale
2 Rue Béranger
92320 Chatillon - France
Laurent.Gallo@missiles.aeromatra.com

Abstract

This paper presents a method that estimates a robot displacements in outdoor unstructured terrain. It computes the displacements on the basis of associations of 3D points sets produced by consecutive stereovision frames, the associations being determined by tracking pixels from one image frame to the other. The paper details the various steps of the algorithms, and presents first experimental results are presented: they show that the algorithm is able to estimate the 6 parameters of the robot position with a relative error smaller than about 5%, processing several hundreds of images over several tens of meters.

1 Introduction

Future cross country robots will have to explore, map or traverse larger and larger areas. This is a tremendous challenge for roboticists, that must conceive systems endowed with *autonomous long range navigation* capacities. Indeed, several applications brings forth constraints, such as communication delays or the absence of precise knowledge on the environment, that void the possibility to efficiently teleoperate the machine (*e.g.* planetary exploration or intervention robotics).

At LAAS, we have tackled various aspects related to autonomous long range navigation in unstructured terrains for over ten years, and experimented some in realistic conditions [1, 2]. We are convinced that to efficiently achieve high level missions defined over a large scale of space and time, a certain degree of *deliberation* is necessary in order to anticipate events, take efficient decisions, and react adequately to unexpected events. In particular, this robot ability to *plan* its activities calls for the building of various environment representations, at several levels of abstraction (topological maps, geometric maps, object representations...). As a consequence, a good knowledge of the robot position is required for the purpose of coherent representation building, at least locally. A position estimate may also be required to ensure that the given mission is successfully being achieved, or to servo motions along a defined trajectory: robot self-localization is actually one of the most important issue to tackle autonomous navigation.

We present here an exteroceptive position estimation technique that is able to estimate the 6 parameters of the robot displacements in any kind of environments, provided it is textured enough so that pixel-based stereovision works well (thanks to progresses on cameras and algorithms, it is even the case for very smooth and flat terrains - the presence of no particular landmark is required). This technique is *passive*, in the sense that it does not calls for any data acquisition strategy: images are just processed as they are provided. The algorithms therefore do not interfere with any other functionality that makes use of the stereo cameras (obstacle avoidance, map building). The next section presents the principle of the method, and sections 3 to 5 describe the various steps of the motion estimation. Section 6 gives first experimental results obtained with the Marsokhod robot Lama, and a discussion concludes the paper.

2 Principle of the approach

The approach we developed and experimented could be called "exteroceptive dead-reckoning": it computes an estimate of the 6 displacement parameters between two stereo frames on the basis of a set of 3D point to 3D point matches, established by tracking the corresponding pixels in the image sequence acquired while the robot moves. Depending on the time spent by stereovision and on the number of pixels to track, the tracking phase lasts a variable number of frames, which can be reduced to one (which is the case for all

the results we present in this article).

The principle of the approach is extremely simple, but we paid a lot of attention to the selection of the pixel to track: in order to avoid wrong correspondences, one must make sure that they can be faithfully tracked, and in order to have a precise estimation of the motion, one must choose pixels whose corresponding 3D point is known with a good accuracy. Pixel selection is done in three steps: an *a priori* selection is done on the basis of the stereo images (section 3); an empirical model of the pixel tracking algorithm is used to discard the dubious pixels during the tracking phase (section 4); and finally an outlier rejection is performed when computing an estimate of displacement between two stereo frames (*a posteriori selection* - section 5).

Among the many contributions related to egomotion determination, a vast majority is devoted to the "motion and structure from motion" problem (see [3] for a review). Using multiple cameras off course reduces drastically the complexity of the egomotion determination, as it allows to recover to structure of the environment at each step, and numerous contributions can be found in the litterature (*e.g.* [4, 5, 6]).

3 Selection of the pixels to track

To initiate the process as a stereo frame comes up, one must select a set of pixels to be tracked. On one hand, one would like to track pixels whose corresponding 3D point is known with a good accuracy: this is done thanks to an error model of the pixel-based stereovision algorithm. On the other hand, one would like to select pixels that are likely to be successfully tracked in the forthcoming image sequence: this is done by studying the behavior of the auto-correlation function in the neighbor of the pixels of the image.

An error model for pixel correlation-based stereovision: A dense disparity image is produced from a pair of images thanks to a correlation-based pixel matching algorithm (we use the ZNCC correlation criteria[1]). False matches are avoided thanks to a reverse correlation and to various thresholds defined on the correlation score curve (essentially on the value of the highest score, and on the difference between this score and the second highest peak in the curve). To get quantitative informations on the precision of the computed disparity (and therefore on the coordinates of the 3D points), we studied a set of 100 images acquired from the same position. As in [8], it appeared that the distribution of the standard deviation on the disparity estimate can be well approximated by a Gaussian. A more interesting fact is that there is a *strong correlation* between the shape of the correlation curve around its peak and the standard deviation on the disparity: the sharper the peak, the more precise the disparity found. This correlation defines an *error model*, that is used during the pixel matching phase to estimate the error on the computed disparity (figure 1 - more details on this model can be found in [9]).

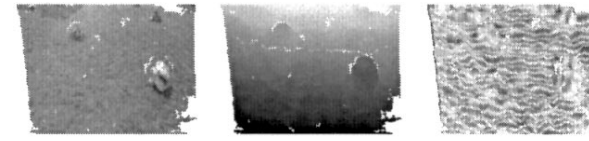

Figure 1: *A result of our stereovision algorithm: from left to right, original image (only correlated pixels are shown), disparity image, and standard deviation on the disparity estimated with our error model.*

However, there are matching errors that occur at the border between two regions of very different intensity values located at different depths (figure 2): as a consequence, the object shape in the disparity image is artificially growed of half the size of the correlation window. These errors, often referred to as "occluding contours artifacts" [10] can not be filtered out thanks to the thresholds on the correlation curve or to a blob filtering algorithm[2]. Moreover, their estimated error tend to be very small: it is practically impossible to avoid the selection of such pixels considering only the stereovision algorithm model.

Figure 2: *False matches at the border of a rock: disparity image (left), and correlated pixels (right).*

Selecting good candidates for the tracking algorithm: In textured environments, simple area-based matching techniques are extremely efficient to track pixels in an image sequence (see section 4). However, due to noise in the image and the sampling performed by cameras, the tracking algorithm often eventually drifts: after a few image frames, tracked pixels do not correspond to the same terrain points than the points corresponding to the original pixels. This off course occur especially on smooth, low textured areas, but can also occur on highly textured areas if they contain repetitive patterns, or along contours in the image. Therefore, checking a simple threshold on the

[1] Note that using the Hamming distance computed on Census transformed images [7] as a similarity measure give results as good as the ZNCC criteria.

[2] Note that the use a matching score based on census images reduces considerably these artifacts.

standard deviation on the grey levels of the correlation window is not sufficient to ensure that a pixel will be successfully tracked.

To avoid the selection of pixels in the image that are likely to drift during the tracking phase, we defined a measure other the image that represents how similar is a pixel to its neighbors. The general principle is to use a measure that is determined by the way the pixels will be tracked: this measure is based on the computation of the correlation score of one pixel with each of its neighbors, using the same correlation score and window size as the tracking algorithm (auto-correlation). These scores define a correlation peak (a surface), and the shape of this peak indicates how different is one pixel from its neighbors: the sharper the peak, the more different are the neighbors from the pixel. We use the greatest value of the correlation scores found for the neighbors as an indicator of the sharpness of the peak (see figure 3).

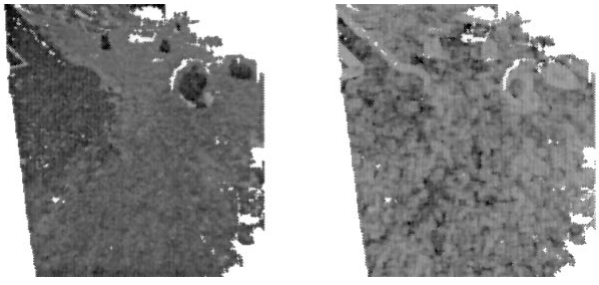

Figure 3: *Local similarity measure computed over a whole image. Left: original image, right: similarity measure encoded as grey levels. The darker pixels are good candidates for the tracking algorithm. One can note that the pixels corresponding to occluding contours are not good candidates for the tracking algorithm: indeed, in the direction defined by the contour, the correlation windows are very similar. Fortunately, this allow to discards all the occluding contour artifacts produced by stereovision.*

Note that this measure gives an indicator related to the expected *precision* of the tracking algorithm for a pixel, but not related to the ambiguity (*certainty*): to evaluate an ambiguity measure would require the computation of correlation scores for a wide neighborhood, which is extremely time consuming. The next section describes how the ambiguities are avoided by focusing the search area.

Pixels selection: The set of candidate pixels to track is defined by applying thresholds on the depth standard deviation estimate of the 3D points and on the corresponding pixel similarity measure. The pixels that will actually be tracked are then randomly chosen among the remaining candidates.

4 Tracking pixels in an image sequence

Although the pixels to track have been carefully selected, some errors (drifts or false matches) can occur during the tracking phase. In order to avoid such errors, we tested various matching criteria (SSD, ZNCC, Census...) and various template updating strategies on several image sequences to determine the best ones. We have not performed systematic tests as in [11], but it appeared that ZNCC and Census worked much better than any other matching score on our images.

Thanks to stereo image sequences, we can detect when a tracking algorithm is drifting by tracking "stereo-corresponding" pixels in the two images, and by checking that after the tracking phase, the returned pixels are still corresponding in the new stereo pair. However, tracking in parallel pixels in a stereo pair takes twice the time to track pixels in one image. We therefore only used this possibility to study off-line the tracking algorithm with stereovision, in order to establish statistics on various tracking algorithms and with various correlation window sizes. This helped to determine the best matching score, template update strategy and optimal window size: we retained the ZNCC correlation score computed over a 9×9 window, and update the template by interpolating the target image around the sub-pixellic matching estimate and with the previous template. Moreover, these statistics allowed us to easily determine the threshold values on the maximum correlation score and on the difference between the second highest peak in the surface, thresholds under which the algorithm is suspected to drift or to return a false match.

The tracking phase is done as follows: given a set of pixel to track and their corresponding 3D points defined on the stereo frame T_0, the search zone in the image acquired at time T_1 is centered around their predicted position, using the transformation $Tr_{T_0 \to T_1}$ provided by the robot internal sensors. The size of the search zone is determined according to the uncertainty on the estimated transformation. This prediction is important: it helps to focus the match search in a small area, and therefore reduces the probability to return a false match. Figure 4 shows the result of tracking a set of pixels in two images acquired from two positions distant of about 0.1 meter. One can see that most pixels have been successfully tracked.

5 Estimation of the motions

At the date T_1, when a new stereo acquisition is performed, the pixels of the tracked set whose 3D coor-

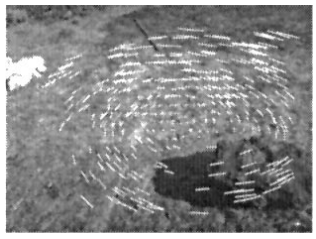

Figure 4: *Result of the tracking algorithm on a set of selected pixels. The lines display the displacement of every tracked pixel between two images.*

dinate estimate is now below a certain accuracy are discarded. We then partition the whole set of remaining matched 3D points into triplets, and estimate the 3D transformations defined by these triplets. Statistics are performed on the computed transformations in order to reject the possible remaining outliers: the mean and the standard deviation on the translation coordinates are computed, and triplets for which the difference between the corresponding translation and the mean translation is greater than 3 times the standard deviation are discarded (figure 5). The final estimate of the 3D transformation $T_{0\to 1}$ is performed using a least-square estimation method based on a singular value decomposition [12].

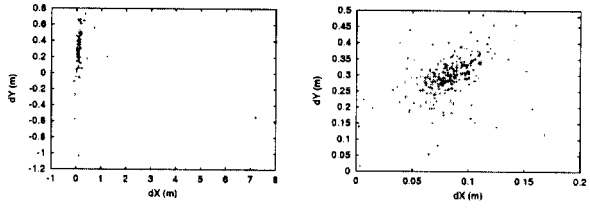

Figure 5: *A plot of the (dX, dY) translation estimates for a set of about 200 triplets of corresponding 3D points. The right figure is a close up of the left one.*

6 Experimental results

The whole motion estimation method runs on board the robot Lama (figure 6) and has been extensively tested over several thousand of images. It is part of a larger demonstration which integrates up to ten distinct functionalities (encapsulated in GenoM generated modules [13]). In particular, one of these functionality consists in probabilistic global obstacle map building: it strongly relies on the results of the motion estimation algorithm in order to merge the local maps. The robot motion is also servoed on the position given by the motion estimation method, as the goal to reach is specified in Cartesian coordinates.

Today, the main localization loop lasts about 7-8 seconds (image acquisition, stereo-correlation and motion estimation). Since we cannot deal well with movements larger than 50cm between two video frames (because the images must be very similar for the method to work), this drastically limits the speed of the robot to less than 5cm per seconds. Nevertheless, much optimization and rougher testing will be done and significantly reduce CPU requirements: we are convinced that this time can be reduced by a factor of ten on the same hardware (see section 7).

Figure 6: *The robot Lama taken by surprise while dealing with an obstacle. Lama is a Marsokhod model built by VNII Transmach, that belongs to Alcatel Space Industries. It has been equipped by LAAS with two stereo pair, a compass, 2 inclinometers, wheel encoders and on-board computers (two 300MHz power-pc and two 25MHz MC68040) on a VME rack operated by VxWorks.*

Figure 7 present the result of the method (called "Steo", which stands for "stereovision based odometry") for on a 25 meters long run. Much of the trajectory incorporated full $3D$ movements (running over rocks, as in figure 6). Because of such motions and a lot of rotations, the position integrated on the basis of the odometry quickly becomes more erroneous than the visual estimate. Qualitatively, Steo seems to give position estimates of about 4% (that is 1m for 25m in this example). This has been verified over several experiments, processing several hundreds of images each time. A more quantitative comparison for the same trajectory can be seen on figures 8 and 9. Steo computed position is always better than the position computed by odometry.

We can notice an interresting part, at approximately 15 meters from the starting point, where the overal slope of Steo's error increases suddendly (figure 8). This is due to a bad estimation between two images (which is then integrated over the time). Detection of such errors could noticeably improve position estimate. It is still to be noted that this comparison was achieved on a kind of soil where odometry works well (to preserve the significance of the comparision). A slipping soil would have obviously revealed a much more significant difference between odometry and Steo.

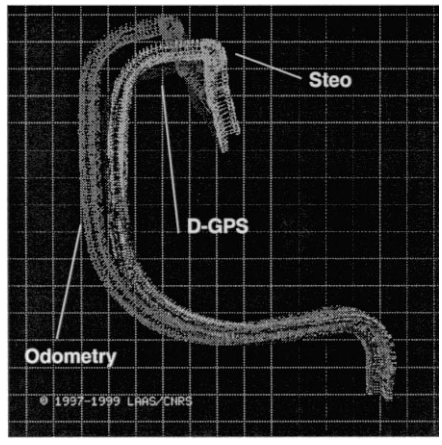

Figure 7: *Visual comparison between odometry, steo and D-GPS (top view of a 50 meters run, full 3D motion not shown). Grid cells and frame size are 1 meter large.*

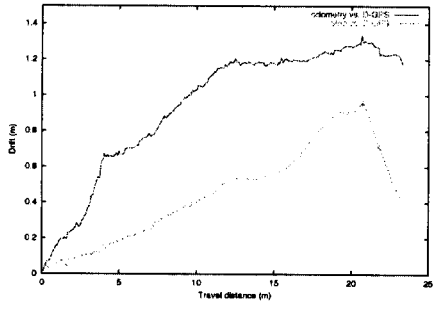

Figure 8: *Quantitative comparison between position computed by odometry and position computed by steo thanks to a D-GPS system. One can see that steo miscomputed an elementary motion at approximately 15 meters from the starting point. Fusion of odometry and steo could have avoided this.*

7 Discussion

We think the method has proven its feasibility and its efficiency, but however there are several improvements that must be evaluated. As an example, here is a list of the most important parameters we use, some comments about the choices we did and an estimated percentage of the time implied in the whole localization loop for the corresponding step of the algorithm. We choosed these values on the basis of experimentations only: the choices we did appear to work quite well, but future work will explore in details the influence of all these parameters, both from the point of view of computation time and precision of the results.

- **Camera positioning**: A parameter which does *not* seem to be important is the aiming of the pan-tilt unit regarding the direction of the motion. This a good point since it make the method a pssive one.

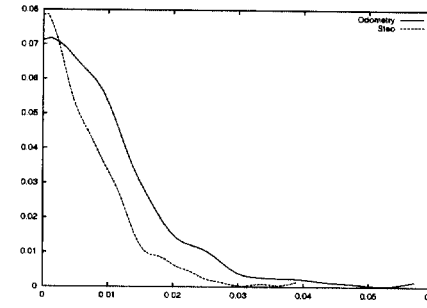

Figure 9: *Normalized histogram representing the error made locally between two images and averaged over 250 images. It somewhat shows the probability (Y-coordinate) that a given error (X-coordinate) occurs.*

- **Video images size**: cameras produce 768×576 images. We apply a reduction factor of 3, leading to 256×192 images.
- **$3D$ images size**: stereo-correlation is also performed on 256×192 images, leading to a 3D disparity map about 99% full. Disparity computation and 3D image building lasts about 15% of the overall loop time.
- **Auto-correlation template size**: the autocorrelation step (figure 3) is performed with the same 9×9 template size as the tracking phase (below). This step is very slow and is accomplished only on pixels that present a sufficiently good $3D$ precision (see section 3). The pixel selection step still lasts for 20% of the overall time (an optimized version of the algorithm, such as ussually performed in stereovision, should reduced this time by a *huge* factor).
- **Pixel template size**: for pixel correlation between successive images, we use a 9×9 template region for each pixel. Depending on the cameras quality, 7×7 or smaller templates may work well too: this would considerably reduce the time consumed by the pixel tracking phase.
- **Search zone size**: we search the pixels in an upcoming image in a 41×41 region centered around the predicted pixel new position (the prediction is done thanks to the odometry - see section 4). This is pretty big and consumes a lot of time, but for close pixels there can be errors as big as this in the odometry prediction. We need to integrate a true $3D$ odometry, using the inclinometers, and check whether the search zone size can be improved in any way.
- **number of pixels tracked**: we track 500 pixels between each images, and sometimes loose up to

70% of them. Much work is still to be done to reduce the lossage percentage, which would allow to reduce the number of *a priori* selected pixels. The tracking step lasts about 20% of the loop time: it depends on *i)* the number of pixels tracked, *ii)* the size of the templates and *iii)* the size of the search zone. It obviously can (and will...) be considerably speeded up.

Besides these various parameters settings, some methodological points are to be evaluated :

- **Tracking pixels over several stereo frames**: The obvious drawback of computing elementary motions only between two consecutive stereo frames is that the errors on the motion estimation cumulates over time, just as it happens when integrating the data of the robot's internal sensor. One way to reduce this errors is to use the possibility to track some pixels over several stereo frames: it allows to determine various displacements parameters every time a stereo image comes up.
- **Dealing with dynamic environments**: An other important issue to study is the possibility to apply this technique on dynamic environments, with moving parts such as vegetation in windy conditions, but also other robots or working people. If applied as is in such conditions, the algorithms would return totally erroneous motion estimates. This problem calls for the segmentation of the optical flow computed on selected pixels before computing the 3D transformation.

However precise this motion estimation technique can be, it is anyway not sufficient to tackle long range navigation. Indeed, as we stated in the introduction, the development of several different data processing and environment modeling algorithms is required to tackle the localization problem. All these algorithms are complementary, and provide position estimates with different characteristics: in particular, a model of each of these algorithms is required in order to filter the various position estimates into a consistent one, and to plan or trigger their activation. A better qualification of our technique is required on order to know the uncertainties on the position estimates it evaluates.

References

[1] R. Chatila and S. Lacroix. A case study in machine intelligence: Adaptive autonomous space rovers. In A. Zelinsky, editor, *Field and service Robotics*, number XI in Lecture Notes in Control and Information Science. Springer, July 1998.

[2] R. Chatila, S. Lacroix, T. Siméon, and M. Herrb. Planetary exploration by a mobile robot : Mission teleprogramming and autonomous navigation. *Autonomous Robots Journal*, 2(4):333–344, 1995.

[3] T.S. Huang and A. N. Netravalli. Motion and structure from feature correspondences: A review. *Proceedings of the IEEE*, 82(2):252–258, Feb. 1994.

[4] A-T. Tsao, Y-P. Hung, C-S. Fuh, and Y-S. Chen. Ego-motion estimation using optical flow fields observed from multiple cameras. In *IEEE Conference on Computer Vision and Pattern Recognition, San Juan (Porto Rico)*, pages 457–462. National Taiwan univ., June 1999.

[5] P.K. Ho and R. Chung. Stereo-motion that complements stereo and motion analyses. In *IEEE Conference on Computer Vision and Pattern Recognition, San Juan (Porto Rico)*, pages 213–218, June 1999.

[6] Z. Zhang and O. Faugeras. Estimation of displacements from two frames obtained from stereo. Technical Report RR-1440, INRIA, June 1991.

[7] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Third European Conference on Computer Vision, Stockholm (Sweden)*, May 1994.

[8] L. Matthies. Toward stochastic modeling of obstacle detectability in passive stereo range imagery. In *IEEE International Conference on Computer Vision and Pattern Recognition, Champaign, Illinois (USA)*, pages 765–768, 1992.

[9] S. Gautama, S. Lacroix, and M. Devy. On the performanece of stereo matching algorithms. In *Workshop on Vision, Modelling and Visualization, Erlangen (Germany)*, Nov. 1999.

[10] R. Sara and R. Bajcsy. On occluding contour artifacts in stereo vision. In *IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico*, pages 852–857, June 1997.

[11] J. Martin and J. Crowley. Comparison of correlation techniques. In *International Conference on Intelligent Autonmous Systems, Karlsruhe (Germany)*, pages 86–93, March 1995.

[12] R. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1426–1446, Nov/Dec 1989.

[13] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *Special Issue of the International Journal of Robotics Research on Integrated Architectures for Robot Control and Programming*, 17(4):315–337, April 1998. Rapport LAAS N97352, Septembre 1997, 46p.