INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Real time correlation-based stereo: algorithm, implementations and applications

Olivier Faugeras - Bernard Hotz - Hervé Mathieu - Thierry Viéville
Zhengyou Zhang - Pascal Fua - Eric Théron - Laurent Moll
Gérard Berry - Jean Vuillemin - Patrice Bertin - Catherine Proy

N° 2013
août 1993

PROGRAMME 4

Rapport de recherche

ISSN 0249-6399

Real time correlation-based stereo: algorithm, implementations and applications

Olivier Faugeras - Bernard Hotz - Hervé Mathieu - Thierry Viéville
Zhengyou Zhang - Pascal Fua - Eric Théron - Laurent Moll
Gérard Berry - Jean Vuillemin - Patrice Bertin - Catherine Proy

Programme 4 — Robotique, image et vision
Projet Robotvis

Rapport de recherche n° 2013 — août 1993 — 49 pages

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Abstract: This paper describes some of the work on stereo that has been going on at INRIA in the last four years. The work has concentrated on obtaining dense, accurate, and reliable range maps of the environment at rates compatible with the real-time constraints of such applications as the navigation of mobile vehicles in man-made or natural environments.

The class of algorithms which has been selected among several is the class of correlation-based stereo algorithms because they are the only ones that can produce sufficiently dense range maps with an algorithmic structure which lends itself nicely to fast implementations because of the simplicity of the underlying computation. We describe the various improvements that we have brought to the original idea, including validation and characterization of the quality of the matches, a recursive implementation of the score computation which makes the method independent of the size of the correlation window, and a calibration method which does not require the use of a calibration pattern.

We then describe two implementations of this algorithm on two very different pieces of hardware. The first implementation is on a board with four Digital Signal Processors designed jointly with Matra MSII. This implementation can produce 64×64 range maps at rates varying between 200 and 400 ms, depending upon the range of disparities. The second implementation is on a board developed by DEC-PRL and can perform the cross-correlation of two 256×256 images in 140 ms.

The first implementation has been integrated in the navigation system of the INRIA cart and used to correct for inertial and odometric errors in navigation experiments both indoors and outdoors on road. This is the first application of our correlation-based algorithm which is described in the paper. The second application has been done jointly with people from the french national space agency (CNES) to study the possibility of using stereo on a future planetary rover for the construction of Digital Elevation Maps.

We have shown that real time stereo is possible today at low-cost and can be applied in real applications. The algorithm that has been described is not the most sophisticated available but we have made it robust and reliable thanks to a number of improvements. Even though each of these improvements is not earth-shattering from the pure research point of view, altogether they have allowed us to go beyond a very important threshold. This threshold measures the difference between a program that runs in the laboratory on a few images and one that works continuously for hours on a sequence of stereo pairs and produces results at such rates and of such quality that they can be used to guide a real vehicle or to produce Discrete Elevation Maps. We believe that this threshold has only been reached in a very small number of cases.

(Résumé : tsvp)

Stéréoscopie en temps réel à base de corrélations : algorithme, implémentations et applications

Résumé : Cet article fait la synthèse de travaux réalisés dans le domaine de la stréréoscopie à l'INRIA ces quatre dernières années. Ces travaux se sont concentrés sur l'obtention de cartes de profondeur denses, précises et fiables à des cadences compatibles avec les contraintes du temps réel liées à des applications comme la navigation d'un véhicule dans un environnement artificiel ou naturel.

La classe d'algorithmes qui a été sélectionnée parmi les différentes méthodes possibles de stéréoscopie est celle des algorithmes à base de corrélations. Seules ces méthodes peuvent produire des cartes de profondeurs suffisamment denses à partir d'une structure algorithmique qui se prête facilement à des implémentations rapides de par sa simplicité intrinsèque. Nous décrivons différentes améliorations apportées au schéma initial, en particulier la validation et la caractérisation de la qualité des appariements, une implémentation récursive des scores de calcul qui rend la méthode indépendante de la taille de la fenêtre de corrélation, et une méthode de calibration qui ne nécessite pas l'utilisation d'une grille de calibration.

Nous décrivons ensuite deux implémentations de cet algorithme sur deux systèmes matériels très différents. La première implémentation a été réalisée sur une carte industrielle dotée de quatre processeurs de traitement du signal (DSP) réalisée conjointement par l'INRIA et Matra-MSII. Cette implémentation permet de calculer des cartes de profondeurs de taille 64×64 à des cadences variant entre 200 et 400 msec, selon l'intervalle de disparité requis. La seconde implémentation a été réalisée sur une carte développée par DEC-PRL et peut réaliser le calcul d'une carte d'intercorrélation de deux fois 256×256 en 140msec.

La première implémentation a été intégrée au système de navigation du robot mobile de l'INRIA et utilisée pour corriger les erreurs des capteurs inertiels et odométriques lors d'expérimentations de navigation autonomone en intérieur et en extérieur, sur une route. Ce travail constitue la première application de l'algorithme décrit dans ce papier. La seconde application a été réalisée conjointement avec une équipe de l'agence nationale française spatiale (CNES) pour étudier les possibilités d'utilisation de la stéréoscopie lors de futures explorations planétaires nécessitant la construction de cartes numériques de terrain.

Nous avons ainsi démontré qu'il est possible aujourd'hui de réaliser à faible coût un module de vision stéréoscopique temps-réel et utilisable au sein d'applications effectives. Cet algorithme n'est pas le plus sophistiqué des méthodes actuellement disponibles, mais il a été rendu robuste et fiable grâce à de nombreuses améliorations. Bien que chacune de ces améliorations ne soient pas vraiment fondamentale au niveau théorique, ajoutées les unes aux autres, elles permettent de dépasser une frontière rédibitoire. Cette frontière est celle qui sépare les programmes qui tournent en laboratoire sur quelques images et ceux qui travaillent de manière continue des heures durant sur des séquences de paires stéréoscopiques et produisent des résultats à des cadences et à des niveaux de qualité tels que l'on peut les utiliser pour guider un robot mobile sur sa trajectoire ou produire un modèle numérique de terrain. Nous croyons que cette frontière n'est franchie encore qu'assez rarement.

Real time correlation-based stereo: algorithm, implementations and applications

Olivier Faugeras Bernard Hotz Hervé Mathieu
Thierry Viéville Zhengyou Zhang
INRIA
2004 route des Lucioles, B.P. 93, 06902 Sophia-Antipolis cedex FRANCE
, **Pascal Fua**
SRI International
AI Center - 333 Ravenswool
Menlo Park, 94025 CA USA
fua@ai.sri.com
, **Eric Théron**
MS2i
Les Quadrants - 3 av du Centre
78182 Saint Quentin en Yvelines Cedex FRANCE
, **Laurent Moll**
Ecole Polytechnique
91128 Palaiseau Cedex FRANCE
moll@cma.cma.fr
, **Gérard Berry**
Ecole des Mines
Bld Albert Einstein
Sophia Antipolis FRANCE
berry@cma.cma.fr
, **Jean Vuillemin Patrice Bertin**
DEC-PRL
85 avenue Victor Hugo
92563 Rueil Malmaison cedex FRANCE
vuillemin@prl.dec.com
, **Catherine Proy**
CNES TOULOUSE
18, avenue Emile Belin
31055 Toulouse FRANCE
proy@hathor.cnes.fr

Contents

List of Figures

1 Introduction

Dense depth maps can be obtained from stereo at relatively high rates and resolutions using simple algorithms and hardware. They are useful for automatic land vehicles navigation in particular but in general for any task requiring a dense three-dimensional representation of three space. This article presents a summary of the work that has been going on at INRIA on the construction and use of dense three-dimensional maps from stereo. This work includes algorithmic development and coding on standard architectures, the porting of the algorithm on two very different parallel architectures, the use of one of these architectures in a robot navigation application indoors and on roads, and the construction of Digital Elevation Models (DEMs) for the navigation of a rover for future planetary exploration.

The literature on stereo is very large and we will not attempt to review it here. Let us just briefly recall the fact that the main difficulty in stereo vision is to establish correspondences between pairs (triplets, etc...) of images. Trying all possible correspondences is still out of the possibilities of current computers because of the combinatorial explosion problem and people have been using constraints to reduce it. These constraints are basically of three kinds:

1. Geometric constraints imposed by the imaging system: probably the most important such constraint is the epipolar constraint thanks to which we can transform a two-dimensional search for correspondence into a one-dimensional one.

2. Geometric constraints arising from the objects being looked at: we can assume, for example that their distance to the imaging system varies slowly almost everywhere. This is, for example, the origin of the disparity gradient constraint.

3. Physical constraints such as those arising from models of the way objects interact with the illumination. The simplest and most widely used such model is the Lambertian model [15].

Stereo algorithms also differ according to the type of tokens they attempt to match and to the type of features they use to represent them. Almost all possibilities of tokens have been proposed, from the single pixel, the edge pixel, curves of various sorts up to image regions. For each of these tokens numerous features have been used, the limit being set by the researchers' imagination.

The algorithms for stereo fall broadly into four main categories

Correlation-based algorithms : intensity based area-correlation techniques have been investigated extensively for commercial applications in stereo-photogrammetry [17, 9] but are also one of the oldest methods used in computer vision [12, 25]. A recent application of this class of techniques to Planetary rovers can be found in [23].

Relaxation-based algorithms : The basic idea of this class of techniques is to allow the pixels that are to be put into correspondence make "educated guesses" as to what their match should be and then let the matches reorganize themselves by propagating some of the above constraints. Three famous examples of this class of algorithms are the Marr-Poggio algorithm [20, 21], the Grimson algorithm [13, 14], and the Pollard-Mayhew-Frisby algorithm [27].

Dynamic programming : The problem of matching primitives between images can also be cast as a problem of minimizing a cost function. Dynamic programming is a way of efficiently minimizing (or maximizing) functions of a large number of discrete variables. Successful attempts at using dynamic programming for solving the stereo matching problem are those of Baker and Binford [4], and Ohta and Kanade [26]. In both cases, they were using edges as the basic primitives.

Prediction and verification : This is a category of stereo algorithms where the tokens put into correspondence are of a higher symbolic level than pixels. This approach has been followed in particular by Medioni and Nevatia [24], Ayache and Faverjon [2], Ayache and Lustman [3], Yachida [18], and Robert [28], among others.

For a recent review, the reader is referred to [5].

Since we were interested for our applications in obtaining dense three-dimensional maps at a reasonable computational cost, this oriented us from the beginning toward the first category. The algorithm described in this paper is a variant of this class of techniques.

We started working on the algorithm in 1990 [10]. This original version was improved later and ported on a DSP board in collaboration with Matra-MSII and on a XLinks board by Molle, Berry, and Vuillemin. These hardware implementations have been used in several navigation and exploration tasks of which we describe two. The first application is the navigation of the INRIA mobile cart indoors and on the road using a combination of three sensory processes: odometry, inertia, and correlation based stereo. In this application the stereo is used in a very simple manner to localize the edges of the hallway for the indoors part, and of the road for the outdoors part. The second application is the construction of Digital Elevation Models (DEMs) for the navigation of a rover in a rocky environment. The DEMs are intended to help planning trajectories and are built incrementally by fusing local 3-D representations obtained by the stereo rig mounted on the rover from several viewpoints.

The paper is organized as follows. In section 2 we recall briefly the original algorithm described in [10, 11] and the latest improvements. In section 3 we describe the first hardware parallel implementation on a board composed of four DSP's from Motorolla and present its performances. In section 4 we describe the second hardware implementation on the DECPeRLe-1 board and present its performances. In section 5 we present the indoors and outdoors navigation experiments, in section 6 we present the DEM's construction experiment.

2 Description of the algorithm

The algorithm that we are about to describe falls in the category of the correlation based stereo algorithms. The tokens which are used in the correspondence process are the image pixels themselves with one feature, the intensity at the pixel. The algorithm uses two geometric constraints, the epipolar constraint to reduce the search for correspondences, and the constraint that the disparity (or depth) is locally constant in the vicinity of a pixel. The second constraint is clearly only an approximation. The algorithms also uses the constraint that the intensities at two corresponding pixels are approximately the same. This is in fact a physical constraint related to the hypothesis that the observed objects are approximately lambertian.

2.1 Calibration and rectification

Calibration of a stereo rig is necessary for using the epipolar constraint and for three-dimensional reconstruction.

2.1.1 Camera sytem calibration

Calibration of a stereo camera system requires a calculation of intrinsic and extrinsic parameters for both cameras. This amounts to finding a relationship (perspective projection) between the 3-D points in the scene and their different camera images [8, 29, 30]. This is a crucial stage in the vision process as it allows to simplify the correspondence problem and to obtain a 3-D representation of the results. Calibration has been performed in two ways. Initially it was performed using images of a predefined calibration grid. Specific points were extracted to obtain a sufficient number of combinations (pixel, 3-D point) to compute the perspective projection matrices. Using a calibration grid might be problematic for some applications. Therefore we also used the so called “weak” calibration technique [7, 19] for the Mars rover system. Weak calibration means that only the epipolar geometry of the stereo rig is known, but not the intrinsic parameters of the cameras. This only requires the knowledge of a small number of pixel correspondences between images. It does not require either the knowledge of any three-dimensional information or the use of a special calibration grid. One drawback of this calibration scheme is that 3D reconstruction is done in an unknown affine or projective frame [6]. Hence, adjustment against a metric frame is compulsory before exploiting results, at least at the present stage of our knowledge.

2.1.2 Epipolar geometry simplification: Image rectification

The pinhole camera model implies that the correspondent of a given point lies on its epipolar line in the other image. Corresponding points can then be found by scanning every point's epipolar lines. Unfortunately the epipolar lines form a pencil of lines, i.e. they all go through a point called the epipole. This makes the scanning task fairly complex and thus inefficient. This is why the images are reprojected onto a plane parallel to the line between the optical centers, in the case of a binocular stereo rig, or to the optical centers plane for a trinocular system. In the case of a binocular rig, we can align the epipolar lines with the image rows which greatly simplifies the correspondence process. In the case of a trinocular rig, an L camera setup allows to align the epipolar lines with the image rows or columns and to reduce the image deformations due to rectification with no perturbing effects on the following pairing stage.

2.2 Matching algorithm

A number of correlation-based algorithms attempt to find points of interest on which to perform the correlation. This approach is justified when only limited computing resources are available, but with modern hardware architectures it becomes practical to perform the correlation over all image points and retain only matches that appear to be "valid". The hard problem is then to provide an effective definition of what we call validity and we will propose one below.

Correlation scores are computed by comparing a fixed window in the first image to a shifting window in the second. The second window is moved in the second image by integer increments along the corresponding epipolar line and a curve of correlation scores is generated for integer disparity values. The mesured disparity can then be taken to be the one that provides the largest peak. To compute the disparity with subpixel accuracy, we fit a second degree curve to the correlation scores in the neighborhood of the extremum and compute the optimal disparity by interpolation.

2.2.1 Correlation criteria

To quantify the similarity between two correlation windows, we must choose among many different criteria the one that produces reliable results in a minimum computation time. We denote by $I_1(x,y)$ and $I_2(x,y)$ the intensity values at pixel (x,y). The correlation window has dimensions $(2n+1)\times(2m+1)$. Therefore, the indexes which appear in the formula below vary between $-n$ and $+n$ for the i-index and between $-m$ and $+m$ for the j-index. We have extensively tested the four criteria defined below (we take the case of horizontal epipolar lines with the same image row index, so that we have no y- or vertical disparity):

$$C_1(x,y,d) = \frac{\sum_{i,j}[I_1(x+i,y+j) - I_2(x+d+i,y+j)]^2}{\sqrt{\sum_{i,j} I_1(x+i,y+j)^2} \times \sqrt{\sum_{i,j} I_2(x+d+i,y+j)^2}}$$

$$C_2(x,y,d) = \frac{\sum_{i,j} I_1(x+i,y+j) \times I_2(x+d+i,y+j)}{\sqrt{\sum_{i,j} I_1(x+i,y+j)^2} \times \sqrt{\sum_{i,j} I_2(x+d+i,y+j)^2}}$$

$$C_3(x,y,d) = \frac{\sum_{i,j}[(I_1(x+i,y+j) - \overline{I_1(x,y)}) - (I_2(x+d+i,y+j) - \overline{I_2(x+d,y)})]^2}{\sqrt{\sum_{i,j}[I_1(x+i,y+j) - \overline{I_1(x,y)}]^2} \times \sqrt{\sum_{i,j}[I_2(x+d+i,y+j) - \overline{I_2(x+d,y)}]^2}}$$

$$C_4(x,y,d) = \frac{\sum_{i,j}[I_1(x+i,y+j) - \overline{I_1(x,y)}] \times [I_2(x+d+i,y+j) - \overline{I_2(x+d,y)}]}{\sqrt{\sum_{i,j}[I_1(x+i,y+j) - \overline{I_1(x,y)}]^2} \times \sqrt{\sum_{i,j}[I_2(x+d+i,y+j) - \overline{I_2(x+d,y)}]^2}}$$

where $\overline{I_k(x,y)}$, $k=1,2$ is the average of image k within the window of size $(2n+1)\times(2m+1)$. These four criteria can be expressed compactly in vector form if we consider the vectors of size $(2n+1)(2m+1)\times 1$ obtained by stacking the columns of the image windows. With obvious notations, we can rewrite them as follows:

$$C_1(x,y,d) = \frac{\|\mathbf{I}_1(x,y)-\mathbf{I}_2(x+d,y)\|^2}{\|\mathbf{I}_1(x,y)\|\cdot\|\mathbf{I}_2(x+d,y)\|} \quad C_2(x,y,d) = \frac{\mathbf{I}_1(x,y)\cdot\mathbf{I}_2(x+d,y)}{\|\mathbf{I}_1(x,y)\|\cdot\|\mathbf{I}_2(x+d,y)\|}$$

$$C_3(x,y,d) = \frac{\|\mathbf{J}_1(x,y)-\mathbf{J}_2(x+d,y)\|^2}{\|\mathbf{J}_1(x,y)\|\cdot\|\mathbf{J}_2(x+d,y)\|} \quad C_4(x,y,d) = \frac{\mathbf{J}_1(x,y)\cdot\mathbf{J}_2(x+d,y)}{\|\mathbf{J}_1(x,y)\|\cdot\|\mathbf{J}_2(x+d,y)\|}$$

The C_1 and C_3 criteria use the difference between the gray levels of the images and must be minimized with respect to the disparity d. The C_2 and C_4 criteria multiply the gray level values together and must be maximized with respect to d. C_3 and C_4 are similar to C_1 and C_2 respectively, except for the fact that the mean gray level value over the correlation window is substracted from the intensity values. From the expressions of C_i, $i=1,\cdots,4$, it is apparent that the value of C_1 does not change if we replace the two images I_1 and I_2 with aI_1+b and aI_2+b with the *same* values of a and b, that the value of C_2 does not change if we replace I_1 by aI_1 and I_2 by bI_2, that the value of C_3 does not change if we replace I_1 by aI_1+b_1 and I_2 by aI_2+b_2, and that the value of C_4 does not change if we replace I_1 by $a_1I_1+b_1$ and I_2 by $a_2I_2+b_2$. We have found C_3 and C_4 performed best in practice because they are the most invariant to affine transformations of the images which may result from slightly different settings of the cameras. C_2 has similar performances to C_3 and C_4 except when the difference in the distribution of gray levels between the images is important. C_1 clearly produces worse results than the others criteria.

2.2.2 Validating matches

As shown by Nishihara [25], the probability of a mismatch goes down as the size of the correlation window and the amount of texture increase. However, using large windows leads to a loss of accuracy and to the possible missing of important image features. For smaller windows, the simplest definition of validity would call for a threshold on the correlation score; unfortunately such a threshold would be rather arbitrary and, in practice, hard to choose. Another approach is to build a correlation surface by computing disparity scores for points in the neighborhood of a prospective match and checking that the surface is peaked enough [1]. It is more robust but also involves a set of relatively arbitrary thresholds.

Here we propose a definition of a valid disparity measure [11] in which the two images play a symmetric role and that allows us to greatly reduce the probability of error even when using very small windows. We perform the correlation twice by reversing the roles of the two images and consider as valid only those matches for which the reverse correlation has fallen on the initial point in the left image.

This validity test is likely to work in the presence of an occlusion. Indeed, let us assume that a portion of a scene is visible in the left image I_1 but not in the right image I_2. The pixels in I_1 corresponding to the occluded area in I_2 will be matched, more or less at random, to points of I_2.. The reverse correlation will usually find better matches in I_1 for those points. The matches for the occluded points will therefore be declared invalid and rejected.

In fact, the density of such consistent matches in a given area of the image appears to be an excellent indicator of the quality of the stereo matching. An occasional "false positive" (a pixel for which the same erroneous disparity is measured when matching both from left to right and right to left) may occur. But, except in the presence of repetitive patterns, we have never encountered a situation that gave rise to a large clump of such errors.

When the correlation between the two images of a stereo pair is degraded our algorithm tends, instead of making mistakes, to yield sparse maps. In other words, a relatively dense disparity map is a *guarantee* that the matches are correct, at least up to the precision allowed by the resolution being used. If we reject not only invalid matches but also isolated valid matches (using a simple method based on successive erosions and dilatations) we can increase even more the ratio correct/incorrect matches without losing a large number of the correct answers.

2.2.3 More information about matches

It is very interesting to really know if a validated match is reliable or not. If the match is situated in a dense area in the disparity map, the probability of a correct correlation is very high, except on repetitive patterns. But the form of the correlation curve (criterion value for all integer disparity values) can be used to decide if the probability of the match to be an error is high or not. Indeed, errors occur when a wrong peak slightly higher than the right one is chosen. So, if in the correlation curve we can notice the presence of several peaks with approximately the same height, the risk of choosing the wrong one increases, especially if the images are noisy. We have therefore defined a confidence coefficient proportionnal to the difference of height between the two most important peaks (which must be sufficiently distant when using small windows for which the correlation curve has a lot of small noisy peaks). On repetitive patterns the correlation curve has a periodic-like shape and the confidence will receive a very low value.

We can extract another type of information from the correlation curve. Indeed, the shape of the optimal peak shows us if the matched points are situated in bland areas or not. The narrower the peak is, the more precise the localisation of the matched point is. So, a good way to quantify the accuracy of the sub-pixel disparity computed by the parabolic approximation is to measure the spread of the optimal peak. In order to do this, we assume that the peak can be locally represented as a gaussian with standard deviation σ and take the sub-pixel precision proportionnal to σ.

2.2.4 Hierarchical algorithm

To increase the density of our potentially sparse disparity map, we use windows of a fixed size to perform the matching at several levels of resolution (computed by subsampling gaussian smoothed images), which is almost equivalent to, but computationally more efficient, than matching at one level of resolution with windows of different sizes More precisely, it amounts to performing the correlation using several frequency bands of the image signal.

We then merge the disparity maps by selecting, for every pixel, the highest level of resolution for which a valid disparity has been found. The reliability of our validity test allows us to deal very simply with several resolutions without having to introduce, a correction factor accounting for the fact that correlation scores for large windows tend to be inferior to those for small windows.

The computation proceeds independently at all levels of resolution and this is a departure from traditional hierarchical implementations that make use of the results generated at low resolution to guide the search at higher resolutions. While this is a good method to reduce computation time, it assumes that the results generated at low resolution are more reliable, if less precise, than those generated at high resolution; this is a questionable assumption especially in the presence of occlusions. For example in the case of tree images, it could lead to a computed distance for the area between some neighbouring trunks that would be approximately the same as that of the trunks themselves, which would be wrong. Furthermore, in the absence of repetitive patterns, the output of our algorithm is not appreciably degraded by using the large disparity ranges that our approach requires.

2.2.5 Trinocular algorithm

As suggested by several researchers [3, 18, 28], more than two images can and should be used whenever practical. When dealing with three images or more, we take the first one to be our reference frame, compute separately disparity maps for all pairs formed by this image and one of the others and merge these maps by selecting for example the match that has produced the best correlation score. In this way we can reduce the size of occluded areas and generate a denser depth map.

In particular, we use at INRIA a trinocular stereo system in which the cameras are situated on the vertices of a right-angle triangle. This particular disposition is needed to cause small deformations of the images in the rectification process. Indeed, to simplify the research of the best candidate along the epipolar line, the images are first reprojected onto the same image plane (parallel to the plane defined by the three optical centers) so that all epipolar lines become parallel. It is easy to show that the axes of the rectified images referentials can be chosen in such a way as to make the epipolar lines horizontal on the same line or vertical on the same column, without introducing severe deformations. The image from the camera located at the square corner of the right-angle triangle is taken to be the reference one. This reference image is then correlated horizontally with the second image and vertically with the third. The two disparity maps obtained are then merged together to produce a dense fused depth map.

Correlating both horizontally and vertically has the advantage to introduce redundancies which allow us to obtain more reliable results. Indeed we can verify that two matches are compatible by testing if the matched points in the second and third images lie on the same diagonal epipolar line. If this condition is true, then the two matches measure the same depth and we can be almost absolutely sure they are correct because the search of the two corresponding points is made on completely different areas horizontally and vertically. If not, at least one match is false, and we have to select the best one. We have tested this trinocular algorithm on outdoor rocks scenes and have noticed that such a simple philosophy can build error-free depth maps.

2.2.6 Algorithmic implementation

In this paragraph we restrict our attention to a binocular correlation aligned horizontal epipolar lines, so that we have no verical disparities. Let us consider criterion C_2 for the moment. A first simplification is obtained by noticing that the first term in the denominator is constant when x and y do not vary. We can therefore consider the simpler criterion

$$C_2'(x,y,d) = \frac{\sum_{i,j} I_1(x+i,y+j) \times I_2(x+d+i,y+j)}{\sqrt{\sum_{i,j} I_2(x+d+i,y+j)^2}}$$

For simplification purposes, we split the computation into different parts:

$$\begin{aligned}
O(x,y) &= \max_d\{N(x,y,d) \times R(x+d,y)\} \\
N(x,y,d) &= \sum_{i,j} I_1(x+i,y+j) \times I_2(x+d+i,y+j) \\
R(x,y) &= 1\sqrt{M(x,y)} \\
M(x,y) &= \sum_{i,j} I_2(x+i,y+j) \times I_2(x+i,y+j)
\end{aligned}$$

We can see that the numerator uses $(2n+1)(2m+1)$ redundant multiplications. Using recursions over the indices i and j, we can avoid redoing the same computation. We compute the numerator N as follows:

$$\begin{aligned}
P(x,y,d) &= I_1(x,y) \times I_2(x+d,y) \\
Q(x,0,d) &= \sum_j P(x,j,d) \\
Q(x,y+1,d) &= Q(x,y,d) + P(x,y+2m+1,d) - P(x,y,d) \\
N(0,y,d) &= \sum_i Q(0,y,d) \\
N(x+1,y,d) &= N(x,y,d) + Q(x+2n+1,y,d) - Q(x,y,d)
\end{aligned}$$

Similarly, for the denominator M:

$$
\begin{aligned}
P_2(x,y) &= I_2(x,y) \times I_2(x,y) \\
Q_2(x,0) &= \sum_j P_2(x,j) \\
Q_2(x,y+1) &= Q_2(x,y) + P_2(x,y+2m+1) - P_2(x,y) \\
M(0,y) &= \sum_i Q_2(0,y) \\
M(x+1,y) &= M(x,y) + Q_2(x+2n+1,y) - Q_2(x,y)
\end{aligned}
$$

The equations for R and O are as above. Those simplifications are represented graphically in figure 1. The computation of the correlation criterion for a given value of x and y is

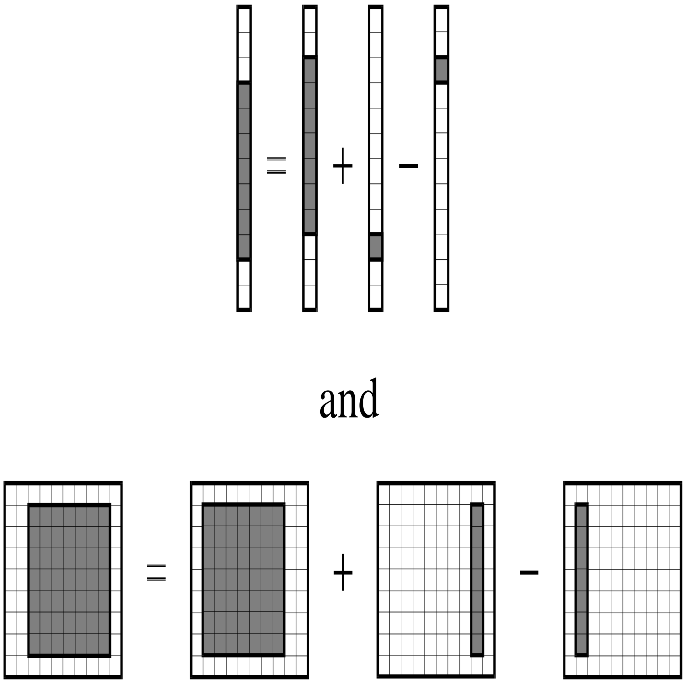


Figure 1: Simplifications to the computation of criterion C_2.

performed by first calculating and storing the first square-root term of the denominator. Then the cross-term of the numerator and the second square-root term of the denominator are computed for each disparity value and divided by the previous square root term to obtain the criterion values.

The use of these relationships allows us to avoid any redundancies in the criterion computation and makes the processing time independent of the window size. We can also take large windows to produce more reliable results without being penalized in time. There is no problem of roundness because the values I_1 and I_2 are integer. Similar ideas can be used for computing criterion C_1.

The criteria C_3 and C_4 cannot be computed exactly in the same way because the mean values $\overline{I_k}, k = 1, 2$ which are substracted are the same for all points of the correlation windows. If we first substract at every pixel the mean value (computed on a rectangular neighbourhood of the same size as the correlation window and centered on points) and then perform the correlation using the non-normalized criteria C_1 or C_2, it amounts to using the C_5 or C_6 criteria defined as below:

$$C_5(x,y,d) = \frac{\sum_{i,j}[(I_1(x+i,y+j) - \overline{I_1(x+i,y+j)}) - (I_2(x+d+i,y+j) - \overline{I_2(x+d+i,y+j)})]^2}{\sqrt{\sum_{i,j}[I_1(x+i,y+j) - \overline{I_1(x+i,y+j)}]^2} \times \sqrt{\sum_{i,j}[I_2(x+d+i,y+j) - \overline{I_2(x+d+i,y+j)}]^2}}$$

$$C_6(x,y,d) = \frac{\sum_{i,j}[I_1(x+i,y+j) - \overline{I_1(x+i,y+j)}] \times [I_2(x+d+i,y+j) - \overline{I_2(x+d+i,y+j)}]}{\sqrt{\sum_{i,j}[I_1(x+i,y+j) - \overline{I_1(x+i,y+j)}]^2} \times \sqrt{\sum_{i,j}[I_2(x+d+i,y+j) - \overline{I_2(x+d+i,y+j)}]^2}}$$

C_5 and C_6 produces almost the same results as C_3 and C_4, respectively; there is sometimes a little difference between scores curves but most of the time the criteria vary in the same way. The use of C_5 or C_6 allows us to have a robust normalized cross-correlation in a minimum computation time.

2.3 3D Reconstruction

By intersecting the optical rays of two matched pixels we reconstruct the corresponding 3-D point. This is done by inverting a 3×3 matrix. Prior knowledge of the epipolar position uncertainty and disparity precision is used to compute a 3-D covariance matrix for every reconstructed point.

When weak calibration is used, 3-D reconstruction is performed in a projective frame. As of today, the reconstruction results can only be used for applications in a metric frame of reference which, in the case of the Mars rover, must be obtained without the help of a calibration grid. This can be done if the robot features a calibrated laser vision system providing metric information. Switching from the projective frame to a metric one can be done after computing the projective transformation betweeen the two frames which can be achieved from five correspondences between laser points and matched image pixels. Figure 2 shows an original stereo pair of a rock scene used to test vision algorithms developed for the VAP project in Toulouse. Two corresponding epipolar lines are also shown. Figure 3 shows the rectified stereo pair after estimation of the epipolar geometry. Figure 4 shows on the left the depths (distances to the cameras) and on the right the elevations (distance from the ground). Both are represented in shades of gray, dark meaning close, white meaning far, and black meaning that the algorithm has returned "don't know". Finally, figure 5 shows two perspective representations of the 3-D reconstruction of the scene of figure 2: the left part shows the reconstruction obtained by calibrating the stereo rig traditionnally with a known three-dimensional calibration pattern, the right part shows the same reconstruction obtained by "weakly" calibrating the stereo rig, using only the images of a few points in the scene. The two reconstructions have then been put in the same metric frame for display purposes by using the 3-D metric coordinates of five points in the two views. The two reconstructions are clearly very similar, thus validating the weak calibration approach.

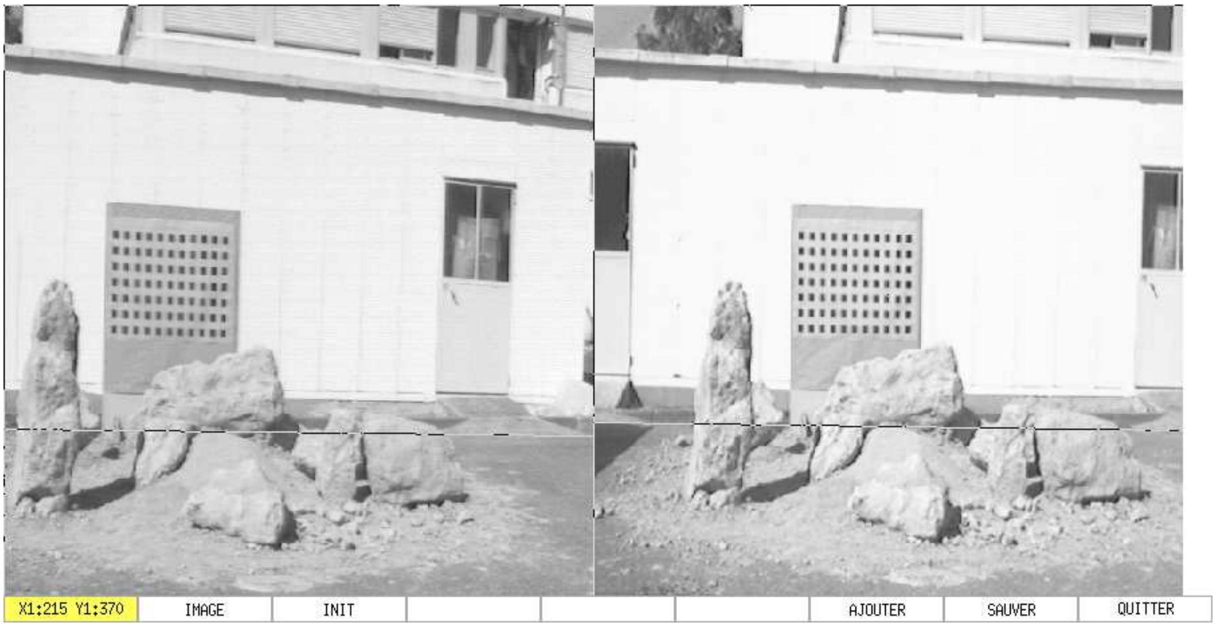


Figure 2: Original stereo pair

3 DSP board implementation

We now describe the first of the two parallel implementations of the stereo algorithm described in section 2. The idea of this implementation is to use in parallel several powerful processors. Each of this processors can be programmed in a fairly high-language such as C and the existence of cross-compilers available on most workstations has made the parallelization of the workstation version of the code relatively easy.

3.1 Architecture

DSPs (Digital Signal Processors) are well adapted for many computer vision algorithms because of their hardware architecture. This is why we developed the MD96 board, a Multi-

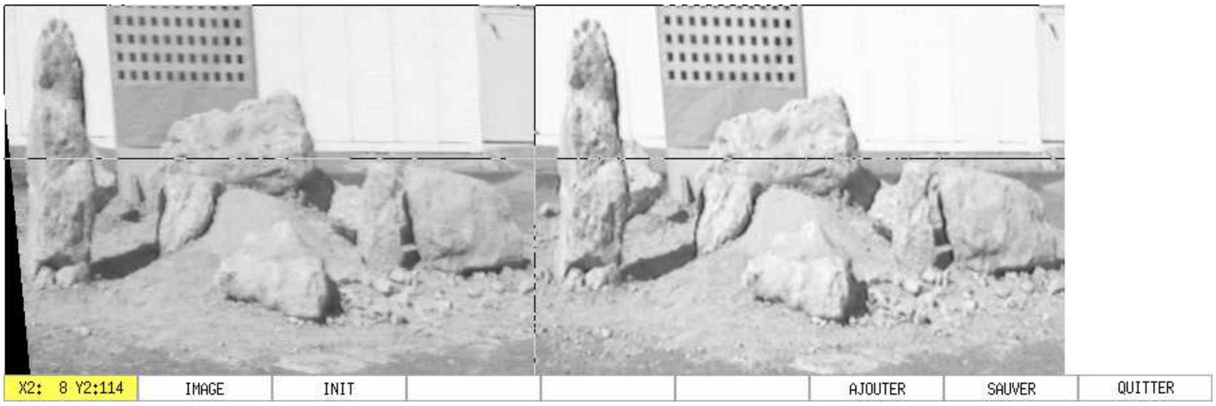


Figure 3: Rectified stereo pair

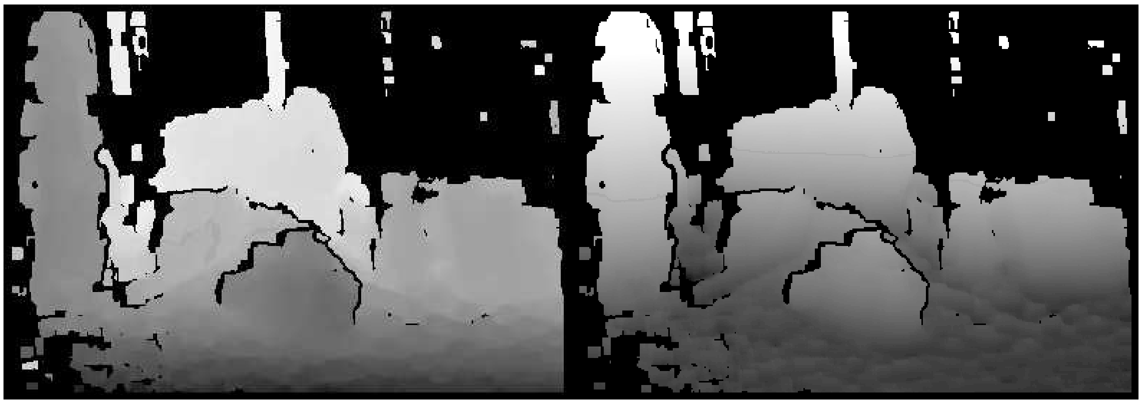


Figure 4: Depth and elevation

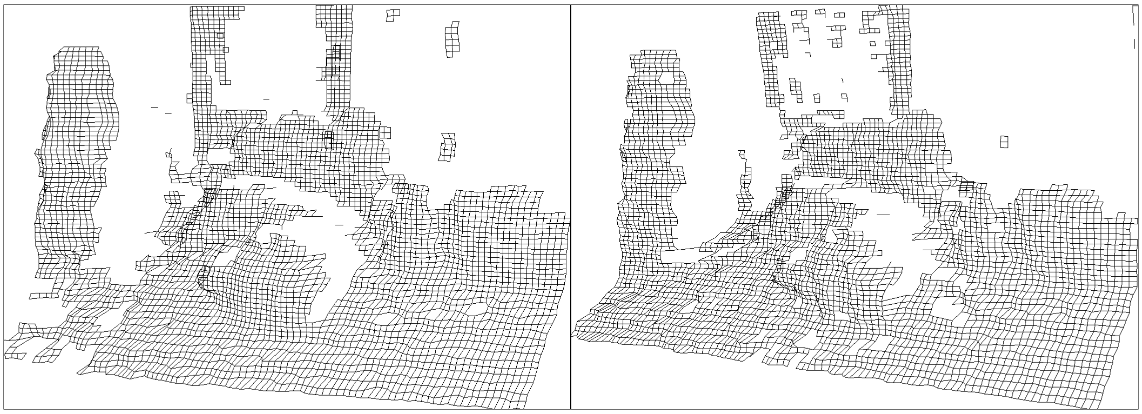


Figure 5: 3D models (grid calibration and self-calibration)

DSP board, with four Motorola 96002 Digital Signal Processors and interfaced with the VMEbus. This design was done in 1990 within an ESPRIT European project as a collaboration between Hervé Mathieu from INRIA and Eric Théron from Matra MSII. Detailed technical information about this board can be found in [22]. The MD96 board has a peak processing power of 240 MFLOPS (Mega FLoating-point Operation per Second) and has the following features:

- Four Processing Elements working in parallel. Each one of them is a 96002 Digital Signal Processor running at 40 MHz with two 256Kx32 (256 Kilo-words of 32 bits) memory modules packaged in JEDEC shape.
- An on-board shared bus between the four Processing Elements, the VMEbus and a 256Kx32 memory (Communication Memory). This shared bus allows Processing Elements to use fast communication channels between them and to share data through a communication memory.
- Each DSP can be master or slave on the VMEbus allowing the board to work without any master board. (except for booting). The VMEbus interfaced module is fully compliant with the VMEbus specification (Revision C.1).
- The MD96 is made of standard CMOS/TTL components, and is implemented on an extended Euro-Card (220 mm x 233 mm).

3.1.1 DSP96002 Overview

The DSP96002 is a dual-port IEEE floating-point programmable CMOS processor. The device is available with a 33 MHz (resp. 40 MHz) clock, for 16,5 (resp. 20) Million Instructions per second (MIPS) and 49.5 (resp. 60) MFLOPS.

The main features of the DSP96002 are :

- A 32x32 bits floating-point and integer multiplier unit.
- A 32/64 bits floating-point and integer ALU.
- A full 32 bits Address Generation Unit.
- 2Kx32 bits internal memory (in three banks).
- Two A32/D32 channels DMA controller.
- Full compatibility with IEEE 32/64 floating point and integer data format. This means that format conversions are not required. In fact, the internal device architecture has been designed for efficient C implementation.

The architecture of the MD96 board is shown in figure 6.

3.1.2 Software

The programming of the board can be done in a fairly high-level language such as C and compiled with the C Compiler Intertools delivered by Intermetrics Inc. A C library allows to drive the MD96 board via a host computer, and some C libraries have been written for the MD96, allowing communication between DSPs or between a DSP and other boards connected on the same VMEbus.

3.1.3 Conclusion

As a conclusion, the MD96 board accepts up to 9 mega-bytes of fast access memory, and all the programming can be done in C. Each DSP works on its memory with zero Wait State and no bus arbitration, and the shared bus is used as a communication channel by each DSP with one Wait State, or by the VMEbus with a minimum of arbitration.

3.2 Implementation

We now describe how the binocular correlation described in section 2 has been implemented on the MD96 board.

Our current configuration uses two MD96 boards which are embedded in a VMEbus box containing a Motorola MVME167 with a 68040 processor running the real-time operating system VxWorks from Wind River System Inc.. This box also includes a video frame grabber allowing to scan synchronously two 512×512 interlaced images.

Our implementation has four parts :

- Rectification in order to obtain horizontal epipolar lines.
- Binocular correlation programmed in floating-point to achieve sub-pixel accuracy. We have implemented criterion C_5.

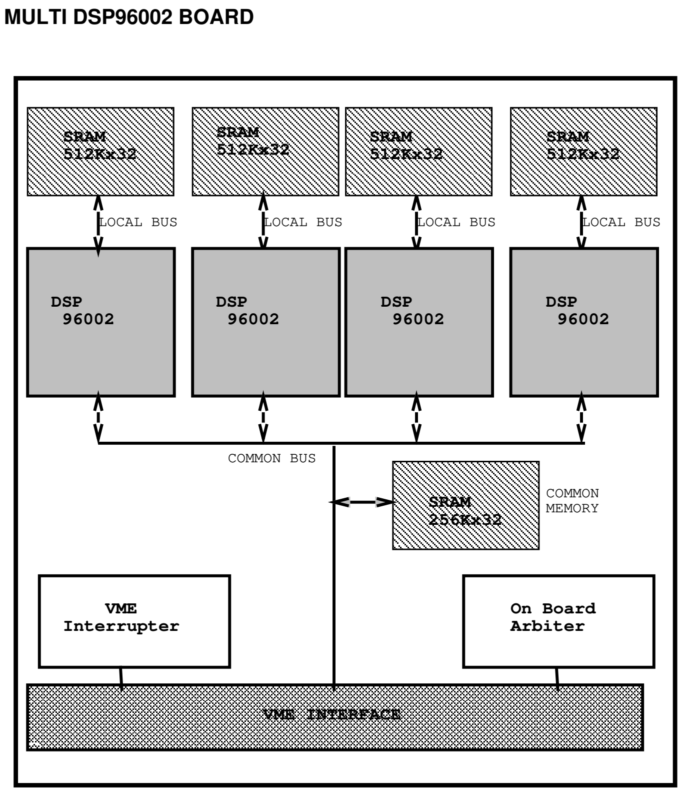


Figure 6: MD96 hardware architecture.

- The disparity map obtained at this stage is noisy and we clean it up using a succession of averaging, dilations and erosions.
- 3D reconstruction is then performed to obtain three-dimensional coordinates.

The algorithm can easily be parallelized by splitting the input images into horizontal bands, therefore the implementation was easy and all the effort was put on the optimization of the data flow. Our current implementation goes as follows.

- The first step consists of loading the input image (coming from the image frame grabber) into the communication memory of the MD96 board. This operation is performed by the host computer.
- Then the host computer starts all the DSP's at the same time. Each DSP runs its own program starting with the rectification. Each DSP has an identification number and thus knows which image area it has to process. The horizontal bands have to overlap since when computing the correlation score at pixel (x, y) with a $(2n+1) \times (2m+1)$ correlation window, the processor needs to access all the pixels with y-coordinates between $(y-m)$ and $(y+m)$ (see figure 7).
- When each processor is done, the Host Computer reads the resulting sub-images of 3D points..

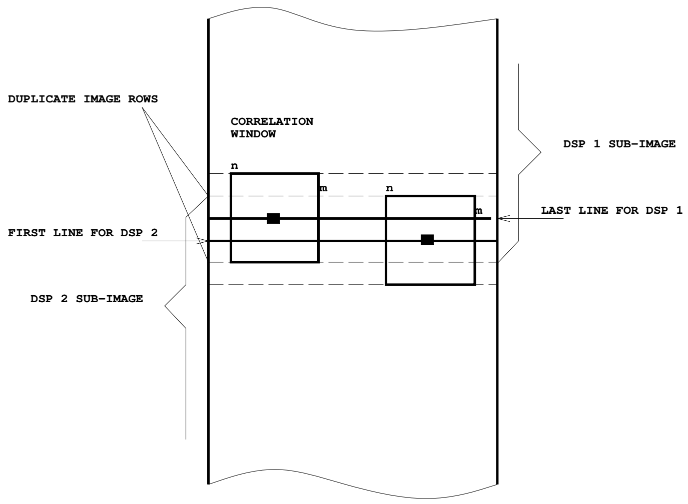


Figure 7: The horizontal image bands which are processed by each DSP have to overlap by half of the vertical size of the correlation window

3.3 Results

We now give some detailed timings of our implementation.

3.3.1 Sub-Process Percentage

The following table shows the average percentage of the total processing time for each of the four parts of the program. The percentage of the correlation is of course the largest and it should be there that some effort in code optimization (for example programming in assembly language) could benefit the most.

Algorithm	**%**
Rectification for 2 Images	8
Correlation	79
Cleaning	6
Reconstruction 3D	7
Total	100

3.3.2 MD96 board versus Sparc-Station

The following table shows the total execution time for the MD96 board and the Sparc SS10 implementations. The image size is 256×256, the disparity range 20 and the correlation window is 7×7.

Algorithm	**MD96 (8 DSP)**	**Sun Sparc SS10**
Computation Time	3.6 sec.	14.0 sec.

3.3.3 Influence of parameters settings

The three graphs of figure 8 show the influence of the image size, the disparity range, and the number of DSP's used on the total processing time.

They include only the execution time of the program and not the time to load the input or to read out the results (IO time). Nevertheless the number of MD96 does not influence the IO time for two reasons. First, our hardware interface allows to load the same data to several MD96 boards without any delay and second, reading the results is, of course, not dependent on the number of DSP's but depends on the image size only.

Thus, increasing the number of DSP's does not increase the IO time. Nonetheless, as the number of DSP's becomes bigger, the image bands become smaller but the number of overlapping lines stays the same as required by the size of the correlation window. Nevertheless only the rectification part is affected and that is only 8 percent of the execution time.

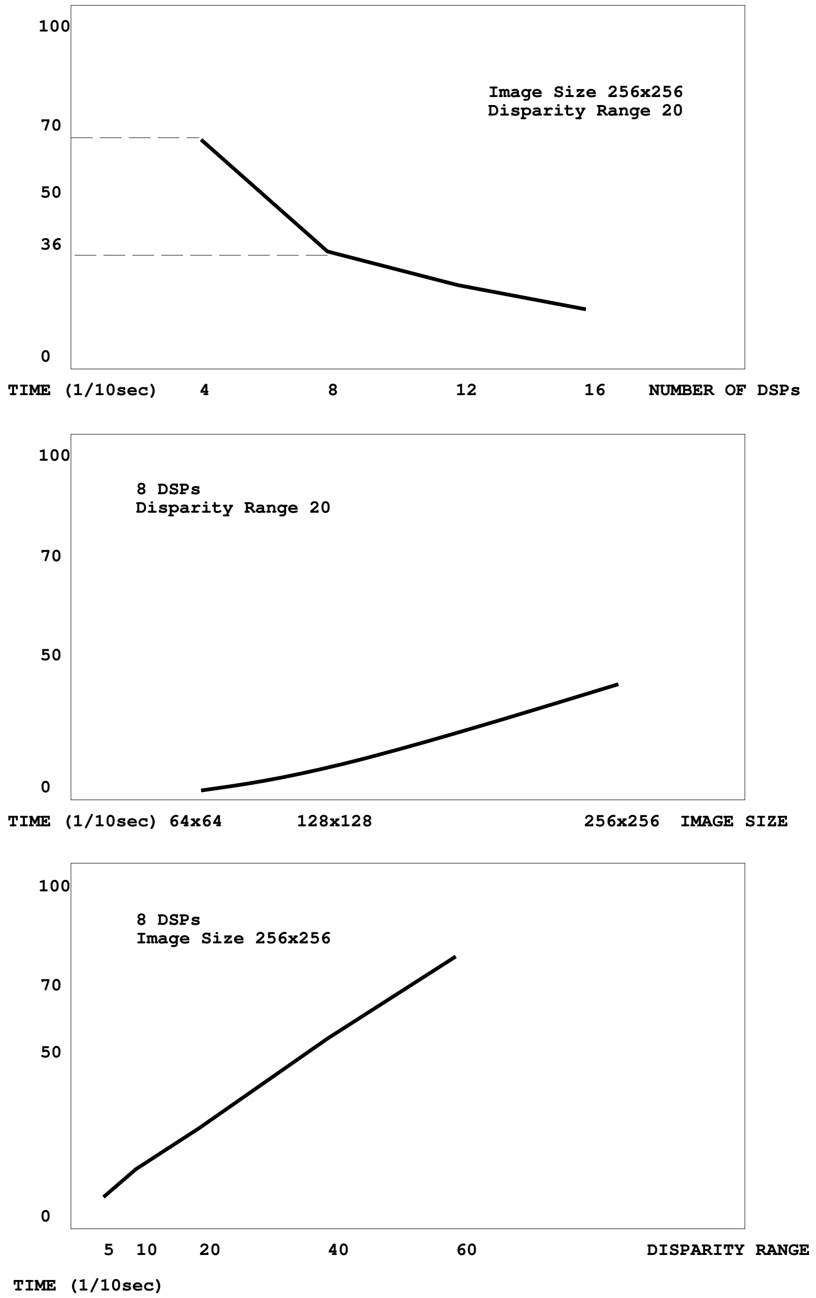


Figure 8: The influence of the number of DSPs, the image size, and the disparity range on the computation time

3.3.4 Some figures about execution time

Table 3.3.4 shows processing times for a variety of DSP configurations, image sizes, and disparity ranges. Since the MD96 board has four DSP's, and the tests were made using the full board configuration, the **NUMBER OF DSP** is a multiple of four.

NUMBER OF DSP's	IMAGE SIZE	DISPARITY RANGE	TIME (1/10 sec)
8	64×64	5	<1
8	64×64	10	1
8	64×64	20	2
8	128×128	5	4
8	128×128	10	5
8	128×128	20	9
8	128×128	40	15
8	256×256	5	16
8	256×256	10	23
8	256×256	20	36
8	256×256	40	63
8	256×256	60	88
4	64×64	5	2
4	64×64	10	2
4	64×64	20	4
4	128×128	5	7
4	128×128	10	11
4	128×128	20	17
4	128×128	40	29
4	256×256	5	31
4	256×256	10	44
4	256×256	20	70
4	256×256	40	120
4	256×256	60	169

3.3.5 Some examples

Two MD96 boards have been integrated in the programming environment of the INRIA group under VxWork and are used routinely to process sequences of images. We show in figure 9 three stereo pairs of a sequence that was acquired outside the laboratory and processed on line, the results of the processing of the previous pair being stored in parallel on disk. These pairs can be fused by cross-eye fusion. The images are 256×256, the disparity range is 24 pixels and the correlation window is 9×9. The generated disparity maps are 192×192 and the three-dimensional reconstructions are shown in figure 10 as cross-eye stereograms. One can clearly see the 3-D structure of the scene.

4 PeRL implementation

4.1 Introduction

In this section, we present an implementation of the computationally most intensive part of the stereo algorithm described in section 2 which is quite different in philosophy from the

Figure 9: Three stereo pairs of an image sequence taken outdoors.

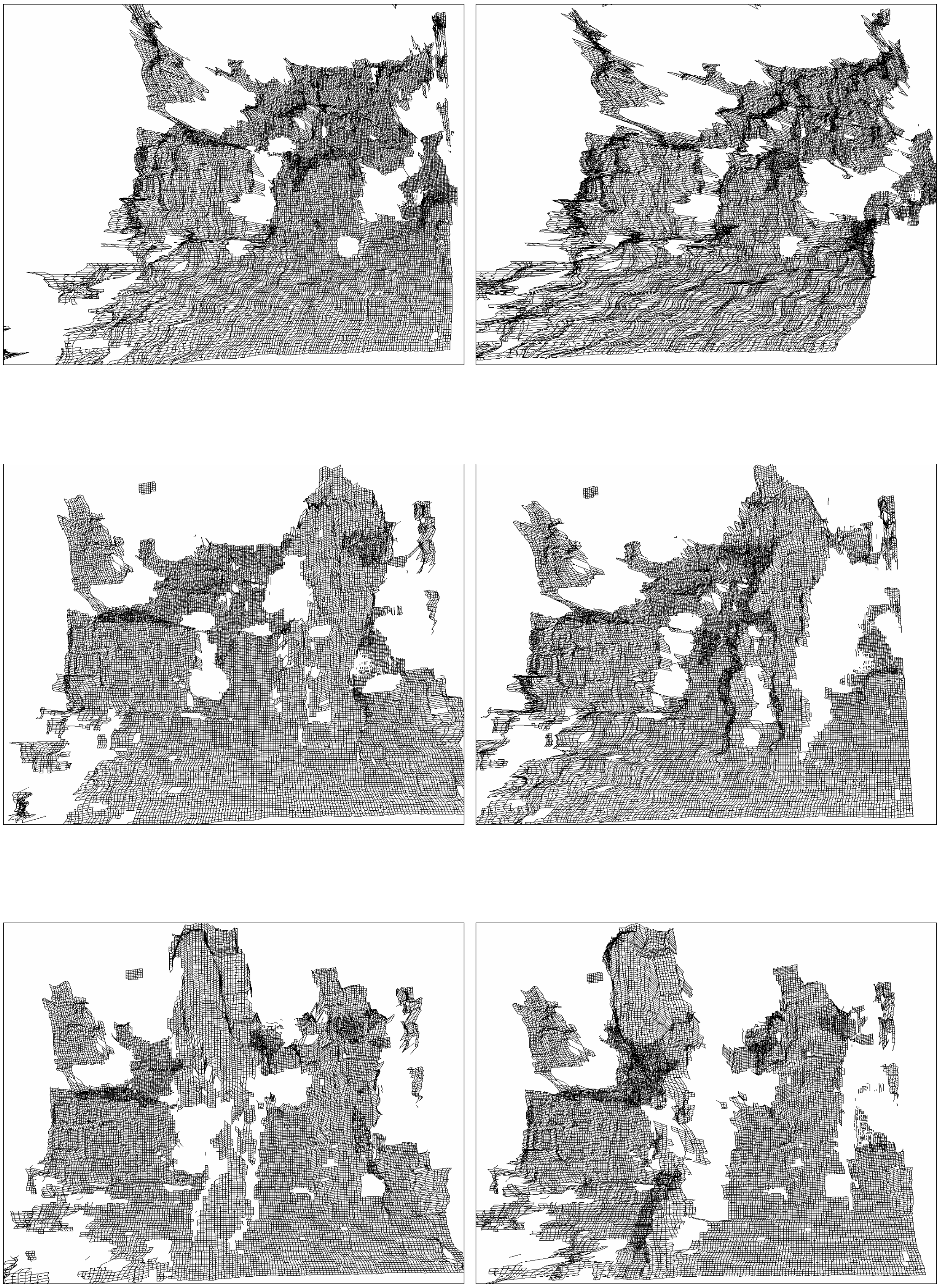


Figure 10: Three cross-eye stereograms of the 3-D reconstructions of the previous stereo pairs obtained by the correlation-based algorithm.

one reported in section 3. The DSP implementation is an attempt to have both high-level programming facility and efficiency, the *Programmable Gate Arrays* (PGAs) we will describe in this section is more efficient at the cost of a somewhat more difficult programming.

This implementation was realized at Ecole des Mines Sophia-Antipolis by Laurent Moll, student at Ecole Polytechnique, under the direction of Gérard Berry.

We will first present briefly the board and then describe the implementation itself; we will eventually give the compared performances of the design.

4.2 The PeRLe-1 board

The board was developped at DEC-PRL by the Circuits group led by Jean Vuillemin. It is part of the *Programmable Active Memories* (PAM) program which started with a functional prototype called PeRLe-0. PeRLe-1 is a far more powerful model, and it is easier to use.

This board is composed of 23 Xilinx *Logic Cell Arrays* (LCAs). These circuits named XC3090 are composed of 16x20 elementary *Configurable Logic Blocks* (CLBs), which contain a fully programmable combinatorial function with four or five boolean input variables and one or two output variables. The CLBs also contain two synchronous registers with clock-enable and asynchronous reset control signals. We can hence adopt the fully synchronous model for the programming of the board.

The center of the board is composed of a 4x4 matrix of LCAs, which are called *matrix LCAs*. Each of the sixteen LCAs is linked to its four-neighbours by 16-bit-wide buses called *direct connects*. The border LCAs can be used to send signals on external connectors (for instance audio ports or TURBOchannel buses). The matrix is typically used for the computation, the control of the design being left to the peripheral LCAs.

Four on-board 256kwords static RAMs with a synchronous interface can be used by the design for storing purposes at a speed exceeding 25MHz. Their data buses are wired to four LCAs called *switches*, which are generally used as intelligent switching matrices between the RAMs and the matrix. The connections are composed of 16-bit-wide buses for each line and column of LCAs in the matrix. Two other LCAs called *controllers* provide the control and the address bus signals. Each of them is in charge of two RAMs. The clear separation between the data and the control comes from the usual uselessness of communication between these two functions: in general, RAMs provide data in a certain order which is independant from the values of these data. Eventually, there is a LCA called *FIFO switch*, which is used as interface with the host.

The general *bandwidth* of the board is 16 bits. It will therefore be difficult to use more than 16-bit data, though not impossible, as we can multiplex the communications through time or even use parts of the switch buses to access more bits. This general architecture is shown in figure 4.2.

The communication with the host uses the DEC 32-bit 25MHz TURBOchannel bus. There are two ways of exchanging data with the host: first, a 24-bit-wide line called *LCBus* on which both the board and the host can write at any time. This solution is used mainly for the control of the design by the host. The other solution, used for the transfer of data, is a buffered waiting queue of the FIFO style (First In First Out). It provides a perfectly

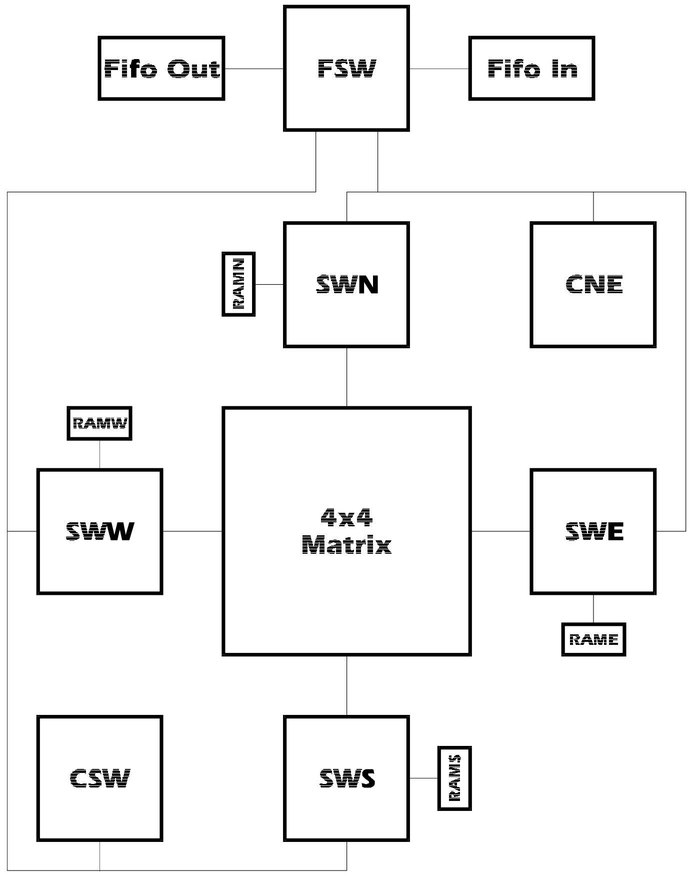


Figure 11: General architecture of the PeRLe1 board

synchronous interface with the host. The data can be as long as 32 bits plus some *tag* bits. For large amounts of datas, one can use a *Direct Memory Access* (DMA) to the host's RAM almost at the speed of the bus (i.e. 100Mbytes/s). The DMA transfer is very well implemented since it can run in background while the machine does something else (including using the bus).

The clock provided by the board is programmable and nonetheless very precise. It can work in continuous or in step-by-step mode. The *auto-stop* mode is a very important though unusual feature: in this mode, the clocks automatically stops whenever the input FIFO is empty or the output FIFO is full. A design is therefore guaranteed to always have data to process, and to be allowed to send a result to the host via the FIFO.

Thanks to the registers, the synchronous interface with the host and the auto-stop mode, the whole board is finally very easy to program even though it is hardware.

4.3 The implementation

We have implemented the second part of the algorithm, i.e. the correlation of the rectified pair of images because, as shown in section 3, it is the most computationally intensive. Indeed, the rectification and reconstruction stages have complexities $NI \times NI$ if NI is the image size, whereas the correlation stage has a complexity proportional to $NI \times NI \times D$ where D is the size of the disparity range (typically an order of magnitude below NI).

4.3.1 The computation

The computation is performed as described in section 2.2.6. In particular, we use the *same* simplifications.

The datapath for the design is the following:

- We traverse I_1 and I_2 along columns x and $x+d$. At the end of each column, increment $x \rightarrow x+1$. After NI such increments, change $d \rightarrow d+1$ and $x \rightarrow 0$.
- We read now the result $Q(x, y, d)$ along row y. We can use the recursion to calculate $N(x, y, d)$. At the end of each line, increment $y \rightarrow y+1$. After NI such increments, change $d \rightarrow d+1$ and $y \rightarrow 0$.
- We compute the denominator in the same way, since the principle of the computation is identical, except that we must use P_2 and not P during the first step.
- We caompute the correlation score by dividing N by $\sqrt{M}$. We then choose the best d.

4.3.2 The hardware implementation

The essential difficulty of implementation comes from the rearrangement of the variables order. We must indeed change the order of the dataflow to use the recursions. There are two cases for the rearrangement, with adapted solutions:

- If, of the three variables x, y and d in some order, we must not change the place of the third variable (the variable which changes the least often), we can use a *double-buffer*: we write in a RAM while we read in a different order in another RAM; when the value of the third variable changes, we permute the roles of the RAMs. As the dataflow is not interrupted, this system is very interesting, because we can use the pipeline capabilities of the hardware, which is a great factor of speed. We just have to fill the first RAM before, and then, the pipeline goes on at full speed.

- If we must change the position of the third variable, we can not use a double-buffer. We are obliged to proceed in two different and sequential steps. In addition, the amount of RAM needed is bigger as we must store all the datas before being able to read them in a different order. In our case, with usual values of NI and D ($NI = 256, D = 32$), we need $NI^2 \times D$ =2Mwords, which is too much for a single RAM on the board. We must hence use the host's memory (which we access via DMA to keep good performances), but we nevertheless have to put a double-buffer after the DMA, because the access to the memory is done by at least 64-word blocks.

As there is an unavoidable split in the dataflow, we can use two different designs on the board, switching from the first to the second after the completion of the DMA transfer to the host's RAM.

We can now propose a hardware datapath:

- RAM West sends the necessary I_1s and I_2s to the matrix LCAs.

- A multiplier calculates P.

- The datas pass then through a delayed accumulator, implementing the recursion formula $Q(x, y+1, d) = Q(x, y, d) + P(x, y+2m+1, d) - P(x, y, d)$.

- RAM North and East implement the double-buffer, changing the variables order from (y, x, d) to (x, y, d).

- The datas pass through another delayed accumulator which implement the second recursion $N(x, y, d) = N(x, y, d) + Q(x+2m+1, y, d) - Q(x, y, d)$.

- The results are then sent to the FIFO to be stored by the host.

- Note that the denominator can be computed by the same circuit, so we use the first $NI \times NI$ clock ticks for it. We store it in RAM South.

We change the design and after that:

- We read the memory in the proper order (x, d, y).

- A double-buffer finishes the rearrangement of the variables: (d, x, y).

- The numerator comming from the double-buffer is divided by the square root of the denominator, which has been stored in RAM South.

- A *max-unit* finds the best d (the d for which the calculated value is maximum) and sends it to the host for further processing.

We must now deal with the width of the datapath: the former implementations were using standard 32b floating point variables. With PeRLe-1, we must unfortunately try to reduce the sizes of the variables to 16b. As the images use 8b pixels, the result of the multiplication can use a full precision (16b). The accumulator, then, must not accumulate errors; it is hence 24b wide, but its output uses only 16b, losing only 1b of relative precision on each of the K numbers added. The datapath is still 16b wide until the last calculus: the square root and the division. As a division is not very simple to do, we took the decision to use the logs of the numerator and the denominator, which simplifies a lot the calculus of $\frac{N}{\sqrt{M}}$ because:

$$\log \frac{N}{\sqrt{M}} = \log N - \frac{1}{2} \log M$$

We compute the log by a simple *table-lookup* using the free RAM, which is the quickest and the simplest way to do it. A problem arises because of the log function: the influence of the least significant bit out of 16 has an influence on the result not before the 20th. We are then obliged to keep at least 20 bits until the end.

4.3.3 The operators

We use some original hardware operators in these two designs that should be describe at least briefly:

- The delayed accumulator: it is made of three parts; the first one is a long delay, composed of K registers, which provides the old value necessary to the recursion; the second one is a substracter the operands of which are the entry and the delayed entry; the third part is a standard accumulator, which is an adder using its own output as an input for the latter cycle. The adder (and the substracter, which is strictly equivalent) is a standard parrallel adder made of 1b full adders, and retimed every second bit, least significant bits first, in order to cope with the maximum speed (i.e. the RAMs speed).

- The multiplier is a standard combinational one, which uses a carry-save format to do the internal partial sums. It needs back-end adders to get a regular binary form as output. It is also retimed every second bit.

- The counters are those developed by Jean Vuillemin. They are arbitrary-length constant-time counters. They use a standard few bit counter and an anticipated $+1$ accumulator. The result of the accumulater is enabled on the output only every terminal count of the first counter. The accumulator has hence much time to do the $+1$ operation.

- The max-unit uses a pretty smart system: it compares two numbers begining with the most significant bits. We must still retime every second bit, so the comparison can not be made in a single cycle. The advantage of the most significant bits ahead is that: if the choice can be made, the replacement can occur, if it can not, the most significant bits are equal, so whatever the final result of the comparison, they are already correct.

4.4 Performances

We can now calculate the global time of excution, for the standard values of $NI = 256$ and $D = 32$ (the speed being independent of K because of the recursion), and for a double correlation (image1 $\rightarrow$ image2 and image2 $\rightarrow$ image1):

- It is the first time the board is used with real-time reprogrammation, but the performances were already optimized: downloading the design and reseting the board takes 22ms, so 44ms here for two designs.
- The first design runs for 2 denominators and D numerators for $NI \times NI$ cycles each plus the initial filling time for the pipeline: $256 \times 256 \times 35$ cycles. The first design can run as fast as 35ns, so the total time is 0.080s.
- The second design must run twice because of the two correlations but it must not be reloaded. Its total running time is: $256 \times (32 + 256 \times 64)$ cycles of 37ns, so 0.216s.

We finally obtain a total time of 0.28s for two correlations, so 0.14 by correlation. We can now compare with the former implementations:

Sparc 2	DSP(x4)	PeRLe-1
59 s	9.6 s	0.28 s

We obtain a factor of 34 between the PeRLe-1 board and one MD96 board, i.e. four DSPs. Note that even though all the computational power of the board has not been used, the implementation is still far quicker than any software implementation.

Thanks to the relative easiness of development, we can be sure that PAMs will be a very good help for intensive and real-time needs in the future.

5 Robot navigation application

5.1 Description

As an illustration of the use of the DSP implementation of the stereo algorithm, we are now going to describe the development and implementation of a reactive visual module which is being used on the INRIA cart to automatically correct for deviations of the predefined trajectory. Note that this module uses both inertial and visual cues as well as odometry. We only describe the implementation and the experimentation of this module, the algorithm having been already discussed in section 2.

Position of the problem The INRIA cart has to follow a predefined trajectory, but due to errors and perturbations, it deviates from this trajectory and we must compensate for the errors.

Considering a planar motion in a frame of reference attached to the robot, we can decompose the navigation errors into three components: a lateral error along the Y-axis, a longitudinal error along the X-axis and an angular error related to the absolute orientation angle of the robot a.

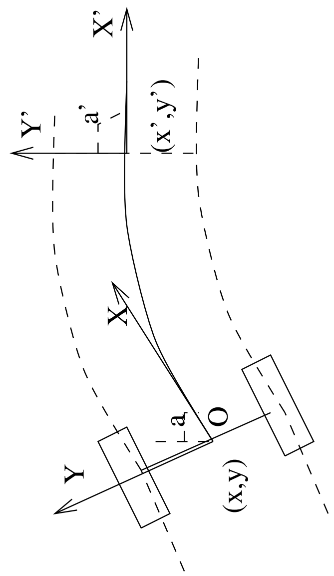


Figure 12: Kinematic of the mobile robot, with differential wheels.

The crucial parameter to correct is the angular position of the robot because its lateral displacement is subject to a cumulative error whose amplitude is a linear function of time and of the angular position error. As a consequence, we must estimate the angular displacement of the robot by combining visual, inertial and odometric cues in order to correct the main errors which occur while following the predefined trajectory.

The residual lateral correction must also be effective, but is usually difficult to obtain from odometric or inertial cues. The integration of a linear position from inertial cues is known to be subject to errors which vary quadratically with time, and in our configuration, the odometric cues are also subject to unobservable perturbations (uncertainty in calibration, sliding, etc...). Therefore, vision is required to correct this source of errors.

Finally, it is not essential to correct a small longitudinal error for road following for two reasons : (1) a small error in the direction of the robot heading is not going to put the robot off the road as a lateral error would, especially for rectilinear parts of the road, and (2) for curved parts of the road (considering a 90 deg turn for instance) this error is in fact reported as a lateral error, and thus will be corrected by the system. In practice, this error is quite small is any case.

Using inertial cues for ego-motion estimate The use of inertial measurements on a robot has been reported in previous papers such as [32, 31]. In particular, we have demonstrated the possibility to implement and calibrate a low cost inertial system on a robot, in order to obtain information about the robot ego-motion [33]. Inertial measurements yield the same kind of information as that provided by passive navigation algorithms using artificial vision, but with a different dynamic range and precision. Thus, cooperation between these two sensory modalities is a priori useful.

As of today, inertial sensors such as linear accelerometers and either angular rate sensors or gyroscope are available at costs which are similar to those of a camera.

In our application, the inertial system computes the horizontal orientation (heading) with high accuracy for short periods of time. It returns this information as a regularly updated value, but without any synchronization with respect to the odometric measurements.

The error is calculated as the difference between the expected and the measured headings. In order to take into account the fact that we do not know the exact time when the measure was made (sometime between the present and last sampling times), this angular value is estimated as the mean between the two intermediate values. The heading is corrected for the next displacement, and the heading error is calculated when a new command is sent to the mobile robot. The resulting equations of heading correction are:

$$\begin{aligned} HeadingError &= \tfrac{LastInertialHeading+PresentInertialHeading}{2} \\ &- ExpectedHeading \\ NextHeading &= ExpectedHeading - \kappa ErrorHeading \end{aligned} \tag{1}$$

The adaptive parameter κ has been adjusted to obtain an optimal correction, as detailled in [34]. It is easy to verify that the gain of the correction must be $1/3$ in order for the correction to be asymptotically stable. As discussed in [34] this simple correction seems to be a nice compromise for the control of a mobile robot.

Computing the angular and lateral errors using stereo Now, having a 3D map of the scene it is very easy to detect the road/corridor since it corresponds to a flat region of the scene "at height 0", in front of the robot. We have detected the edges of the road/corridor by thresholding the height of the 3D scene and have extracted the left and right limits of the road/corridor (which have been modeled as locally rectilinear edges).

Considering these two parallel lines we can first compute their median line, the axis of the road/corridor. Second we know from calibration the axis of the robot. The lateral error, the distance between the road axis and the robot axis at the robot location, and the angular error, the angle between those two lines, can thus be computed.

Detection of outliers measures of lateral errors is easily implementable: we reject values which are too high and only accept the value as correct if, assuming approximate values of the robot and road widths, the corresponding correction cannot put the robot off the road.

This is shown in figure 13. The angular error is used to reset inertial measurements.

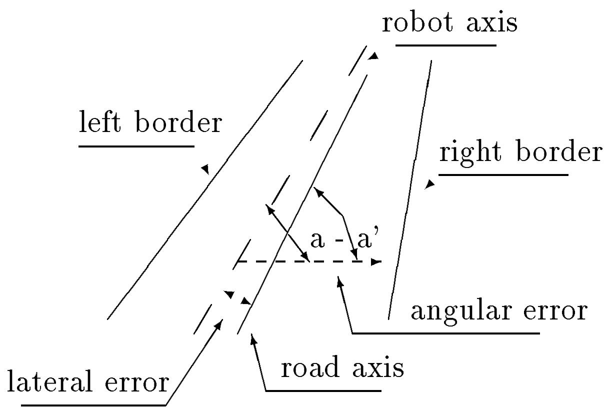


Figure 13: The definition of the lateral and angular error of the robot on the road/corridor

The general algorithm has been implemented as follows :

- Split the depth map into two parts corresponding to the left and right parts of the road, using the initial estimate of the road axis obtained from the positioning of the robot.
- For each part :
  - Define a few search areas for the road side which are located at different distances from the robot :
  - For each area :
    * Compute the histogram of the height of the road.
    * Threshold and filter the histogram.
    * Search for the first points at a non-zero height, they yield the location of the side of the road.
  - Compute the mean and variance of the edge location for the set of measures obtained in each search area.

- Compute the angular and lateral errors of the mobile robot (mean and covariance matrix).
- Fuse the information with the inertial and odometric cues (Kalman filtering).

Note that we have exploited the simplicity of the scenes to avoid the use for heavy computations. When defining the trajectory in terms of a list of absolute positions/orientations to attain, we have also recorded the places where visual information can be reliably computed and the areas where it cannot (because we expect to see nothing for instance). Moreover, we have used a very simple thresholding mechanism to reject unreliable estimate of the road edges (for instance the road width must be in a certain interval, the edges relative orientation must not differ too much, etc...). We have observed that these heuristics allow to easily reject erroneous estimates, since either there is something to see, and the estimation is reliable, or there is nothing to see and it is not. As a consequence, the adjustment of these parameters is not very crucial, and the final behavior is not very sensitive to their specific choice.

5.2 Results

We have been able to navigate robustly for tens of minutes with the INRIA cart running at 1m/sec indoors (the corridor width was 1.6 m while the robot width was 0.8 m) and 1.5m/sec outdoors (the road width was 5 m while the robot width was (still !) 0.8 m) the limit being the motor torque, not the algorithm.

Analysis of the visual process For indoors scenes, because of lack of texture, the stereo algorithm of section 2 provides relevant values mainly at edges and is thus not much more efficient than edge-based stereo algorithms (in the sense that it does not produce denser range maps); useful parts of the scene are : doors/windows, limits of dividing walls, posters or painted lines on the floor or on the walls. But for outdoors scenes, the stereo algorithm produces much denser maps, since useful parts of the scene are : shadows or textures on the road, vegetation, parked cars, sidewalks; in that case, this dense stereo method is preferable.

The computation time is less than one second for the computation of the 3-D map (128 $\times$ 128) which is performed on the DSP board MD96 described in section 3 and 0.35 sec for the computation of the road edges which is performed on a Sun4-II Sparc station.

In principle, the technique for the road extraction is sensitive to the number and quality of the points detected as belonging to the road edge. We thus have measured the stability and the repeatability of the method for several set of measures. As in the algorithm, the estimation takes into account the variance of each estimate and thus uses an optimal estimation in the least-squares sense (maximum likehood, which reduces in that case to the minimization of a criterion weighted by the variances). As an example of the kinds of results we have been able to obtain, we show three estimates which have been obtained in three different experiments. The results are displayed in table 5.2. The first row shows the real value, the second the number of independent measurements which have been taken into account to produce the optimal mean value showed in the third row and the optimal standard deviation showed in the fourth row. To show the importance of taking into account the variance of each estimate, we found a mean of 231 mm and a standard deviation of 215 mm in the first experiment by doing simple averaging. It demonstrates that the estimated variances are meaningful and must be taken into account.

	Exp. 1	Exp. 2	Exp. 3
Real value (mm)	300	-300	200
Number of measurements	13	16	12
Optimal mean value (mm)	301	-303	212
Optimal standard deviation (mm)	83	68	71

Analysis of the visual compensation In order to test our system we had to artificially increase the navigation errors because the system was too stable to observe any relevant visual correction.

We have thus introduced a drift in the inertial system of 50 to 200 deg/hour. We also have voluntarily introduced some errors in the trajectory : (1) an initial lateral and longitudinal error varying from 0 to 20 cm and up to 15 deg in orientation, (2) a continuous error (lateral translation) along the trajectory up to 10 cm for each sampling period, (3) some step-like errors up to 50 cm, lateral and longitudinal. For the previous quantities the mobile cart was always capable to recover, except in cases of a transmission problem. For larger errors, we may not be able to recover from vision and this gives an idea of the limits of our implementation at the present stage.

In order to insure the stability of the mobile robot guidance, we have implemented a compensation mechanism which can take into account inertial and visual cues to compensate both angular and lateral errors, but with a weight in favor of the inertial cue for the angular error, and in favor of the visual cue for the lateral error. With this simple strategy (obtained by a correct balance of the variances in a standard Kalman filter) we have obtained a stable control, we have observed the capability of the inertial system to act as a short-term correction, especially when the vision was not usable, and have verified that the vision can perform a long-term correction and thus cancel the inertial system drift.

At the beginning of a trajectory (bootstrapping phase), the system needs 1 to 3 sampling periods to obtain a good visual correction, and thus the initial velocity must be rather small (10 cm/sec indoors and 30 cm/sec outdoors). In the steady-state phase, errors to be corrected are less than 1deg and 10cm indoors and less than 1 degree (2 degrees during the first minute) and 50cm outdoors. In agreement with these results the uncertainty (standard deviation) of the measures initialized at 5 degrees and 50cm decreases quickly to 0.5 degree and 10cm indoors and 0.5 deg (2 degrees during the first minute) and 30cm outdoors.

A few measures are much less precise (1.5m of standard deviation) and thus not taken into account. In detail, outdoors, more than 80% of the measures have less than 30cm of uncertainty, whereas only 5% have more than 50cm of uncertainty.

Some recorded data We show some results indoors in figures 14 and 15.

Figure 14: A session of autonomous navigation, indoors, see text.

With more details, because this is a more difficult task, considering the nature of the scene, we have shown two examples of corrections in figures 16 and 18. The positioning of the robot after its correction obtained from the measurements shown in figure 16 is shown in figure 17.

Lines have been drawn in the pictures to allow the analysis of the behavior of the visual compensation :

- A white line corresponds to the axis of the road (hallway) at a distance of 10m, in the estimated position before the visual compensation. It is drawn as if on the floor.

Figure 15: Another session of autonomous navigation, indoors, see text.

- Three black lines corresponds to the left/right edges and the axis of the road (hallway), as found in the 3D reconstruction; they intersect.
- Two light polylines join the points which have been detected as road edges, for the left and right edges.
- The white line and the three dark lines are also drawn in a top view in the upper part of the image, without any relation with the image itself.

With this representation the reader can easily observe the good results of the algorithm. The image sequences contain not only the “best” results but also some estimations which have been rejected by the algorithm as described previously. This case is easy to detect on the picture considering the road estimations reported in the images.

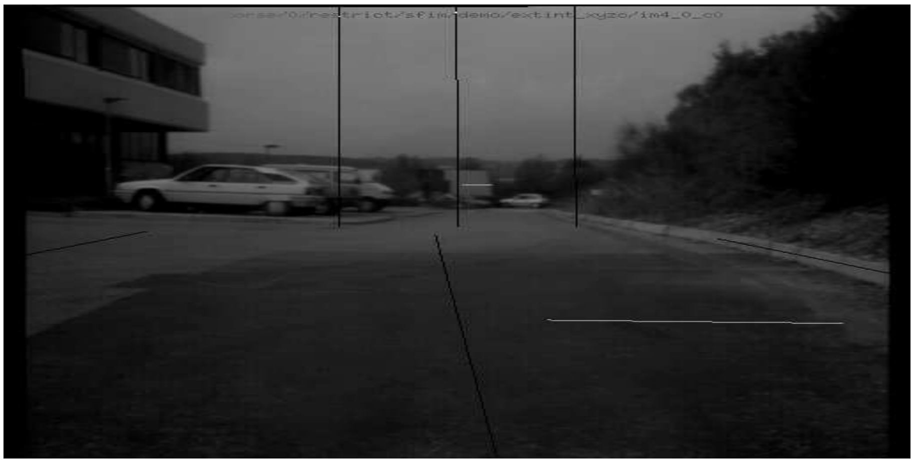


Figure 16: One example of lateral correction, outdoors, see text.

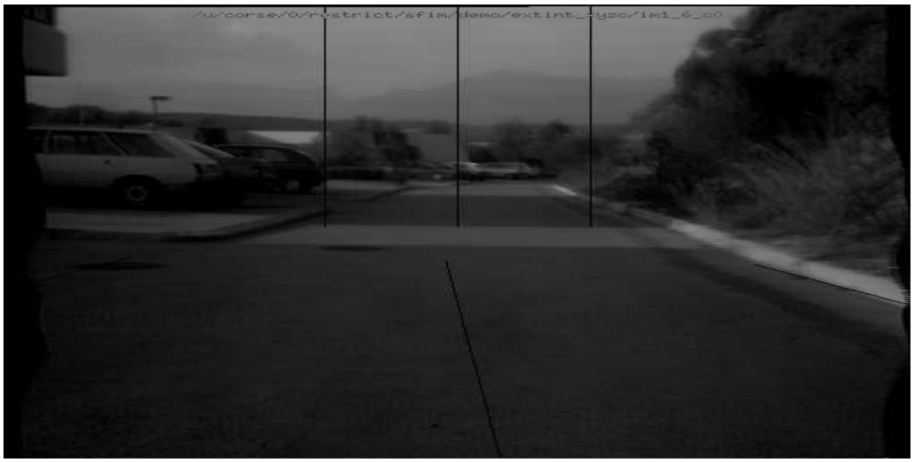


Figure 17: The positioning of the robot after the previous correction, see text.

Conclusion This mechanism is an example of an operational and efficient use of reactive vision, with multi-sensor fusion of inertial and visual cues. Using this mechanism, we can

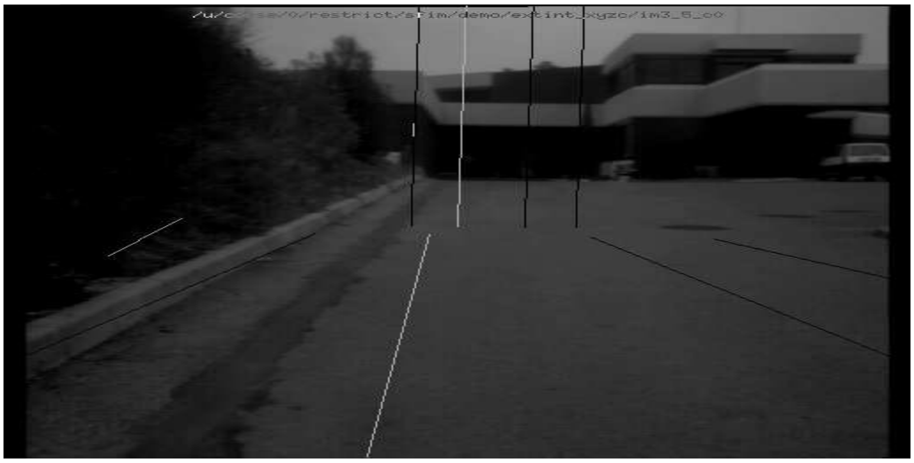


Figure 18: Another example of lateral correction, outdoors, see text.

correct for positioning errors and recenter the mobile robot on its trajectory. It is quite sophisticated at the implementation level, but its architecture is a simple instanciation of the following rule for the multi-sensor cooperation :

- Inertial cues correct the robot trajectory for short term errors, and mainly its orientation.
- Visual cues, in this case, correct long term errors and mainly the lateral bias of the robot on its trajectory.

A 3D model of the visual surroundings is generated by the system and can be used for higher level mechanisms of perception.

The system is "cheap" in the following sense : we need only two cameras, an image acquisition module, a one axis low-cost gyroscope and a 4-DSP high-speed computer board[22] for a rate of correction of 1.5 sec (computation time).

6 DEM Computation

6.1 Description

The correlation-based stereo algorithm described in section 2 has been integrated into the French autonomous planetary rover program (VAP). The goal of this program is to build an intelligent rover for planetary exploration. Long term plans aim for Mars surface exploration. The rover should be capable of navigating through rough terrains. It must consequently feature a powerful vision system to allow it to perceive and model its local environment.

Such a model will then be used in path planning and subsequent obstacle avoidance. To this end, we need to solve the two following problems: Firstly, register successive 3-D visual maps reconstructed by the correlation-based stereo in order to better localize the rover itself; Secondly, fuse all data acquired during the rover motion in order to build a Digital Elevation Model (DEM) of the environment, directly usable by a path planning module.

6.2 Registration

The objective of this step is to compute precisely the displacement of the vehicle between successive views in order to register the 3-D visual maps reconstructed by the stereovision system. This step is indispensable for the following reasons:

- better localize the mobile vehicle,
- eliminate errors introduced in stereo matching and reconstruction,
- build a more global DEM of the environment.

However, geometric matching in general is a difficult unsolved problem. We have studied the state of the art, and there does not yet exist a robust algorithm. Fortunately, the problem can be simplified in our application. Indeed, our Mars rover is equipped with several sensors such as an odometer and several inertial measurement systems which can provide an estimate of the displacement with a reasonable accuracy. We can first apply the rough estimate of the displacement to the first frame to produce an intermediate frame; then the motion between the intermediate frame and the second frame can be considered to be small. The method to be described in the sequel then refines this initial estimate in order to register the two frames with a very good precision.

The key idea underlying our approach is the following. Given that the motion between two successive frames is small, a point in the first frame is close to the corresponding point in the second frame. By matching points in the first frame to their closest points in the second, we can find a motion that brings the two sets of points closer. Iteratively applying this procedure, the algorithm yields a better and better motion estimate.

6.2.1 Algorithm

The algorithm is an iterative process, each iteration consisting of the following steps.

1. Find closest points. For each point in the first frame (after applying the previously estimated motion), find its closest point in the second frame. k-D trees (abbreviation for *k-dimensional binary search tree*) have been implemented to speed up the search process.

2. Match points. We only retain as potential matches the pairings of points between the two frames whose distances are less than some maximum tolerable distance D_{max}. This constraint is justified since we know that the motion between the two frames is small and hence the distance between two corresponding points cannot be very big.

3. Update the matching. Instead of using all matches recovered so far, we exploit a robust technique to discard several of them by analyzing the statistics of the distances. The basic idea is that the distances between corresponding points should not be very different from each other. We first compute the mean μ and the sample deviation σ of the distances. Depending on the values of μ and σ, we adaptively set the maximum tolerable distance D_{max}. Eventually we use the newly set D_{max} to update the matching previously recovered: a pairing is removed if the distance is bigger than D_{max}.

4. Compute the motion. A least-squares technique can be applied to the remaining point pairings to compute a better estimate of the motion between the two frames. To this end, we have implemented two efficient algorithms, namely the quaternion method and the dual number quaternion method.

5. Apply the motion. Apply the motion estimated to the first frame, and go to the next iteration until convergence is reached.

For more details of this method, the interested reader is referred to [35].

6.3 Fusion

A depth map delivered by the correlation-based stereo algorithm cannot be directly used for path planning in a rough terrain. Indeed, path planning algorithms need a facet representation of the scene whereas the stereo algorithm only provides a more or less dense and noisy set of points in space which must be structured into a surface. This is a difficult problem because the 3-D points can sample space in a very irregular fashion. Moreover, the stereo algorithm sometimes makes errors which must be removed using robust techniques.

On the other hand, a depth map obtained from one position is very local, and thus not very useful for navigation and environment understanding. We would like to fuse the sequence of data acquired during the exploration in order to build a global map of the surrounding environment.

6.3.1 Algorithm

We call digital elevation models (DEM) a regular grid in the xy horizontal plane. At each node of this grid we compute a height z from the local cloud of neighboring 3D points measured by various sensors. Since we do not intend to model precisely vertical or overhanging areas, we can reasonably assume that there is only one height z for a given (x, y) pair in the horizontal plane. We can therefore model the ground and possible rocks as a surface $z = f(x, y)$. Height z for a certain node xy is obtained by fitting a surface to the local cloud of points using a standard least-squares technique. Intersection of the surface with the vertical straight line going through the node is then computed. Our algorithm can be broken into five steps:

1. Bucket grid point sorting. An initial sorting of 3D points eases addressing for subsequent local surface computations.

2. Least-squares local surface adjustment. Planar and quadric surface models are used. The latter are better suited to rock surfaces but are computationally intensive. Planar models have proved to be fast and suitable if one chooses a sufficiently fine DEM resolution and reduced local clouds. All 3D data required for computation of plane coefficients and corresponding covariances can be condensed into a small number of moments (9 exactly). Once stored, such moments enable us to easily merge new 3D points.

3. Removal of erroneous points and surface correction. This stage is required when sensory 3-D data include large errors to which least-squares are very sensitive. We hypothesize that erroneous points are a minority among local clouds and are hence distant to surfaces adjusted at the previous stage. A first method adjusts surfaces by recursive iteration whilst assigning a coefficient to each point until convergence. Such a coefficient is chosen inversely proportional to the distance from the surface. A second faster method consists in directly removing all dubious points by examining point to surface discrepancies and then readjust local surface coefficients.

4. Construction of a graph of neighboring local surfaces compatibility. The aim of this fourth step is to cancel most of the remaining global errors. We define a criterion indicating whether two local surfaces belong to a common global surface. We keep only major connected parts of this graph.

5. Computation of altitudes. This is done by intersecting the local surfaces with vertical lines going through the nodes of the xy-plane. Small gaps in the terrain model are filled-in by closest neighbor interpolation.

During real operations, sampled 3D points will not all be available simultaneously but will instead complement previous samples. The elevation model will be incrementally updated. Hence, we must plan for an incremental adjustment of local surfaces. This is done with least-squares whilst maintaining prior 3D data. Another method would be to update surface coefficients after subjecting them to a Kalman filter. The latter method has the advantage of taking 3D point uncertainty into account. Hence, it will be used if 3D points covariances are available.

6.4 Results

We now show an experimental example with the same scene that was used to demonstrate the stereo algorithm in section 2. Images have been taken from four positions around the rocks, those taken by the first camera being shown in figures 19. The stereo rig is at about 6 meters from the scene.

First position

Second position

Third position

Fourth position

Figure 19: Images taken by the first camera

We then run the correlation-based stereo algorithm to obtain a set of 3D points for each position. The cloud of 3-D points corresponding to the second position is shown in figure 20.

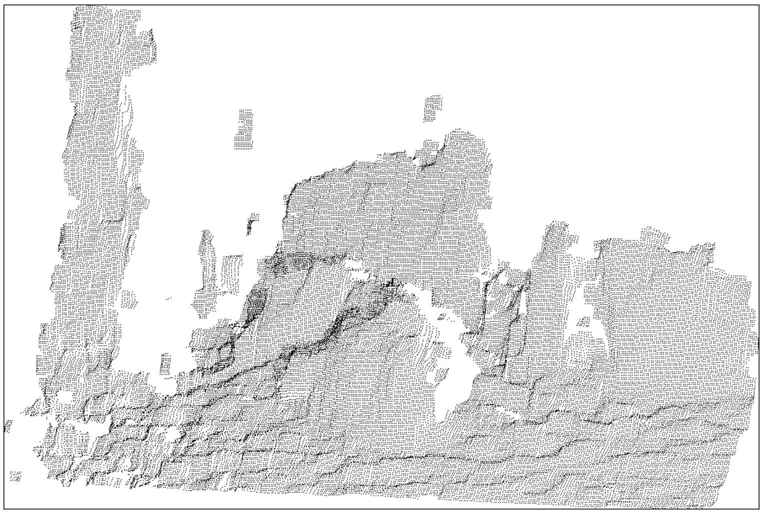


Figure 20: Cloud of the 3D points reconstructed from the second position

As an example, we show in figures 21 and 22 the result of registration between the second and third positions. The correlation-based stereo system reconstructs 71505 points for the second position and 51503 points for the third position. Figure 21 shows the superposition of two sets of 3-D points before registration, where the 3-D points corresponding to the second position are drawn as quadrangles, while those corresponding to the third position are displayed as a shade surface. Between the two positions, there is a rotation of 20 degrees around an axis in the ground plane, and a rotation of 10 degrees around an axis perpendicular to the ground plane. The translation is 2.56 meters. The registration result is quite good, as shown in figure 22. We can clearly observe that the two surfaces overlap nicely but cross each other in a somewhat random fashion because of the random noise inherited from the stereo matching and 3D reconstruction.

Figure 23 represents the DEMs obtained incrementally. The coloration of the facets are in function of their elevations. The fusion procedure is carried out as follows. First, the elevation of each node in the DEM is computed by fitting a plane to the points located in its neighborhood, which requires to compute 9 moments. Besides the elevation, the values of these 9 moments are also saved in the DEM. When a new cloud of points is available, the registration procedure computes its displacement with respect to the previous DEM. The displacement estimate is then applied to all points in the new cloud. Finally, we update the moments for each node of the DEM if there are new points found in its neighborhood, and the elevation is eventually computed. Figure 23 shows clearly the successive filling of the

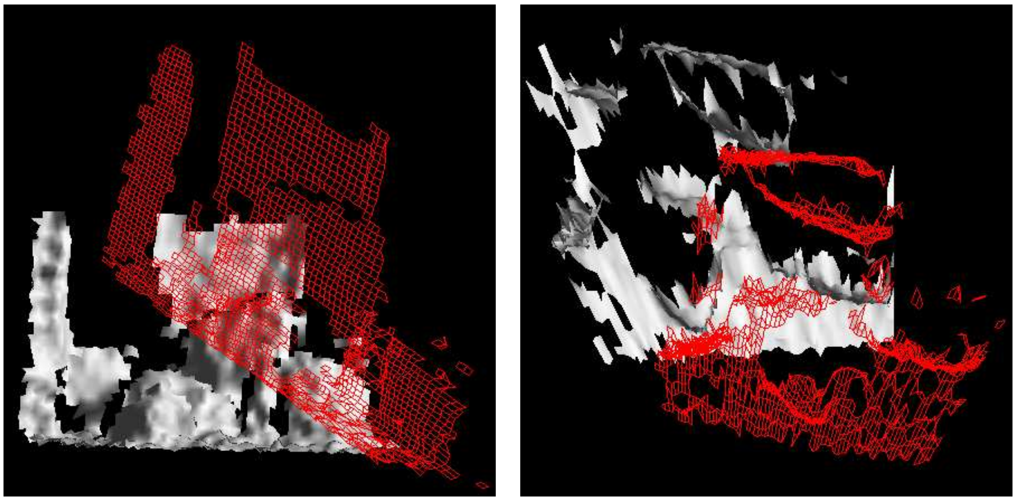


Figure 21: Superposition of two sets of 3D points before registration: Front view and top view

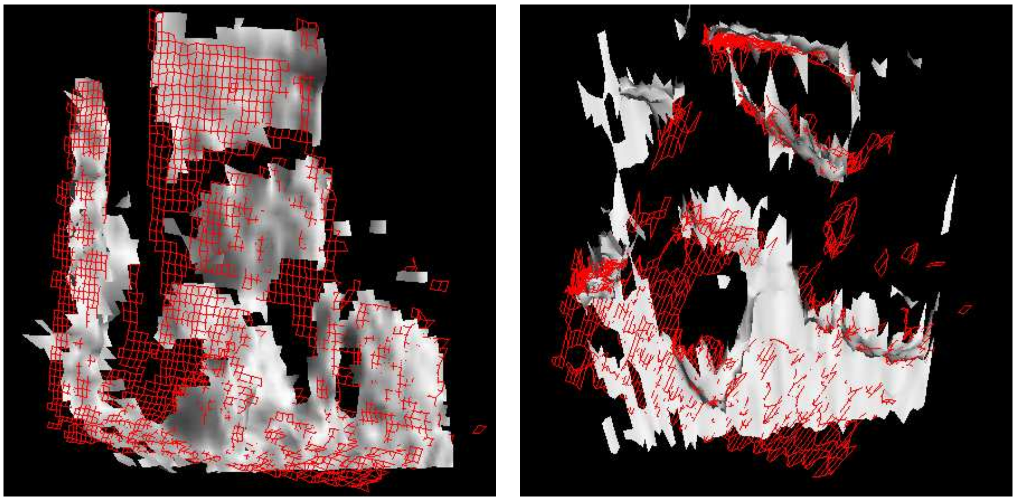
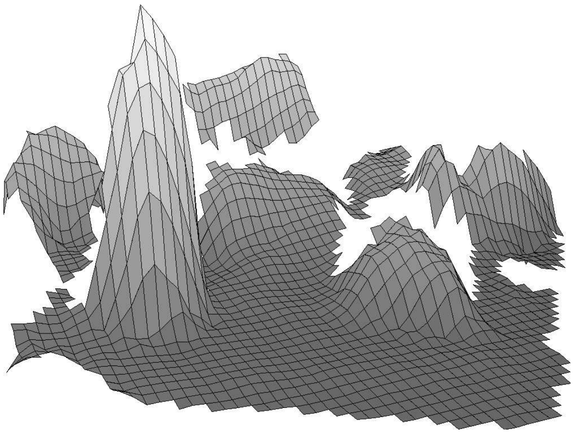
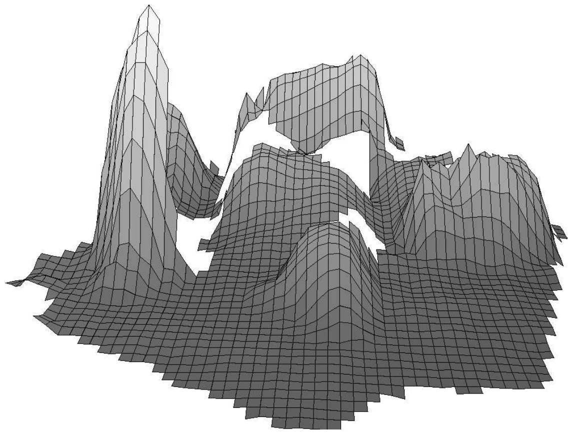


Figure 22: Superposition of two sets of 3D points after registration: Front view and top view

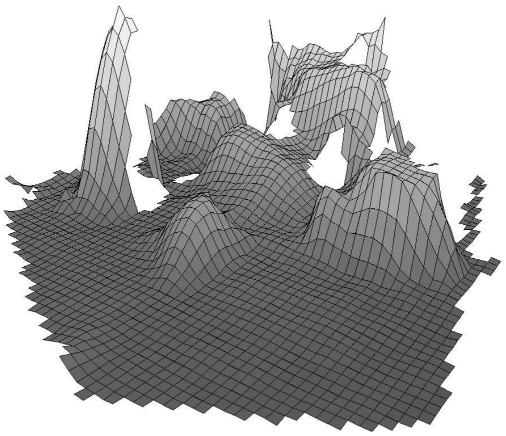
occluded parts. The DEM obtained in the last position is almost complete and almost all obstacles are completely modeled. The DEM is now suitable for the path planning of the rover.



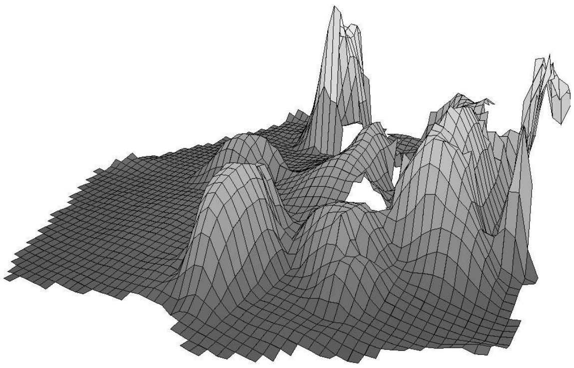
DEM generated in the first position



DEM generated in the second position



DEM generated in the third position



DEM generated in the fourth position

Figure 23: Incremental building of the digital elevation model of the environment

The reader is referred to [16] for more details.

7 Conclusion

In this article we have shown that real time stereo is possible today at low-cost and can be applied in real applications. The algorithm that has been described is not the most sophisticated available but we have made it robust and reliable thanks to a number of improvements. Even though each of these improvements is not earth shattering from the pure research point of view, altogether they have allowed us to go beyond a very important threshold. This threshold measures the difference between a program that runs in the laboratory on a few images and one that works continuously for hours on a sequence of stereo pairs and produces results at such rates and of such quality that they can be used to guide a real vehicle or to produce Discrete Elevation Maps. We believe that this threshold has only been reached in a very small number of cases.

The ingredients for this success have been the choice of a class of stereo algorithms, the method for code development, the implementation on special purpose hardware, and the integration of the stereo module in applications which require reliable results over long periods of time at fairly high rates. As mentioned previously, correlation-based stereo algorithms are perhaps not the most sophisticated algorithms today: They make some simplifying assumptions about the environment but they produce fairly dense results whose validity and accuracy can be quantitatively evaluated. The quality of these results is sufficient for many applications. Moreover, because of the relatively simple structure of the algorithms in this class, they can be easily implemented on simple and cheap parallel architectures. The code we have developed has gone through several iterations, is well documented, and has been tested by quite a few end-users who have suggested improvements and modifications motivated by their specific application. It has been ported on two very different parallel architectures with little effort. In both cases the people involved were not computer vision experts but they did not find it difficult to map the original C-code to their specific architecture. The times involved were of the order of a couple of months. In both cases we have obtained a significant improvement in the throughput with no reduction in quality. Finally, the work of integration of the stereo module in two applications, the navigation of a mobile vehicle, and the construction of composite elevation maps has forced us to take an approach which is more system oriented than pure computer vision. This is more apparent in the navigation application than in the elevation maps application but in the two cases, the consideration of the task to be achieved as the main goal rather than, for example, to produce close to perfect range maps, has changed our way of thinking about the vision task in a way that has allowed us to go beyond the aforementioned threshold.

References

[1] P. Anandan. A Computational Framework and an Algorithm for the Measurement of Visual Motion. *The International Journal of Computer Vision*, 2(3):283–310, January 1989.

[2] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *The International Journal of Computer Vision*, 1(2), April 1987.

[3] N. Ayache and F. Lustman. Fast and reliable passive trinocular stereovision. In *Proceedings ICCV '87, London*, pages 422–427. IEEE, June 1987.

[4] H.H. Baker and T.O. Binford. Depth from edge- and intensity-based stereo. In *Proceedings 7th Joint Conference on Artificial Intelligence, Vancouver, Canada*, pages 631–636, August 1981.

[5] Umesh R. Dhond and J.K. Aggarwal. Structure from Stereo-A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):1489–1510, November-December 1989.

[6] Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In Giulio Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision*, pages 563–578. Springer-Verlag, Lecture Notes in Computer Science 588, May 1992.

[7] Olivier D. Faugeras, Tuan Luong, and Steven Maybank. Camera self-calibration: theory and experiments. In Giulio Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision*, pages 321–334. Springer-Verlag, Lecture Notes in Computer Science 588, 1992.

[8] Olivier D. Faugeras and Giorgio Toscani. The Calibration Problem for Stereo. In *Proceedings CVPR '86, Miami Beach, Florida*, pages 15–20. IEEE, June 1986.

[9] W. Forstner and A. Pertl. Photogrammetric standard methods and digital image matching techniques for high precision surface measurements. In Gelsema, E.S. and Kanal, L.N., editor, *Pattern Recognition in Practice II*, pages 57–72. Elsevier Science Publishers, 1986.

[10] Pascal Fua. Combining Stereo and Monocular Information to Compute Dense Depth Maps that Preserve Depth Discontinuities. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 1292–1298, August 1991.

[11] Pascal Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1), Winter 1993. Available as INRIA research report 1369.

[12] D.B. Gennery. *Modelling the Environment of an Exploring Vehicle by means of Stereo Vision*. PhD thesis, Stanford University, June 1980.

[13] W.E.L. Grimson. A computer implementation of a theory of human stereo vision. *Philosophical Transactions of the Royal Society of London, B.*, 292(1058):217–253, 1981.

[14] W.E.L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):17–34, 1985.

[15] B K P Horn. *Robot Vision.* MIT Press, Cambridge, Massachusetts, 1986.

[16] B. Hotz, Z. Zhang, and P. Fua. Incremental construction of local DEM for an autonomous planetary rover. In *Proc. Workshop on Computer Vision for Space Applications*, pages 33–43, Antibes, France, September 1993.

[17] R.E. Kelly, P.R.H. McConnell, and S.J. Mildenberger. The gestalt photomapper. *Photogrammetric Engineering and Remote Sensing*, 43:1407–1417, 1977.

[18] Yoshifumi Kitamura and Masahiko Yachida. Three-dimensional data acquisition by trinocular vision. *Advanced Robotics*, 4(1):29–42, 1990. Robotics Society of Japan.

[19] Quang-Tuan Luong, Rachid Deriche, Olivier Faugeras, and Théodore Papadopoulo. On Determining the Fundamental Matrix: Analysis of Different Methods and Experimental Results. Technical Report 1894, INRIA, 1993.

[20] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:283–287, 1976.

[21] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proc. R. Soc.*, B-204:301–328, 1979.

[22] Hervé Mathieu. A multi-DSP 96002 BOARD. Rapport Tecnique 153, INRIA, May 1993.

[23] Larry Matthies. Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-Time Implementation. *The International Journal of Computer Vision*, 8(1):71–91, 1993.

[24] Gérard Medioni and Ram Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31:2–18, 1985.

[25] H.K. Nishihara. PRISM, a Practical Real-Time Imaging Stereo Matcher. Technical Report A.I. Memo 780, MIT, Cambridge, MA, 1984. 31 pages.

[26] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search. *IEEE Transactions on PAMI*, 7, No 2:139–154, 1985.

[27] S.B. Pollard, J.E.W. Mayhew, and J.P. Frisby. PMF : a stereo correspondence algorithm using a disparity gradient constraint. *Perception*, 14:449–470, 1985.

[28] Luc Robert and Olivier D. Faugeras. Curve-Based Stereo: Figural Continuity And Curvature. In *CVPR91*, pages 57–62. IEEE, June 1991. Maui, Hawai.

[29] Roger Tsai. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In *Proceedings CVPR '86, Miami Beach, Florida*, pages 364–374. IEEE, June 1986.

[30] Roger Tsai. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, August 1987.

[31] T. Viéville, P.E.D.S. Facao, and E. Clergue. Computation of ego-motion using the vertical cue. *Machine Vision and Applications*, 8, 1995.

[32] T. Viéville and O.D. Faugeras. Computation of Inertial Information on a Robot. In Hirofumi Miura and Suguru Arimoto, editor, *Fifth International Symposium on Robotics Research*, pages 57–65. MIT-Press, 1989.

[33] T. Viéville and O.D. Faugeras. Cooperation of the inertial and visual systems. In T. Henderson, editor, *Traditional and Non-Traditional Robotic Sensors*, pages 339–350. Springer-Verlag, Berlin, September 1989.

[34] T. Viéville, F. Romann, B. Hotz, H. Mathieu, M. Buffa, L. Robert, P.E.D.S. Facao, O.D. Faugeras, and J.T. Audren. Autonomous navigation of a mobile robot using inertial and visual cues. In M. Kikode, T. Sato, and K. Tatsuno, editors, *Intelligent Robots and Systems*, Yokohama, 1993.

[35] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *The International Journal of Computer Vision*, 13(2):119–152, 1994. Also Research Report No.1658, INRIA Sophia-Antipolis, 1992.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399