Depth and Motion Analysis: the machine being developed within Esprit Project 940*

Olivier D. Faugeras and Rachid Deriche
INRIA Sophia Antipolis
2004 route des Lucioles
06565 Valbonne Cedex
France
Nicholas Ayache and Francis Lustman
INRIA
Domaine de Voluceau, Rocquencourt
BP 105, 78153 Le Chesnay
France
Ercole Giuliano
Elettronica San Giorgio
ELSAG S.p.A
Via Puccini 2
16154 Genova
Italy

Abstract

In this article, we describe some of the algorithms for Depth and Motion analysis which have been developed within ESPRIT project 940. Specifically we discuss edge detection, token tracking in sequences of images, and trinocular stereo. These processes are being implemented in hardware to form the core of the DMA machine which we are developing and will provide sophisticated real time Vision capabilities for a large variety of Robotics tasks.

1 Introduction

The Esprit Project 940, called Depth and Motion Analysis (DMA), is a european project involving a number of industrial and academic partners from three different countries: France, Great Britain, and Italy. The DMA project started in June 1986 with three main objectives:

1. Develop algorithmic solutions as general as possible to the general problem of three-dimensional dynamic analysis,
2. Implement in hardware a subset of these algorithms and construct a machine, the DMA machine, with a throughput sufficient for accomodating a wide range of Vision applications,
3. Demonstrate the feasibility of the approach for two applications:
   (a) The recognition, location, and handling of parts for an industrial assembly cell,
   (b) The navigation and exploration in an office environment of a mobile robot.

In this article, we concentrate on points 1 and 2. The reader interested in the latest results on point 2.b is referred to [AFLZ88]

During the first two years of this project, we have carried out a great deal of research on Stereo and Motion analysis. The Stereo work has been performed with the idea of obtaining accurate range maps to support surface and volume representations of the objects and the environment. The Motion work has been performed mostly with the idea of obtaining the egomotion of the robot, but also, and then it is based on Stereo, with the goal of obtaining a fairly detailed kinematic description of an environment in which not only the robot, but also other objects are moving. In both cases, the same tokens are extracted from the images and used as inputs to the Stereo and Motion processors.

This paper is then organized as follows: in sections (2, 3, 4), we describe the algorithms we have developed for extracting tokens, tracking tokens over time, and Stereo. In section (5), we explain the hardware structure of the DMA machine and the implementation of the algorithms described in sections 2, 3, and 4.

*This work was supported by Esprit under project 940

2 Extraction of tokens

The decision was rapidly made to focus on only one sort of token, namely edges because they can be reliably, robustly, and accurately extracted from sequences of images. Other tokens such as regions were not considered because of their much lower reliability. Intensity edges were thought to be sufficient for the applications in man made environments where textures, for example, play a lesser role.

Edges are detected by gradient computation, using the technique developed by Canny ([Can86]) and further improved by Deriche ([Der87]). In particular, Deriche has come up with a very simple recursive implementation of the filters which unfortunately has not been put in hardware for technological reasons. The major ideas of the method are as follows.

2.1 Computing the gradient

The gradient of the intensity surface $I(x,y)$ is the two-dimensional vector $\vec{G}(x,y) \quad [\frac{\partial I}{\partial x}(x,y), \frac{\partial I}{\partial y}(x,y)]^t$ and is perpendicular to the direction of the edge. It is defined only for smooth surfaces and we consider a smoothed version J of I, obtained by convolving it with a smoothing impulse response $K(x,y)$:

$$J(x,y) \quad I \otimes K(x,y)$$

The two-dimensional impulse response $K(x,y)$ is built from a one-dimensional response $h(x)$ derived by Deriche

[Der87] whose expresssion is:

$$h(x) \quad Axe^{-\alpha|x|}$$

where A and α are constants; α controls the width of the filter. This filter computes a smooth derivative, therefore the primitive $k(x)$ of $h(x)$ is the impulse response of a smoothing filter and we can take $K(x,y) = k(x)k(y)$. Since $h(x) \quad \frac{d}{dx}k(x)$, by applying the convolution theorem to $I \otimes h(x,y)$ we have:

$$\begin{aligned} \frac{\partial}{\partial x}(I \otimes K(x,y)) &= I \otimes h(x)k(y) \\ \frac{\partial}{\partial y}(I \otimes K(x,y)) &= I \otimes k(x)h(y) \end{aligned}$$

The gradient of J can be computed by convolving the rows (resp. the columns) of J with h and k.

The advantage of this approach is that the filtering operations required to compute the gradient of the smoothed image are separable (i.e. row, columns) and therefore one-dimensional which is extremely efficient in terms of computation. The disadvantage is that the smoothing is non isotropic: the directions 45° and 135° undergo more smoothing than 0° and 90° and it is expected that edges in these directions will be detected more accuratly than in others, the worst being the case of 45° and 135°

2.2 Finding edge pixels

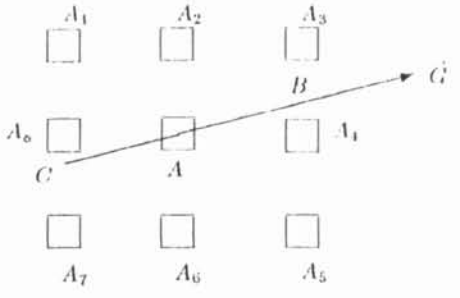
Having computed the gradient at each pixel in the image, potential edge pixels are selected by choosing those which are local maxima of the gradient norm in the direction of the gradient. This is of course a consequence of the of the fact that at an edge, the gradient is normal to the edge and its norm reaches there a local maximum along a cross-section taken along its direction.

When dealing with discrete images, to decide whether a pixel A is a local maximum of the gradient norm in the direction of the gradient $\vec{G}$ we have to do two things:

1. Decide on the size of the neighbourhood of A within which we wish it to be a maximum ;
2. Since the direction of the gradient does not fall necessarily on real pixels, we must interpolate the gradient values which are known only at the pixels.

In our implementation, the neighbourhood is taken to be the 3×3 neighbourhood of pixel A and gradient values are linearly interpolated. For example, in Figure (2.2), the gradient norm at pixel A is compared to the gradient norms at points B and C where they have been computed by linearly interpolating between the values at pixels A_3 and A_4 for B, and A_7 and A_8 for C. If the value at A is strictly more than the values at B and C, then pixel A is marked as an edge pixel, otherwise not.

In practice, this is not sufficient because too many edge points are detected, especially in regions where the contrast is low. If we want to cut down on the number of edge pixels, we must add some thresholding operation. Thresholding is a plague which pops up in many areas in engineering but is to our knowledge unavoidable and must be tackled with courage. Here the idea is to keep only the edge pixels for which the gradient norm is above some threshold T. If we set T too low, then too many pixels are still present and if we set it too high then connected chains of edge pixels start breaking into smaller chains which is highly undesirable. The basic question therefore is : how we choose T ? There is no good answer to this question and the choice of T must be guided by the application and the lightning conditions.



A variant of this thresholding idea has been introduced by Canny ([Can86]) and is called hysteresis thresholding. It is meant to keep the edge pixels connected as much as possible. To achieve this goal, we use two thresholds T_1 and T_2 with $T_2 < T_1$. If we think of a chain of connected edge pixels as a whole then, if for one pixel in the chain the gradient norm is higher than T_1, we will keep all pixels in the chain with a gradient norm larger than the lower threshold T_2.

2.3 Linking the pixels together

For reasons of compactness, and because the processes that come after the edge detection do not manipulate anything but the edge pixels, it is reasonable to abandon the image grid by linking together the connected edge pixels and keeping only the connected chains. These chains can be xtracted from the binary edge map produced as indicated in the previous section by a simple raster scan algorithm for which it is only necessary to explore the image once. The details of the algorithm are given in [Gir87].

2.4 Approximating the chains with line segments

As a means of reducing the information even further, we have implemented an algorithm that computes a polygonal approximation of the edge chains. This algorithm was originally developed by Berthod ([Ber86]).

3 Tracking Tokens

Given a sequence of images, in order to find the egomotion of the robot, one has to establish correspondences between tokens in images taken at different time ([FLT87]). This can be achieved by tracking moving objects in the scene. Our approach uses the line segments corresponding to the edges extracted from the scene (see section 2). However, it is worthwhile to note that other tokens such as points of interest (corners, triple points..) can be considered without affecting deeply the algorithm.

Our tracking approach is based on a combination of a prediction step and a matching process. Kalman filtering is used to help tracking by providing reasonable estimates of the region where the matching process has to seek for a possible match between tokens. This idea is quite general and has been used at several points in the DMA project (see [AF87,AF88], for example). Correspondences in the search area are found through the use of a similarity function based on features of the line segments. When working with a large sequence of frames, it is possible that some objects may appear or disappear totally or partially ; our Kalman filtering based approach allows us to handle this problem of occlusion in a effective way since it produces a prediction of the motion.

3.1 Prediction

Each token (i.e. line segment) is characterized by the following five parameters :

- The orientation θ of the line segment.
- The magnitude of the gradient along the line segment.
- The length L of the line segment.
- The distance of the origin to the line segment denoted by the parameter c.
- The distance denoted d, along the line from the perpendicular intersection to the midpoint of the segment.

A Kalman filter is used to aid tracking by providing reasonable estimates of the region where the matching process has to seek for a possible match between tokens. Kalman filtering is a statistical approach to estimate a time-varying state vector X_t from noisy measurements Z_t. For an excellent exposition, see [Jaz70].

Consider the estimation of $X_{t/k}$ from the measurements up to the instant t, Kalman filtering is a recursive estimation scheme designed to match the dynamic system model, the statistics of the error between the model and reality, and the uncertainty associated with the measurements.

In our approach, a Kalman filter is used separately on each of the five parameters defined above to estimate a state vector which is simply the time varying motion parameters of interest namely the position, the velocity and the acceleration for the parameters c and d, the angular position and its angular velocity for the parameter θ and only the position for the parameters length L and magnitude of the gradient G assumed to be constant.

The Kalman filter equations used in this paper involve discrete time steps. State vector notation is used such that $X_t^t = (x_t, \dot{x}_t, \ddot{x}_t)$ is the position, velocity, and acceleration of the parameter considered (c,d,θ,L,or G) at the t^{th} time step. The Kalman filter can be viewed as consisting of the following steps:

- The model of the system dynamics is:

$$X_t = \Phi_{t,t-1} X_{t-1} + W_{t-1}$$

The error of the model from reality is given by W_t, a zero-mean white Gaussian process of covariance Q_t :

$$E(W_t) = 0 \; ; \; E(W_t W_t^T) = Q_t$$

$\Phi_{t,t-1}$ is a matrix which evolves the position x, the velocity $\dot{x}$ and the acceleration $\ddot{x}$ from one time sample to another. It is easy to verify that assuming a motion with constant acceleration leads to the following matrix $\Phi_{\Delta t}$:

$$\Phi_{\Delta t} = \begin{pmatrix} 1 & \Delta t & \frac{(\Delta t)^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}$$

This is the well known equation of a dropped object for a time interval Δt.

- The measurement model used is:

$$Z_t = H_t X_t + V_t$$

where Z_t is a vector of measurements with an uncertainty V_t assumed to be a zero-mean Gaussian process of covariance R_t.

$$E(V_t) = 0 \; ; \; E(V_t V_t^T) = R_t$$

When H_t is the identity matrix, the measurements correspond directly to the state vector.

In our application, the measurement model Z_t assumes that the position x (i.e the value of c,d, θ, L or G) is measurable from the matching process while the velocity $\dot{x}$ and the acceleration $\ddot{x}$ are not. Therefore Z_t is the scalar corresponding to the position x and R_t is simply the uncertainty on x. Choosing this uncertainty in a manner reflecting our a priori estimate of the amount of noise to be expected from the previous step (Digitizing effects, edge detection and polygonal approximation) leads to deal with a small uncertainty for the parameters c, θ and G, and a large uncertainty for the unreliable parameters L and d due to the random effects which break line segments.

- State prediction:

$$\hat{X}_{t/t-1} = \Phi_{t,t-1} \hat{X}_{t-1/t-1}$$

After the measurement at time t-1 has been done, this time update equation predicts the system state at time t from the estimated values of the system state at time t-1. In our application, this equation predicts the value for x at time t from the informations available at time t-1.

- Covariance prediction for state vector :

$$P_{t/t-1} = \Phi_{t,t-1} P_{t-1/t-1} \Phi_{t,t-1}^T + Q_{t-1}$$

This equation gives the statistics relating the estimated state vectors to the unmeasurable ideal state vectors.

- Covariance prediction for the measurement vector:

$$U_{t/t-1} = H_t P_{t/t-1} H_t^T + R_t$$

This equation gives the statistics of the estimated model measurement. In our application, it determines the *search area* for the matching process.

- Kalman Gain:

$$K_t = P_{t/t-1} H_t^T [H_t P_{t/t-1} H_t^T + R_t]^{-1}$$

This equation indicates how much to weight each new measurement. Note that a small uncertainty R_t (precise measurement) causes a large weighting K_t and therefore leads to a corrected state estimate determined mainly by the measurement. A large uncertainty R_t causes a small weighting K_t.

- State correction :

$$\hat{X}_{t/t} = \hat{X}_{t/t-1} + K_t (Z_t - H_t \hat{X}_{t/t-1})$$

This equation is used to update the state model.

- Covariance correction :

$$P_{t/t} = P_{t/t-1} - K_t H_t P_{t/t-1}$$

This equation is used to update the statistics coupling the estimated state vectors to the unmeasurable ideal state vector.

- Covariance correction on the measures :

$$U_{t/t} = R_t (I - K_t^T H_t^T)$$

This equation is used to update the statistics coupling the estimated model measurement after the measurement at time t has been done.

In all the equations the "t_1/t_2" indicates an estimate at time t_1 after a measurement at time t_2 has provided more information. In our application, a new measurement is considered each time a correspondence has been done. All the above equations completely define the Kalman filtering we use in the prediction step.

3.2 Matching

For each token of the image model, a selection of the region where the matching process has to seek for a possible match between tokens is provided through the use a similarity function. This function uses the uncertainties provided by the Kalman filtering step on each estimate. It allows to keep only the tokens within the search area. A score for each correspondence is then calculated in order to disambiguate some possible multiple matches, using a difference measure. To this end, we use a normalized distance between the tokens, defined as a weighted sum of differences between the respective parameters values. Once a token from the current frame has been matched, we use its parameters as a new measure, first to update its corresponding state vector and second to provide the search area where the matching process has to look for a possible match between the image flow model and the next frame.

3.3 Bootstrapping step

First, at $t = 0$ and before the tracking algorithm can operate that is in the bootstraping step and each time that a new token appears, the matching algorithm has no idea where to look for the matching process. Therefore, a set of initial position velocity and acceleration guesses is used to initialize the tracking process. This is done by choosing somewhat arbitrarily values for the position velocity and acceleration but assigning large uncertainty so these initial values will not be weighted heavily as the measurement process continues. In our application, the position is set to be constant while the velocity and the acceleration are set to zero.

The search area is determined through a simple set of attribute tests using the result of the Kalman filtering. For each token of the image flow model, represented by a feature vector of 5 components, we wish to know which token might correspond to it. This is done through the use of a simple set of attribute tests using the current values of the measure, the expected value of the component and its uncertainty. This leads to calculate the Mahalanobis distance, explained below, for each components and to declare a token of the new frame inside a search area if all the distances are less than a fixed threshold.

The correspondence is controlled by the use of a normalized distance based on the attributes of each line segment. This distance works as a cost function. It is calculated for each possible match and the best score is used to validate the most consistent. In selecting a cost function for correspondences, we wanted to take into account the distance between the respective parameters values and the uncertainty associated with each parameter. A good measure is then the so called Mahalanobis distance. It weights each difference measure between the new token and the estimated token by the uncertainty of the estimated token. It is defined as follows:

Let each new token, issued from the matching process, be represented by a feature vector of N components denoted T_m and the estimated token represented by T_e with an uncertainty Λ. The Mahalanobis distance is then defined to be

$$M(T_m, T_e) \quad (T_m \quad T_e)^T \Lambda^{-1} (T_m - T_e)$$

The difference between T_m and T_e is easy to calculate for the length, norm gradient, c and d components. In order to deal with the problem of the difference between two angles θ_1 and θ_2, the cosine of the difference has been used.

3.4 Experimental Results of the Token Tracker

Experiments have been carried out with the proposed token tracking approach. The tracking algorithm has been applied to several sequences of real images taken from a mobile robot.

In fig. 1 to 3 are shown the extracted line segments on a sequence of 3 images.

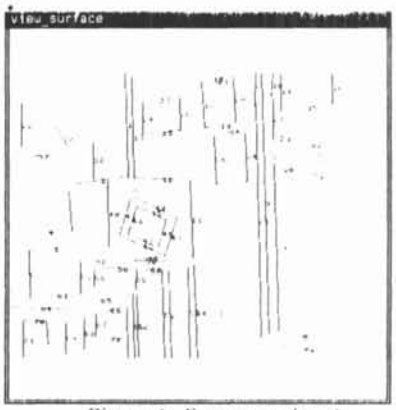


Figure 1: Frame number 1

Figure 2: Frame number 2.

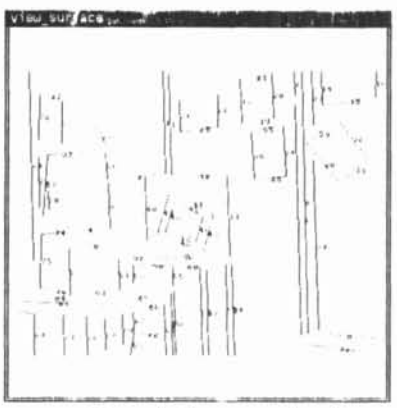


Figure 3: Frame number 3.

The tracking is illustrated through the numbers assigned to each line segment. A close look at the results reveals how some line segments can appear or disappear. A new label is affected as soon as a new segment appears and the process continues without affecting the tracking algorithm. A label corresponding to a good correspondence remains during all the process while false correspondences are removed after three frames generally. It should be pointed out that the al-

gorithm can cope with line segments moving with differents motions. This illustrates the efficiency of the used approach.

4 Stereo

4.1 Motivation

We developed at INRIA an algorithm for the matching of triplets of stereo-images. It has the following characteristics.

1. Inria Trinocular Stereovision method on indoors and industrial scenes (which is the primary interest for ESPRIT P940 Consortium) which have the peculiarity of having a lot of depth discontinuities.
2. It works faster than binocular stereovision, if one excludes the image preprocessing (edge extraction and linking) which must be performed in parallel for each image. Typically less than 2 seconds to match about 3 000 pixels.
3. It can be implemented in parallel in a straightforward manner.
4. It is more robust to errors than binocular stereovision because noisy or occluding boundaries which are often mismatched by binocular stereovision, are reliably checked by the third camera.
5. It is as general as the method for two cameras.

The entire trinocular approach as well as most of the obtained experimental results are reported in other papers [AL87a], and also in [AL87b]. Here, we give only a synthetic presentation of the trinocular approach and the results.

4.2 Geometry of Trinocular Stereovision

Figure 4 shows the geometry of trinocular stereovision. We have three cameras, modelled by an optical center C_i and an image plane P_i. Given a physical point P, its image on camera i is defined as the intersection of the straight line PC_i with the image plane P_i. Let us denote I_i, $i = 1,2,3$ the image of P on camera i. I_1, I_2, I_3 are called homologous image points.

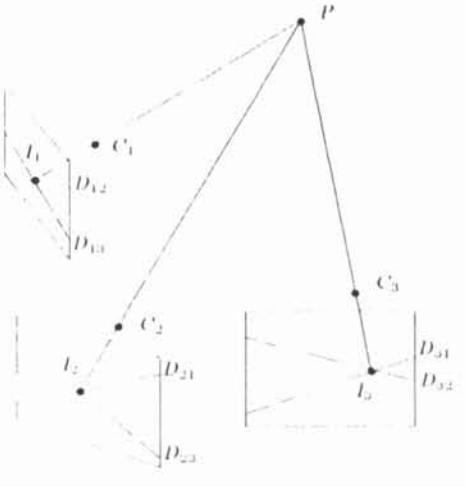


Figure 4. Geometry of trinocular stereovision

Given a pair (i,j) of cameras and a physical point A, the epipolar plane P_{ij} is defined by the triplet of points (C_i, A, C_j). The intersection of this epipolar plane with camera i is the epipolar line I_{ij}, while its intersection with camera j is the epipolar line D_{ji}. D_{ij} and D_{ji} are called conjugated epipolar lines. Any point I_i on D_{ij} (resp. I_j on D_{ji}) has its homologous image point I_j on D_{ji} (resp I_i on D_{ij}). Therefore, using two cameras, the search for homologous image points is a search along conjugated epipolar lines.

Any triplet (I_1, I_2, I_3) of homologous image points is such that I_1 lies at the intersection of the epipolar lines D_{12} and D_{13} defined by the two other image points I_2 and I_3. Therefore the search for homologous image points between two images can now be reduced to a simple verification at a precise location in the third image. For instance checking that (I_1, I_2) form a pair of homologous image points consists in verifying the presence of I_3 at the intersection of D_{31} and D_{32}.

4.3 Structuring the images

4.3.1 Tokens to be matched

The basic idea of the approach is to avoid to work directly at the pixel level but instead to extract visual tokens and to build a symbolic data structure from each of the images. These data structures can then be matched much more efficiently and safely than the original pixel grids.

On the choice of the extracted tokens, some important properties must be fulfilled. The tokens must be :

1. Compact, to allow for a concise description of the images, and to reduce the complexity of the stereo-matching algorithm.
2. Intrinsic, i.e. they must correspond to the projection of some physical object, in order to make the stereo-matching process meaningful.
3. Robust to the acquisition noise.
4. Discriminant, i.e. possess some properties which distinguish them from one another, to facilitate the matching process.
5. Precisely located, to allow for accurate 3D reconstruction.
6. Dense in the image, to allow for the computation of 3D points in every part of the scene.

We chose as tokens the linear segments coming from a polygonal approximation of the edges (see section 2).

In our opinion, these tokens fulfill very well the first 5 properties, and relatively well the 6th property, depending on the amount of texture in the observed scene. Anyhow, as it is not possible to derive meaningful 3D measurements within uniform region, we believe that these tokens represent an almost optimal solution.

4.3.2 Neighborhood Graph

We choose to structure the segments by a *neighborhood graph*. This data structure is useful for

- accessing the segments lying within a given region of the image and having a given orientation.
- a last consistency check to remove any remaining matching errors.

The neighborhood graph is defined by a set of nodes connected by edges. A node contains a segment and the geometric and luminance based properties of this segment. An

edge connect two nodes corresponding to neighboring segments in the image.

The computation of this neighboring graph is done in time linear with respect to the number of segments thanks to the use of *bucketing techniques*.

Buckets are computed in the following manner: we superimpose a virtual grid composed of square windows on the image and compute, for each window, the list of segments intersecting it. This structure is computed in linear time with respect to the number of segments. Moreover, at the expense of a unique preliminary sorting of the segments orientations (done with algorithmic complexity O(nLogn)), buckets are actually filled with segments sorted by increasing orientation. Accessing a segment in a given region of the image and having a certain orientation is then reduced to a dichotomy search in each bucket of the region.

The previous report shows an example of the computation of buckets and neighbors.

4.3.3 Matching Algorithm

The scheme of the algorithm is the following:

- Hypotheses Prediction and verification: triplets of potential matches are derived from the previously constructed graphs by simple geometric verifications, of the kind expressed in the previous section.

- Hypotheses Final Validation: local consistency checks are performed to remove erroneous matches at the end of the previous stage.

The algorithm works as follows (see figure 4):
for each segment S_1 of image 1 **do**

consider the segments S_2 of image 2 intersecting the epipolar line D_{21} corresponding to the middle I_1 of S_1 within a given disparity interval

if S_2's features (angle, length, gradient) are compatible with S_1's **then do**

compute the intersection I_3 of the epipolar lines D_{31} and D_{32} of I_1 and I_2 in image 3, and compute the predicted orientation of a matching segment;

if a segment S_3 lying within a given neighbourhood of I_3 whose orientation is compatible with the predicted orientation and whoses features are compatible with S_1 and S_2 **then**

keep (S_1, S_2, S_3) as a potential matching triplet;

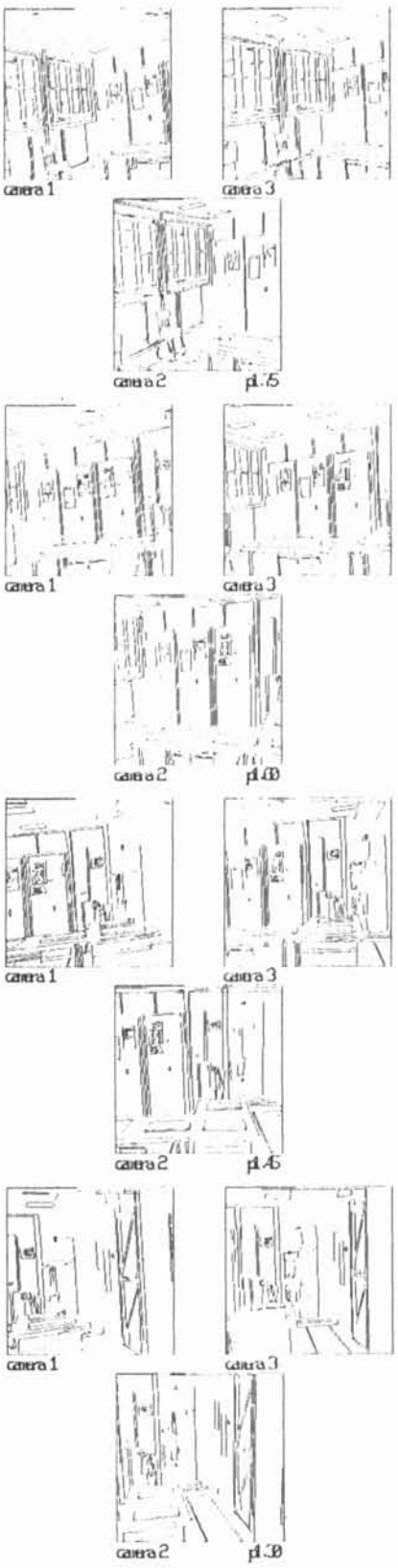
Discussion of the constraints enforced on each segment feature and of the meaning of *neighborhood*, as well as a description of the validation procedure can be found in [AL87a,AL87b].

4.3.4 Results

The stereo-matching program has been tested on several scenes including indoors and outdoors scenes and scenes with industrial parts and typical objets (sphere, cylinder, cone)

We describe first some indoors examples.

We take 6 different triplets of images from different positions of the mobile robot. Figure 5 shows the polygonal approximations of the edges.



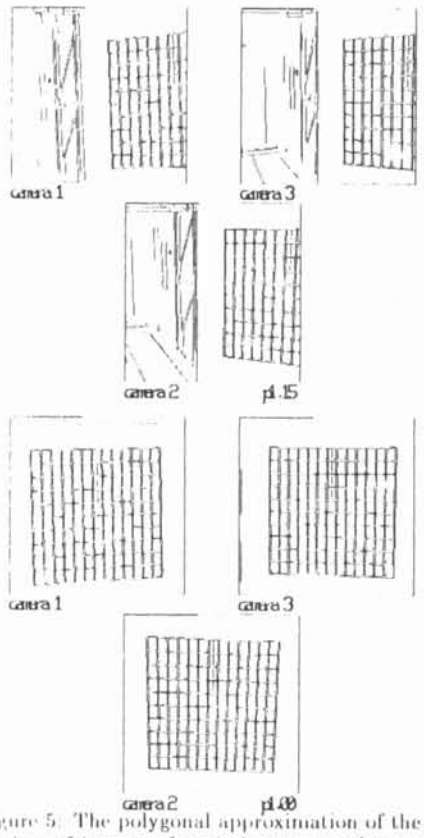


Figure 5: The polygonal approximation of the edges for six triplets of images of an indoors scene (part of the INRIA laboratory)

In general the number of errors is less than 2 percent of the total number of matches. As it is shown in figure 6 we have been able to compute a 3D description of the entire room from the 6 different triplets. This implies in particular that we have been able to compute the displacement of the cameras in the six different positions.

Finally, we also show results on a curved object:a cylinder (Figures 7 and 8). This example shows that the method still works acceptably on scenes with only curvilinear edges.

5 Hardware architecture

The above research clearly requires an application environment with real time capabilities, and within the project particular attention has been devoted to this target to achieve the availability of a performance suitable for quite demanding applications like a mobile vehicle moving in an indoors environment or a manipulating arm dealing with industrial scenes.

Although the project has covered both low and high level topics, the hardware implementation has been focused particularly on lower and intermediate levels of the process, so DMA may be defined as a front-end processor with number crunching capabilities in the range of the billion of operations per second. To achieve a full process implementation including also the higher levels, DMA is interfaced to a host multiprocessor (like EMMA2 [ABCC85] developed by EL.SAG or CAPITAN developed by MATRA) for most demanding applications.

For DMA development the basic starting hypothesis has been to define an "open" architecture, with the possibility of integrating also modules available from the market. Other major hypothesis have been modularity and flexibility of configuration, to achieve efficiency also with different real time requirements and permitting the implementation of additional improvements to the algorithmic chain.

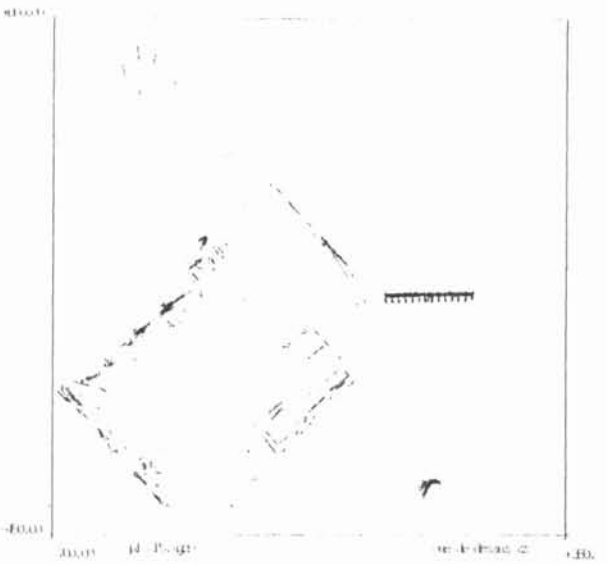


Figure 6: Top view of the laboratory obtained from the six previous observations

In terms of performance the reference target has been the TV video rate at least as a medium term goal, that means an architecture able to support such throughputs, while for the processing modules presently the range 5-25 frames/second has been considered as a good reference given the class of algorithms that have been investigated.

5.1 General architecture

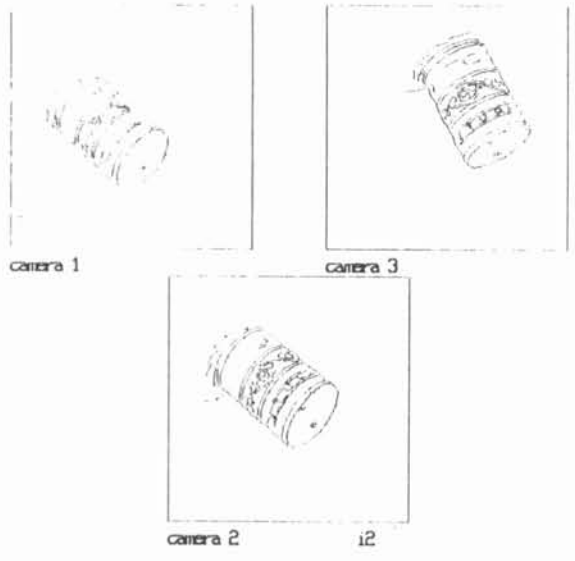
The hardware architecture is shown in Figure 5.10, for a configuration able to support 3 TV cameras; it is obvious that great attention has been devoted to the communication bandwidth, with three different link types:

- A general purpose bus: we chose VME for its high bandwidth associated to a worldwide diffusion, which implies a wide availability of commercial modules that may be included on the DMA; on the other hand this choice permits an easy integration of the DMA into several host environments.

  This bus is used for most images transfers concerned with format no longer "raster scan". Depending on the data rate requested by a specific application, the process can be spread on different VME buses.

- A video bus for high speed transfers of the images in raster scan format; here the choice has been for the compability with the Max-bus (developed by Datacube), that supports communications according to the RS170/CCIR standards as well as the transfers of portions of images (defined as Regions Of Interest).

- Some private inter-board links used for efficient local communications between identical modules that work in parallel.



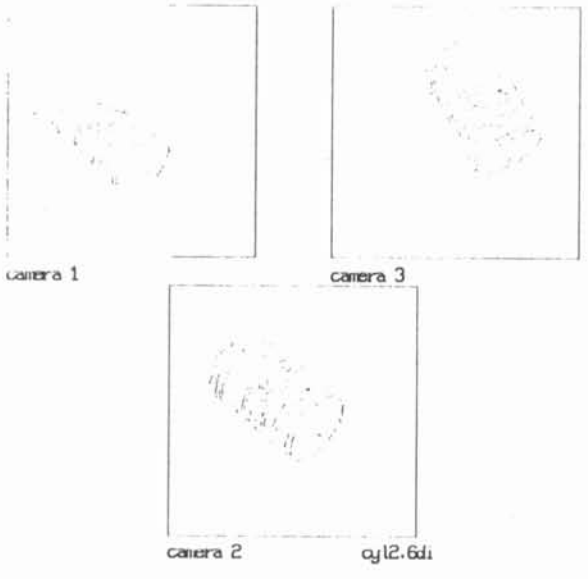


Figure 7: Edges (a) and stereo matches (b) of a triplet of images of a cylinder

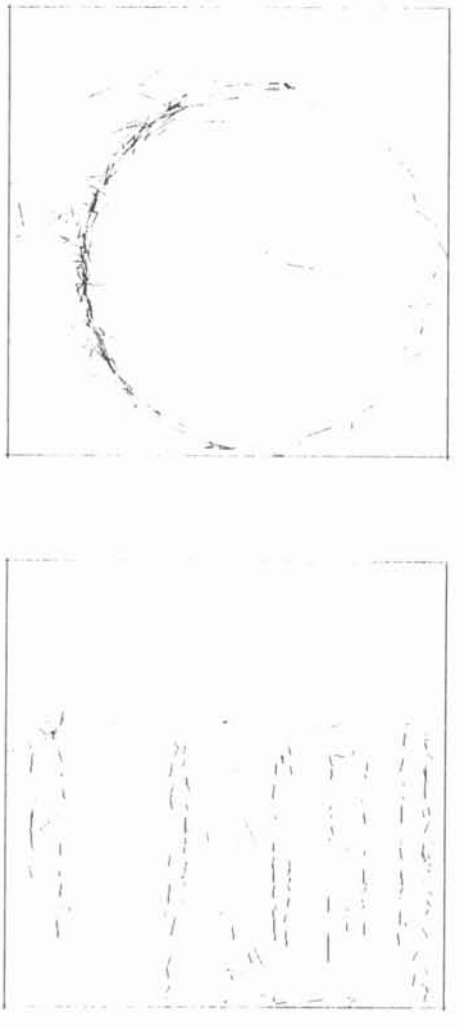


Figure 8: View from above (a) and from the side (b) of the 3D segments reconstructed from the stereo triplet

With the above communication system an overall throughput in the range of hundreds of MBytes/sec may be achieved.

Going now to a more detailed analys of the DMA processing elements, a first consideration is that the dedicated hardware has been limited to edge detection and token tracking, while the intermediate processing sections make wide use of programmable Digital Signal Processors to permit a good flexibility in the algorithmic implementation and maintenance.

We now discuss the different modules.

5.2 Cameras

Most requirements on this subject come from dynamic image acquisition of moving scenes, where the scene blur reduces the accuracy of the edge extraction process. To limit the blur, an effective solution is to adopt of cameras with an electronic shutter, something that is now supported by many manufacturers; the lower limit of exposure time is usually around 1 msec, to still guarantee an acceptable signal to noise ratio.

Another feature of fundamental importance is the compability with non-interlaced cameras, to achieve a full frame operation also in presence of motion, while with standard cameras it is necessary to work with fields to avoid image shifts between odd and even lines.

As a conclusion, the DMA supports RS170/CCIR standards as well as most common non-standard cameras with progressive scanning.

For the configurations with more than one camera, a common synchronisation system is provided.

5.3 Video Digitizer

DMA user video digitizers compatible with the above mentioned cameras and compliant to the Max-bus communication standard for image transfers.

The resolution is 8 bits/pixel (256 gray levels), and the acquisition chain includes features like input multiplexing, signal compensation, and direct implementation of point operations by means of look-up tables.

5.4 Multifunction Sequence Store

This unit is not involved in the real time operation, but is is devoted to the diagnostics and used for development purposes.

This module is able to download synthetic or known images through the VME bus, transferring them to the video bus and acquiring, if requested, results in raster scan format; another fundamental feature is the capability of acquiring a sequence of images up to 1 sec of duration, to debug motion algorithms.

5.5 Edge Detection

This function is being implemented on a couple of boards, achieving a full TV rate operation dealing with a Canny operator including a non maxima suppression to get edges with single pixel thickness.

For the Canny operator a FIR implementation has been chosen, due to the possibility of higher levels of integration with respect to IIR alternatives.

The first board is a general purpose FIR architecture including 4 1D filters with length up to 64 taps, dealing with 8 bit data and 8 bit coefficients; data coming from the digitizer through the video bus are filtered row by row with two masks (derivative and smoothing components, operating in parallel) and the results stored into an intermediate memory buffer, addressable by rows and columns; then the second phase of the filtering is performed, applying the last two masks to the intermediate results read column by column.

The results are available both on the video bus, for real time operation, as well as on the VME bus, mainly for debug purposes and/or operation directly on standard workstations. The output of the filter board is sent to the non-maxima suppression board, implemented at TV rate by means of a 3×3 neighboorhood analyzer devoted to the gradient maximum search.

5.6 Edge Linker

The algorithm to be implemented is characterized by two major compuational steps: edge chain creation and chain fusion. While the first phase is easily implementable in a parallel processing structure, the second one is essentially a global procedure, requiring a high performance of the processing elements (PE's).

The architectural choice has been oriented to a multi-DSP structure, to achieve at the same time high processing power, modularity of configuration and programming flexibility.

To speed-up the execution of specific primitives each DSP takes avantage of a coprocessor, implemented as a piggy-back board using SMT technology.

The DSP selected for this architecture is the Analog Devices ADSP 2100, a 16 bit processor with both program and data buses extended off-chip able to carry out multifunction instructions with a cycle time down to 80 nsec.

For edge linking implementation the coprocessor analyzes in a single DSP cycle the 3×3 neighbourhood of a pixel, triggering if requested a specific routine on the DSP for the subsequent chain functions (open chain, close chain, etc ...).

Each board includes 2 DSP units, with the possibility to build-up clusters of processors where a single PE is the cluster master and all other PE's are slaves.

Each DSP is equipped with 384 Kbytes of high speed data memory and 48 Kbytes of program memory.

The board has a flexible and high bendwith communication section, supporting 3 buses:

- VME bus
- Video bus
- Cluster bus (inter-PE link)

5.7 Polygonal Approximation and Stereo Matcher

Both these functions have requirements for high dynamic range and accuracy; that, together with good programming facilities have been the key targets in terms of implementation hypotheses.

As far as processing requirements go, these functions are quite demanding, but they may be decomposed into parallel processes with good efficiency, so the choice has been for a second multi-DSP architecture, based on the 24 bit processor Motorola DSP 56001.

This architecture is characterized by complementary features to the previous ones used for linking: in this case the possibility of coprocessors has been dropped, and we have preferred to increase the number of DSP modules on each board, 4 instead of 2.

In addition, the design has been carried out considering a future upgrade using the Motorola DSP 96001, which will add to the DMA machine the capability of floating point computations; this upgrade will take advantage of a software compatibility.

The module is designed for communication through the VME bus only, since it is devoted mainly to the processing of

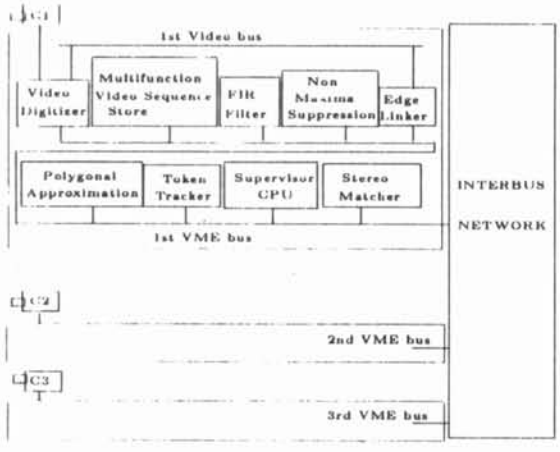


Figure 9: Esprit P940 - DMA Layout

images in symbolic representation, and broadcast transfers are supported.

Each DSP has dual port memories (access from both DSP and VME), with up to 576 Kbytes of data memory and 192 Kbytes of program memory.

5.8 Token Tracker

The token tracker works on symbolic data (tokens are line segments) and so it uses the VME bus as communication channel.

The processing structure has been defined according to the requested throughput of 10 frames/sec; this has forced us to choose a more dedicated approach, but even in this case the possibility of algorithmic improvement has been maintained.

The module has been implemented by a microcoded design, using a microcode memory addressed by a sequencer (Analog Devices AD1401) and providing as output the control of the arithmetic sections and of four address generators (Analog Devices AD1410) with related memory buffers.

The user(s) of the token tracker outputs depend on the global task that is implemented on the DMA machine; typical users are the stereo matcher (to simplify matching procedures) or an additional module computing the 3D structure directly from motion.

5.9 Supervision CPU

This module consists of a general purpose CPU with the function of taking care of system start-up at power-on, while during the normal operation performing functions of data routing and process synchronization.

5.10 Interbus Network

For the DMA configurations with different VME buses, an interbus link is requested; depending on the general system configuration this link may concern dedicated VME expanders or may be implemented using the communication paths of the high level processor; the last choice is quite convenient when the host is a multi-processor.

6 Conclusions

The DMA machine is being developed by the partners of the project 940; prototypes of the boards will be available at the end of 1988, and integration in the two parallel architectures will start mid 1989.

Acknowledgements The authors wish to thank here all the participants of the ESPRIT project 940 which have made writing this paper possible.

References

[ABCC85] Appiani, Barbagelata, Cavagnaro, and Conterno-Manara. EMMA2 - An Industry-Developed Hierarchical Multiprocessor for Very High Performance Signal Processing Applications. In *Proceedings of First International Conference on Supercomputing Systems*, 1985 St. Petersburg, Florida.

[AF87] N. Ayache and O.D. Faugeras. Building, registrating and fusing noisy visual maps. In *Proc. First International Conference on Computer Vision*, pages 73 82, IEEE, June 1987. London, U.K., also an INRIA Internal Report 596, 1986.

[AF88] N. Ayache and O.D. Faugeras. Maintaining representations of the environment of a mobile robot. In Robert Bolles and Bernard Roth, editors, *Robotics Research: the Fourth International Symposium*, pages 337 350, MIT Press, 1988.

[AFLZ88] N. Ayache, O.D. Faugeras, F. Lustman, and Z. Zhang. Visual navigation of a mobile robot: recent steps. In *Proceedings of Int. Symp. and Exposition on Robots*, November 1988.

[AL87a] N. Ayache and F. Lustman. Fast and reliable passive trinocular stereovision. In *Proc. First International Conference on Computer Vision*, pages 422 427, IEEE, June 1987. London, U.K.

[AL87b] N. Ayache and F. Lustman. Trinocular stereovision, recent results. In *Proc. International Joint Conference on Artificial Intelligence*, August 1987. Milano, Italy.

[Ber86] M. Berthod. Approximation polygonale de chaînes de contours. Programmes C, 1986. INRIA.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8 No6:679 698, 1986.

[Der87] R. Deriche. Using canny's criteria to derive an optimal edge detector recursively implemented. *The International Journal of Computer Vision*, 2, April 1987.

[FLT87] O.D. Faugeras, F. Lustman, and G. Toscani. Motion and structure from motion from point and line matches. In *Proc. First International Conference on Computer Vision*, pages 25 34, IEEE, June 1987. London, U.K.

[Gir87] G. Giraudon. *Chaînage efficace de contour*. Rapport de Recherche 605, INRIA, Février 1987.

[Jaz70] A.M. Jazwinsky *Stochastic Processes and Filtering Theory*. Academic Press, 1970.