# REAL TIME 3D LOCATION OF A CAR FROM THREE CHARACTERISTIC POINTS OBSERVED IN A VIDEO IMAGE SEQUENCE

E. MONTAGNE, J. ALIZON, P. MARTINET and J. GALLICE

*Université Blaise Pascal de Clermont-Ferrand, Laboratoire d'Electronique, U.R.A. 830 du C.N.R.S., F-63177 Aubière Cedex, France*

**Abstract.** The wide range of potential applications of vision techniques for autonomous vehicles has led the research world to develop efficient algorithms and to implant them in specific architectures to satisfy real time constraints. This paper describes a direct 3D location method used with a tracking algorithm based on the *smoothness of motion* of all features of the model. Both were implanted in a DSP processors vision machine giving real time information about the following model such as distance, orientation or next frame location.

**Key Words.** Image processing; target tracking; models; prediction; automobiles; real time computer system

## 1 INTRODUCTION

Vision techniques for autonomous vehicles have received considerable attention due to their wide range of potential applications such as vehicle guidance on road, obstacle detection and tracking, traffic scene interpretation...
The European project PROMETHEUS (PROgraM for European Traffic with Highest Efficiency and Unprecedented Safety) brings together different European car manufacturers, subcontractors and research laboratories in different fields of research. As part of this project, Le Laboratoire d'Electronique de Clermont-Ferrand is working on driver assistance systems, and various research programmes have been undertaken :

- estimation of the current position of a vehicle with respect to a normalized road (Chapuis *et al.*, 1991)
- automatic road following by an autonomous vehicle (Jurie *et al.*, 1993)
- obstacle detection and tracking with a range finder coupled with a camera (Alizon *et al.*, 1990; Xie *et al.*, 1993).

Obviously, efficient assistance to the driver involves real time constraints. Thus, the laboratory has developed original architectures using Transputer nets (Dérutin *et al.*, 1991) or DSP processors (Martinet *et al.*, 1991).

This paper describes a technique we have implemented on one of our systems. The aim of our work is to estimate in real time the location of a car from three points in order to inform the driver for his safety.

## 2 LOCATION AND TRACKING

### 2.1 *Location*

The pose estimation of an object consists in matching a part or all points of its 3D model with the 2D projection points and then to solve the position and the orientation of the object. Solving this with $n$ knowing points on the model and their projection is called the *n-point perspective problem*. For a complete estimation of the location, we need to take a minimum of 3 points. There are two important ways in locating an object, *iterative methods* and *direct methods*.
The aim of the first is the minimization of a criterion with a numerical algorithm. This criterion is the sum of the gap between the projected model and the computed model (Lowe, 1985; Tsai, 1987; Daucher *et al.*, 1993). These methods need an original pose estimation of the model which is

not too far from to the solution if we want a convergence of the minimization algorithm.
Direct methods use projective geometry and algebric technics to recover the pose of the object (Horaud *et al.*, 1989; Dhome *et al.*, 1989; De Menthon and Davis, 1990). Often, many mathematical solutions exist and it is necessary to sort them to retain the best one.

For our application, we have chosen a three point model using the algorithm of De Menthon and Davis (1990). The advantage of this method is that it is direct. It thus involves better computation speed than for an iterative approach, giving the best correspondence between a 3D model and its 2D projection.
This direct method uses an orthoperspective approximation. The model is a triangle of known dimensions. Two of the three vertices of the triangle are projected perpendicularly on a plane through the third vertex as shown in figure 1. This plane is perpendicular to the line of sight through the third vertex. These three points are then projected onto the image plane. By knowing the 3D triangle and the lengths and angles computed from the image plane, it is possible to compute the 3D location of these three points in the camera coordinate system.

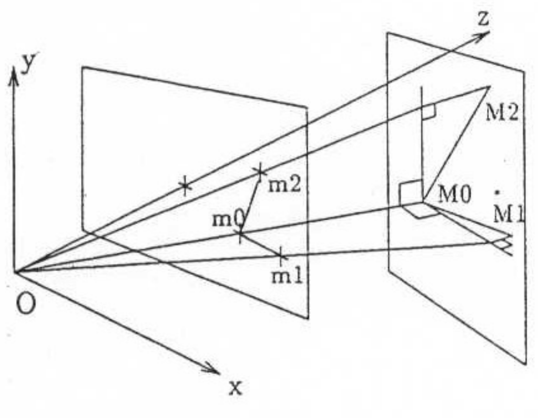


Figure 1: Orthoperspective Projection

### 2.2 *Tracking*

As we can localize a vehicle with the previous algorithm, we now want to follow it through an image sequence. This means that we are able to take from each frame the three image features which are projeted from the characteristic points of the model. So, we need to choose between the selected points if there are more than three of them or to predict some points if an occlusion has occured.
Several methods exist to locate the same physical point in different images. The main theme of this works shows that due to the inertia of a given point, its direction and motion will change little from one frame to the next. This *smoothness of motion* which precludes sudden change is often used to follow several trajectories individually (Jenkin, 1983; Sethi and Jain, 1985, 1987).
For our application, it is unnecessary to try to follow the same point in each frame with uniformity of motion, because the 2D trajectory is rarely continous but rather grouped in a part of the image. So, we prefer to try to follow the $N$ characteristic points of the model since they will have "homogeneous trajectories" ; there is no significant difference between the trajectory of one object point and another.
Our tracking algorithm requires knowledge of the location of the three previous computed models in each frame. With this information, we compute the 3D trajectory of the model and we predict the next 3D location of the model. Using the three previous centres of gravity of the located model, $P_{i-2}$, $P_{i-1}$ and $P_i$, and assuming constant acceleration and homogenous motion, we compute the ratio $k$ of angle between $P_{i-2}\widehat{O}P_{i-1}$ and $P_{i-1}\widehat{O}P_i$ in the plane passing through the three points. Then we compute the predicted point $P_{i+1}$ with the same ratio. From this centre of gravity of the model $P_{i+1}$, the 3D location of the $N$ characteristic points of the model is calculated and each point is now projected onto the image plane.
Around each projected prediction, a window of interest is defined. The sizes of the windows depend on the distance between the object and the camera given by the 3D prediction. All points in a window; named $W_i$, are selected to be the prolongation of the trajectory. If $s_i$ points are selected in window $W_i$, there are $T$ possible trajectories calculated as follows :

$$T = \prod_{i=1}^{N} s_i$$

with $N$ the number of characteristic points of the model. $T$ is the number of combinations of $N$ points each taken in a different window.
In each interest window (figure 2), we compute :

- the distance $d_{in}$ between the selected point $s_{in}$, $n$th point of the window $W_i$, and the prediction point $p_i$ of the same window
- the angle $\alpha_{in}$ formed by the vector $\overrightarrow{s_n p_i}$ and a reference vector chosen horizontally.

For the set of $T$ combinations of selected points, we compute a set of means of distances and angles as follows :

$$\overline{d_l} = \frac{1}{N}\sum_{i=1}^{N} d_{in}$$

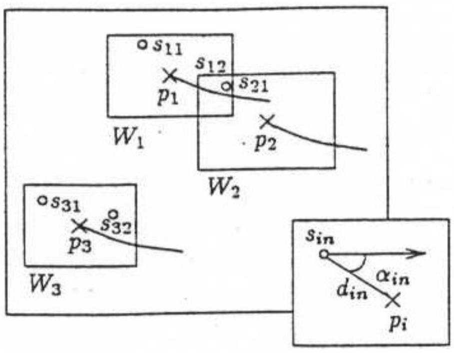


Figure 2: Interest Windows

$$\overline{\alpha_l} = \frac{1}{N}\sum_{i=1}^{N}\alpha_{in}$$
$$l \in (1 \ldots T)$$

with $d_{in}$ and $\alpha_{in}$ the previous calculated distances and angles for a selected point in a window $W_i$.
Then, the $N$ trajectories associated with the $N$ features of the model, $T$ *confidence rates* are computed by the formula :

$$\varepsilon_l = k_d\frac{1}{N}\sum_{i=1}^{N}\left|1-\frac{d_{in}}{\overline{d_l}}\right| + k_\alpha\frac{1}{N}\sum_{i=1}^{N}\left|1-\frac{\alpha_{in}}{\overline{\alpha_l}}\right|$$
$$l \in (1 \ldots T)$$

with $k_d$ and $k_\alpha$ given experimentally to privilege for example the distance if we consider that the notion of distance is more significant than the notion of angle.
If a *confidence rate* is lower than a threshold, we can tell that the selected points are the projected points of the model. If there is more than one trajectory which has an $\varepsilon_l$ below the threshold, we keep points which give the smallest $\varepsilon_l$.
We might have a configuration for which there is no satisfactory $\varepsilon_l$ for example when occlusion occured. In that case, we calculate the rate $\varepsilon_l$ again with $N-1$ points to find the $N-1$ points corresponding to the trajectories of the model. The missing point is substituted not by prediction but by another point which is placed with values $\overline{d_l}$ and $\overline{\alpha_l}$ in relation to the prediction.
If the number of characteristic points of the model is great, it is possible to reiterate the calculation of $\varepsilon_l$ $n$ times to find the $N-n$ trajectories in occlusion configuration. However, if theoretically $n$ can take the value $N-2$, the accuracy of the prediction points obtained with the calculation of $\overline{d_l}$ and $\overline{\alpha_l}$ on only two caracteristic points is less than that obtained from all the points if they had been present. So, if an occlusion continues for several frames, we might have a loss of tracking.

To prepare the tracking, we need to initialize the system using knowledge of the three previous locations. On a workstation, the three points selected for the first three frames are choose manually for the time being.

### 2.3 *Workstation results*

For our application, the three points of the model are the two rear lights of a car and a third light on its roof. For their easy detection, whatever the weather, we have equipped the camera with an infra-red filter fitted with the light wavelength. Thus the lights projected shine and the background becomes dark. Figures 3 represents some of trajectory results of the three point model in a 50 image sequence after initialization. This digital sequence displays two cars driving about on the road. One is the model, the second simulates some parasite lights. We can see from the result the homogeneity between the three trajectories and the stability of our algorithm after the crossing of the two cars.

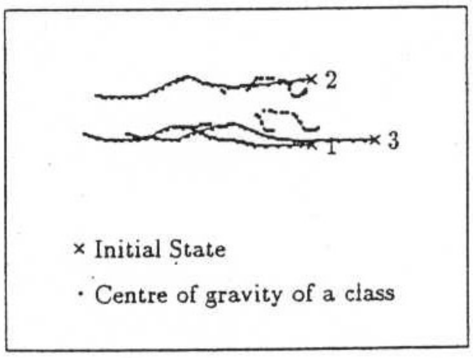


Figure 3: Frame x

## 3 REAL TIME IMPLANTATION

As results on the workstation seem correct, we have decided to introduce the algorithm on one of our specific architectures characterized by its modularity and its real time capabilities : WINDIS.

### 3.1 *Architecture overview*

This architecture, developed in collaboration with l'INRIA de Sophia Antipolis-France (Martinet *et al*, 1991); Rives *et al*, 1993)), implements the concept of active windows (figure 6). An active window is attached to a particular region of interest in the image and is required to extract a desired feature in this region of interest. At each active window, there is a temporal filter able to perform the tracking of the feature along the sequence. Several active windows can be defined at the same time in the image, and different processing can be done in each window. The functionalities of such an architecture are presented in figure 7. As stated above we are dealing with image processing algorithms which only request low or intermediate level processing. This aspect is clearly taken into account at the hardware level by using a mixed architecture :

- **Pipeline Architecture** is associated to the low level making efficient processing like convolution possible which is repetitive through the window. In practice, this is done by means of VLSI convoluors which admit up to a 5X5 mask size. This step can be performed in real time with duration depending on the mask size. The input of the algorithm is the active window and the values of the mask parameters corresponding to the desired processing. The output will be constituted by a list of pixels which potentially belong to the geometric features present in the window.
- **MIMD Architecture** has in charge the intermediate level. At this level, we find merging-like algorithms which use the list of pixels of interest provided by the low level stage and merge them to compute a more structured representation of the geometric features present in the window (for example, slope and distance to the origin for a line or coordinates of center and radius for a circle). In our hardware, this part is done by using multi DSP boards.

Windis architecture comprises three basic modules (corresponding to three different VME boards). Depending on the complexity of the application, the system will be built around one or more of these basic modules in a such a way that we will be able to attempt video rate performances.
The three basic modules are the following :

- **WINDIS Window Distributor Subsystem:** This module is used for window extraction, the execution of low level processing and the distribution of active windows toward the Window Processing Subsystem. Window distribution consists in dispatching lists of selected pixels and grey levels through the window bus.
- **WINPROC Window Processing Subsystem:** We have associated one to sixteen DSP 96002 modules with one distributor module. DSP modules are put together on mother boards and execute medium level processing on windows. Window processing modules provide a geometric description of the required primitive in each window. Four DSP modules can be used with each VME mother board.
- **WINMAN Window Manager Subsystem :** The window manager controls distributor and DSP modules, and executes high level processing of application tasks. Moreover, it is used for the tracking of the active windows throughout the sequence. A 68040 based cpu board implements this module.

For each level, we have introduced parallelism allowing us to reach video rate for most of the application tasks. All the modules satisfy VME requirements and are compatible with MAXBUS video bus from Datacube. The management of such a system is a tricky job and requires the use of a real time operating system. For facility reasons, we have chosen to develop all the system level under VxWorks O. S.

### 3.2 *Implantation - Results*

As for workstation results, images are dark with some shiny spots. Therefore, the WINDIS subsystem can extract a list of pixels simply by comparing brightness to a threshold after a Gaussian $5x5$ filtering. Then it scans the list and sorts the points into different classes. Next some information on classes such as centre of gravity, height, width and surface are computed. All the previous operations are performed by WINDIS in real time and information about classes are transmitted to the high level WINMAN sytem.
First, the high level needs to prepare the tracking by an initialization of the first three locations. So, in the first image, we could calculate all the attitudes of the model corresponding to each triplet of projected points. However, as there are $C$ combinations of $p$ points among $n$, the number $n$ of points must be low to limit the number of combinations. In order to reduce the number of calculated attitudes, we keep only triplets which are rear light projections near a horizontal line with the assumption that the following car is running normally.
Kept combinations called $C_1$ are given to the location algorithm to find their 3D attitudes. On the second frame, we take $C_2$ significant triplets again and with the assumption that no occlusion occured this time, we compute $T = C_1C_2$ *confidence rates* $\varepsilon_l$. Triplets which give $\varepsilon_l$ above a threshold are eliminated. The third frame is used to confirm remaining trajectory and to suppress some which are unsatisfactory. The best trajectory will be the initialization of the track-

ing. If there is no correct trajectory, we need to reinitialize the process.
The tracking module uses class information to define the size of the interest windows. Each has its height and width double those of the associated classes in the previous frame.
Figures 4 show an image of a sequence. The arrow, which has a variable base size dependent on the distance between the two cars, indicates the orientation of the vehicle. We see the interested windows centered around each spot of light. Figures 5 shows another image in which there are parasite lights to prove the efficiency of the tracking.

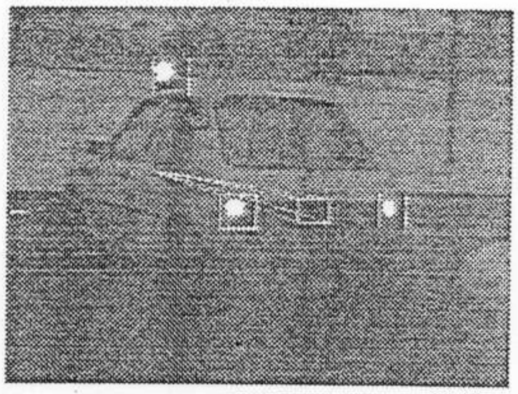


Figure 4: Road sequence

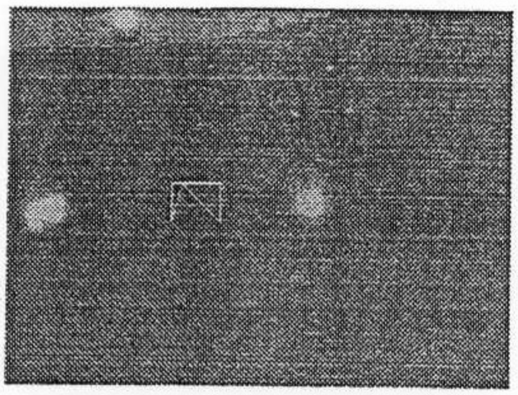


Figure 5: Road sequence with parasite lights

## 4 CONCLUSIONS - PERSPECTIVES

The capability of real time road image sequence for car tracking is a reality. Unfortunately, for the time being, we need to know the model of the car being followed, so it requires as many models as the number of cars we want to locate. Thus, we try to have an efficient trajectory estimate using the location of a mobile car with respect to another mobile car.
We hope to change the location algorithm by another direct method able to locate a car with or without partial knowledge of three points on the model. And the tracking initialization needs to be improved when there are severals spots of light to reduce the number of combination.

## ACKNOWLEDGMENTS

This work is supported by the European Eureka-PROMETHEUS project and is financed by PSA., Renault and the French Ministry of Research and Technology.

## 5 REFERENCES

Alizon, J., J. Gallice, L. Trassoudaine, and S. Treuillet (1990). Multisensory Data Fusion for Obstacle Detection and Tracking on Motorways. In : *Proc. of ProArt Workshop on Vision*, Sophia Antipolis, France, pp. 85-94.

Chapuis, R., J. Gallice, Jurie, and J. Alizon (1991). Real Time Road Mark Following on Motorways. *Signal Processing*, vol. 24, pp. 331-343.

Daucher, N., M. Dhome, J.T. Lapreste, and G. Rives (1993). Modeled object pose estimation and tracking by monocular vision. In : *Proc. of British Machine Vision Conference*, pp. 249-258.

De Menthon, D., and L.S. Davis (1990). Inverse perspective of a triangle : new exact and approximate solutions. In : *Proc. 90 Computer vision, ECCV 90*, pp. 369-373.

Dérutin, J.P., B. Besserer, and J. Gallice (1991). A parallel vision machine : TRANSVISION. In : *Proc. of Computer Architecture for Machine Perception CAMP91*, Paris, France, pp. 241-251.

Dhome, M., M. Richetin, J.T. Lapreste, and G. Rives (1989). Determination of the Attitude of 3D Objects From Single Perspective View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, N 12, pp. 1265-1278.

Horaud, R., B. Conio, and O.Leboulleux (1989). An Analytical Solution for the Perspective-4-Point Problem. *Computer Vision, Graphics and Image Processing*, Vol. 47, pp. 33-44.

Jenkin, M. (1983). Tracking three dimensional moving light displays. *Workshop on Motion Representation Control*, Toronto, Canada, pp. 66-70.

Jurie, F., P. Rives, J. Gallice, and J.L. Brame (1993). High speed vehicle guidance based on vision. In : *Proc. of the 1st IFAC International Workshop on Autonomous Vehicle*, Southampton, United Kingdom, pp. 205-210.

Lowe, D.G. (1985). *Perceptual Organization and Visual Recognition*, Chap. 7, Kluwer, Boston.

Martinet, P., P. Rives, P. Fickinger, and J.J. Borrelly (1991). Parallel Architecture for Visual Servoing Applications. *Workshop on Computer Architecture for Machine Perception, CAMP'91*, Paris, France.

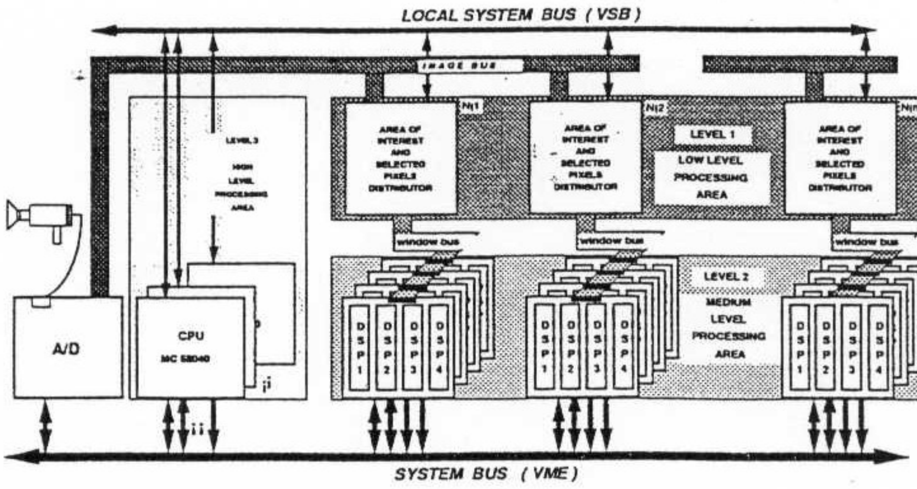
Rives, P., J.J. Borrelly, J. Gallice, and P. Martinet (1993). A Versatile Parallel Architecture for Visual Servoing Applications. *Workshop on Computer Architecture for Machine Perception, CAMP'93*, News Orleans, USA.

Sethi, I.K., and R. Jain (1985). Finding Trajectories of Feature Points in a Monocular Image Sequence. *Conference on Computer Vision and Pattern Recognition.*, San Fransisco, U.S.A., pp. 106-111.

Sethi, I.K., and R. Jain (1987). Finding Trajectories of Feature Points in a Monocular Image Sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 9, N 1, pp. 56-73.

Tsai, R.Y. (1987). A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. In : *IEEE J. Robotics and Automation*, Vol. 3, pp. 323-344.

Xie, M., L. Trassoudaine, J. Alizon, M. Thonnat, and J. Gallice (1993). Active and intelligent Sensing of road obstacles : application to the Eureka-PROMETHEUS project. *4th ICCV*, Berlin, Germany, pp. 614-623.



LOCAL SYSTEM BUS ( VSB )
IMAGE BUS
LEVEL 3
HIGH LEVEL PROCESSING AREA
CPU
MC 68040
A/D
AREA OF INTEREST AND SELECTED PIXELS DISTRIBUTOR
LEVEL 1
LOW LEVEL PROCESSING AREA
window bus
LEVEL 2
MEDIUM LEVEL PROCESSING AREA
DSP 1
DSP 2
DSP 3
DSP 4
SYSTEM BUS ( VME )

Figure 6: Overview of the general system

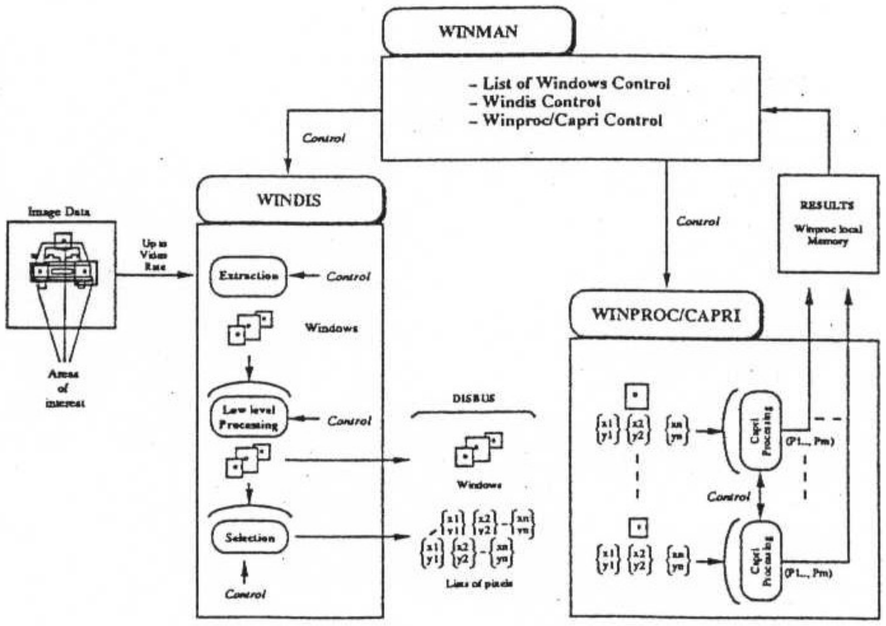


Figure 7: Data flow through Windis