Parallel Architecture for Visual Servoing Applications

Philippe Martinet, Patrick Rives, Pascal Fickinger, Jean-Jacques Borrelly

Parallel Architecture for Visual Servoing Applications

* P. Martinet, ** P. Rives, * P. Fickinger, ** J.J. Borrelly

* Laboratoire d'Electronique,
Université Blaise Pascal de Clermont Ferrand,
U.R.A 830 du CNRS,
Les Cézeaux, 63177 Aubière Cédex

** INRIA - Centre de Sophia Antipolis
2004 route des Lucioles,
BP 109, 06561 Valbonne Cédex

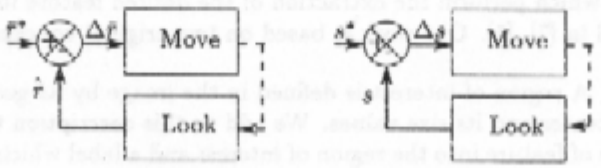
Abstract: Recent improvements in the field of image processing and robot control allow us to hope that using vision data directly in a closed loop control approach is no more an utopic way. This approach is often referred as visual servoing [12] or sensor based control [4]. The work described in this paper deals with the image processing problems occuring with such an approach.

After having briefly recalled the general problem of vision based control, we emphasize the consequencies on the architecture of the vision system. We point out the major difficulties induced by real time constraints and the processing of large dataflow from the sequence of images. An original vision system (WINDIS) based on a modular and parallel architecture concept which has been specially designed for vision based control applications, is then proposed and detailed. Finally some results in simulation are presented and future trends will be evoked.

1 Vision Based Control

Recent advances in vision sensor technology and vision processing now authorize the use of vision data as feedback in a closed loop control scheme. This approach is often referred as visual servoing [12] or vision based control [4]. We can distinguish two different ways to perform vision based control applications [13]: an indirect scheme based on a 3D position servoing, and a direct one based on a visual servoing:

- **3D Position Servoing**: the task consists to bring the camera to a desired position r^*, which means desired location and attitude between the camera and a frame linked to the environment (see Figure 1a). At each iteration of the control loop, an estimate $\hat{r}$ of the actual position has to be achieved from the vision data by building a 3D model of the scene. This scheme works in open loop with respect to vision data and cannot take into account sudden or large variations of the environment, inaccuracies and uncertainties occuring during the processing. Furthermore, such an approach needs to perfectly identify all the 3D models (sensor geometry, environment and robot models).
- **Visual Servoing**: in this approach, the task is directly specified in terms of regulation in the image (see Figure 1b). This requires the design of a set s of visual features which are sufficient and relevant for the completion of the task. Thus, a closed loop can be really performed from vision data, which allows to compensate for the perturbations by using a robust control scheme.



a) 3D Position Servoing b) Visual Servoing

Figure 1: Vision Based Control Schemes

The work described in this paper deals with the visual servoing approach. It is particularly well suited when the task can be expressed in a natural way as a regulation of the attitude (position and orientation) between a frame linked to the camera and a frame linked to the scene. Many application fields involve this kind of task : automatic vehicle guidance, space rendezvous and docking, target tracking and, in a more general way, several robotics applications. The basic idea consists to assume that a task is completed and well performed when a desired image of the scene corresponding to a particular viewpoint of the camera, has been reached. Implementing a visual servoing application needs to solve several problems. In a first step, we have to specify the task in terms of signals to be controlled in the image. The second step requires the choice of the image processing methods for extracting these signals. The last step consists to incorporate the informations from the vision system in a robust scheme performing the control of the motion of the camera. Concerning the first and the last aspects, we referred the readers to previous papers [9], [2]. In the present paper, we focus on the peculiar and difficult problem of image processing with regard to the visual servoing approach. From this point of view, the main constraints which we have to deal with, are the following :

- **Timing aspect** : In a visual servoing scheme, the sampling rate of the closed loop control is directly given by the image processing time consuming. Moreover, to ensure the performancies of the control, this sampling rate has to be sufficiently high. For the most current applications in visual servoing, a rate of 10 images per second can be considered as the lower limit.

- **Image processing aspect** : In a *scene analysis approach*, the main goal is to get high level description of a scene by means of sophisticated time consuming algorithms. Conversly, in a visual servoing approach, we want to only extract from the image the necessary and sufficient informations which ensure stability and convergence of the control scheme. Due to this fact, we only request low or intermediate levels processing providing contour based or region based features like points, lines, areas, center of gravity and so on. Moreover in the most cases, these features are located in some regions of interest and then, the processing of the whole image is not necessary.

- **Tracking aspect** : The features extracted from the image have to be tracked along the sequence. We have also to take into account appearing and disappearing features due to the motions of the camera and objects in the scene.

These requirements involve the use of specific hardware and software architectures having powerfull real time capabilities and modularity. Presently, only a few available industrial product may satisfy these constraints with relatively low performancies with regard to their high cost. Concerning research products, a new generation of image processing systems appears using Transputer nets [3], DSP processors, or dedicated processor like Sympati [11]. Unfortunately, these systems are often designed for general trends in image processing and do not take into account specificities of the visual servoing approach. Due to this fact, we developped a new architecture (hardware and software) specially dedicated to this kind of applications. Our approach consists to split the problem in two levels. The first one is devoted to the tracking of features along the sequence by means of active windows associated to the regions of interest. The second one is to attach to these windows some dedicated image processings (hard and soft) which perform the extraction of the desired feature into each window. Similar approaches can be found in [7], [5]. Our work is based on two original concepts :

- **Active window** : A region of interest is defined in the image by its geometric parameters : the coordinates of its center and its size values. We add to this description two symbolic parameters describing the type of feature into the region of interest and a label which will be propagated from one image to the next during the tracking along the sequence. For simplicity reasons, each region of interest is modelled by a rectangular window defined by $(x_g, y_g, xsize, ysize, feature, label)$. In the most current applications in visual servoing, we will have to track a few window (less than

16) but the parameters of the window could have large changes along the sequence (ie, in an assembly task, we detect a hole in the border of the image and want to center it in fullscreen). The management of the active windows along the sequence is implemented as a recursive filtering using knowledge about the camera motion for updating the windows parameters.

- **Image processing** : As previously noted, we only address the problem of low or intermediate processing levels providing simple geometric features. This kind of processing involve two steps. The first one is often referred as a segmentation step and consists of partitioning the pixels in the window in coherent sets (with respect to a criterion). This partitioning uses only *local* properties in the image and can be, in many cases, implemented as a convolution between a precomputed mask (generally 3X3 or 5X5) and the image. This step can be performed in real time with a delay directly function of the size of the mask. 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 potentialy candidates to the geometric primitive. This first step can be seen as a *splitting* stage. The second step consists of merging the pixels in the list to obtain a representation of the geometric primitive like slope and distance to the origin for a line or coordinates of center and radius for a circle. Different methods can be used to perform this step. We investigate three of them : space transform approaches like Generalized Hough Transform, mean square methods and recursive filtering (Kalman).

2 Overview of the System Architecture

WINDIS architecture is partitionned in three levels necessary to attempt video rate processings. Figure 2 gives the system configuration. We have defined several subsystems:

- **Window Distributor Subsystem (level 1)**: this module has in charge the window extraction, the execution of low level processing and the distribution of windows. Window distribution consists to dispatch list of selected pixels and grey levels through the window bus.
- **Window Processing Subsystem (level 2)**: we have associated one to sixteen DSP modules with one ditributor 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 searched primitive in each window.
- **Window Manager Subsystem (level 3)**: window manager controls distributor and DSP modules, and executes high level porcessing of applications tasks. A 68030/40 based cpu board implements this module.

For each level, we have introduce parallelism allowing us to reach video rate for most of applications tasks.

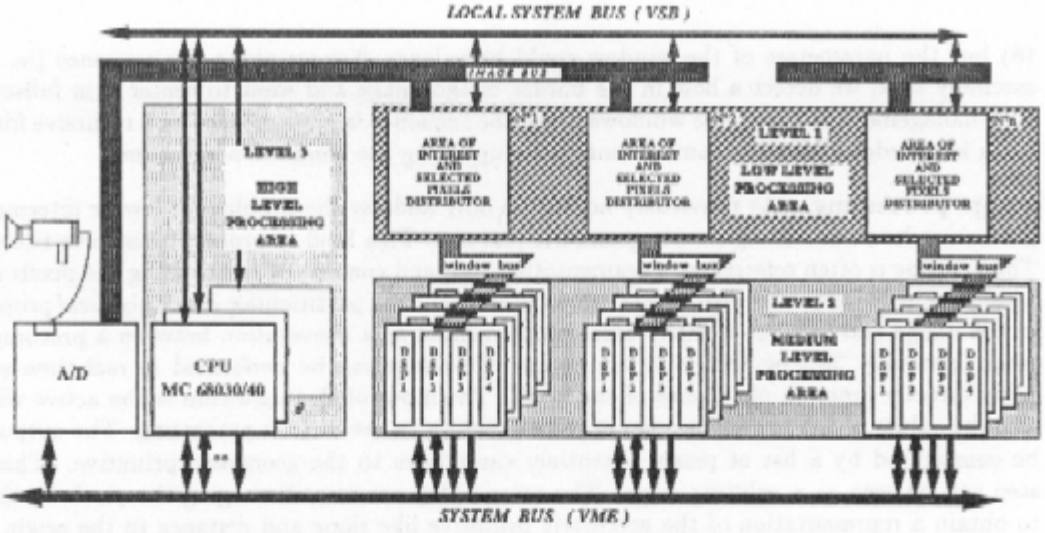


Figure 2: WINDIS : general architecture for window processing

The mains characteristics are now pointed out:

- **Mixed architecture :** Low level is built as a MISD (Multiple Instruction Single Data) machine with regard to the different active windows and allows us to perform efficiently processings like convolution which are repetitive through the window. Intermediate level is implemented as a MIMD (Multiple Instruction Multiple Data) architecture and authorize flexibility with respect to the application: the same processing can be executed on the whole set of the window or, conversely, different processings can be applied to the same area. To reach efficiency, at each processing level, the memory banks have been duplicated and multiplexed; the first bank stores datas from the previous level while the second supplies data for the processing in progress.

- **Modular architecture :** According to the number of regions of interest, their size and the desired time for the processing, the hardware and software capabilities have to be modulated. In the proposed architecture (WINDIS), a single low level can drive one to sixteen intermediate level modules and this elementary structure can be duplicated if necessary. Moreover, several areas of interest can be processed by a same intermediate module and a dynamic distribution management is used to optimize the processing time.

- **Dynamic windows management architecture :** In dynamic vision problems, the sizes and the locations of the windows are time varying. Moreover, features can appear and disappear in the image due to the motion of the camera or of the objects in the scene. These requirements are fully supported in WINDIS architecture. A mechanism of prediction and estimation of the current parameters of the windows is provided by the high level module and allows us to reactualize all the parameters of the different processings attached to a window in real time. Note that the efficiency of this mecanism is mainly due to the facility of simultaneous programming of low level parameters and distribution of the windows on the intermediate level modules.

According to this approach, we have designed a prototype version of WINDIS (figure 3) including:

- one A/D video signal conversion board
- one Window distributor board
- one Window processing subsystem
- one cpu board

At time, we have designed window distribution and processing modules. Their functionalities are now detailed

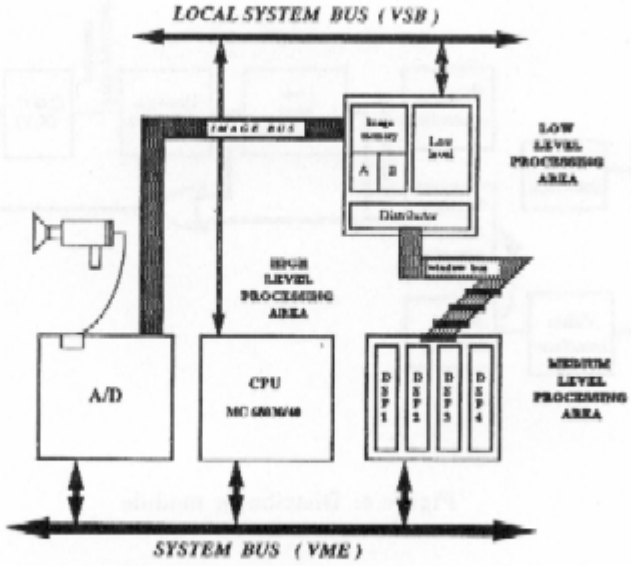


Figure 3: Developped system

3 Description of the Different Subsystems

3.1 The Window Distribution Subsystem

The fonctionalities of this module are the following: image data storage, low level processing on the whole window and distribution of list of selected pixels potentialy candidates to geometric primitive. At the same time, window distributor dispatchs grey level pixels of processed - or not - windows through the window bus. Window distributor module (figure 4) consist on a standard VME/VSB board which includes four stages:

- **Data storage:** At video rate, image memory stores image (N) while window memory provides windows from image (N-1) which are processed and dispatched through the window bus. At each video frame cycle, image and window memories datas banks can swapp. Size of this memories is fixed to 256Kx8 bits.

- **Low level processing:** Windows controler sends extracted windows to low level processing stage where two chips (IMS A110 from INMOS) can execute one of the four following processings at video rate: 1D convolution, 3x3 or 5x5 convolution, or two 3x3 convolutions in parallel.

- **Decision:** For each windows, to extract a list of pixels candidates to geometric primitive, we program a selection criterion. Decision results from a tresholding which can be single, double (min/max) or with hysteresis: fuily programmable logic implements these fonctionalities. In addition, this stage allows to detect maximum and minimum grey levels on the whole window or row by row.

- **Distribution:** Grey levels of processed - or not - windows are send to the DSP modules through the window bus. At the same time, the coordinates generation unit (C.G.U.) dispatchs list of the address of pixels belonging to the primitive.

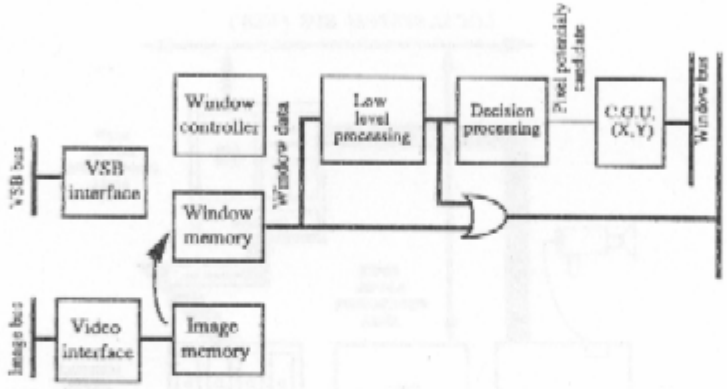


Figure 4: Distributor module

3.2 The Window Processing Subsystem

The window processing subsystem consists in a VME board supporting up to four DSP modules which can be plugged on it . Figure 5 represents a general overview of the window processing subsystem.

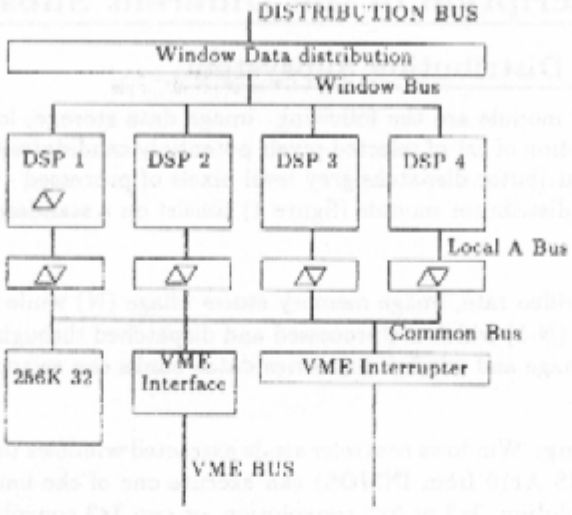


Figure 5: Window processing subsystem overview

VME Board The VME board receives the window data from the Window Distribution Subsystem through the window bus. One or more DSP modules of a selected board may simultaneously receive the window data.

The board also has a 256K by 32 bits shared memory used by the DSP modules and the VME host to communicate. Is also has VME master slave interface and can generate a single VME interrupt. The VME interrupt level and vector are fully programmable.

DSP modules A DSP module (Figure 6) is based on the DSP96002 signal processor running at a 33 Mhz clock. The module has two window data storage areas, a local 256k by 32 bits memory for program and internal data storage.

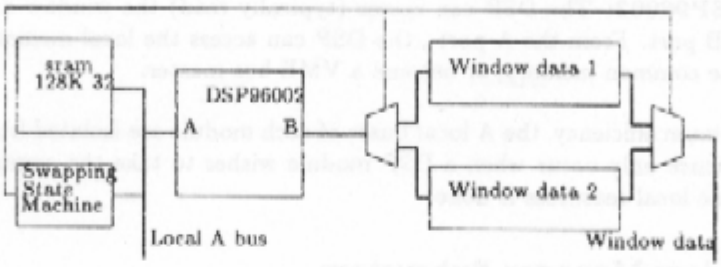


Figure 6: DSP module

The DSP processor has two external general purpose buses (A and B) that can be used as data or program buses. The mapping of the three address spaces used for data and programs of the DSP can be separately done on each bus. The DSP module architecture takes advantage of the internal parallelism of the DSP96002 by routing the program and the image datas on two separate buses. The two window data areas provide a way of pipelining the DSP activity. At any time, one area is filled with the datas of one or more windows, while the other is processed by the DSP. The swapping between the two areas is externally controlled by window manager when the DSP module is ready to process new data and there is no running distribution. The swapping is realized by a state machine activated by an external access to a dedicated address on the A bus.

3.2.1 Window data areas

The two window data areas of a module are made of:

- a 256K by 8 bits memory used to store the window pixels information, sequentially distributed windows will be placed at increasing addresses. This memory is written by the window distribution and read by the DSP96002.
- a 16K(32K) by 18 bits FIFO memory used to store the X Y coordinates of the pixels of interest in the window. These pixels are determined by the low level system after some filtering or edge detection. The X Y coordinates are absolute in the image and coded on 9 bits. Pixels of interest of subsequent windows are also store in the fifo.
- a 1K by 9 bits fifo memory used to store the actual number of pixels of interest for each distributed window

When the swapping of the two window data areas occurs, the new distributed fifos and address generation unit are cleared.

3.2.2 Data path

There are three possible communication paths:

- **From the Distribution bus:** The data coming from the window bus are write only, they can be directed simultaneously to the currently distributed window data area of one or more DSP modules.
- **From the VME bus:** A VME bus requester can access the shared memory on the VME board but also the module local memory (i.e. for program downloading) and the swapping state machine of each module. The DSP Host Interface is also accessible by a VME requester.
- **From the DSP96002:** The DSP can access (typically read) the window data which are connected to the B port. From the A port , the DSP can access the local memory without any bus arbitration, the common memory, or become a VME bus master.

To insure a maximum efficiency, the A local buses of each module are isolated from each other. The bus arbitration sequence only occur when a DSP module wishes to take the common bus or when a request to the module local resources is done.

3.3 The Window Manager Subsystem

Window manager subsystem is a software computed on a 68030/40 cpu board, it provides two kind of control :

- **Window distribution control:** for each window, window manager defines and programs different parameters as window size, window position, low level processing parameters, and selection criterion. Window manager allocates one DSP module for intermediate level processing. During current window distribution, window manager anticipates the parameters programmation of the next window. Then, when current distribution is completed, window manager orders the next distribution without delay.
- **Window processing control:** for each DSP module, window manager initializes windows processings parameters (number of windows, geometric feature to be extracted,...) through VME bus and asks for the processing. DSP modules point out the end of processing to manager: then, results are available for high level processing and can be read through VME bus.

Window manager schedules intermediate level processings with respect to a priority distribution list. To establish this list, window manager calculates computer load with the following parameters: window size, number of windows to be processed, and choice of intermediate level processings. Scheduling occures with fully load balancing through the medium level processing area. At the end of the whole processings, window manager order to swapp image memory in the distributor module: a new distribution can occur.

4 Performance Analysis

4.1 Context of the Evaluation

To evaluate the performance of this architecture, window manager software was written in C language. We design a model of this architecture and simulate the algorithms often used in such applications. This model includes only one window distributor and one to sixteen DSP modules. We evaluate execution time on workstation (68030 based cpu). Simulation results are described in the following sections.

4.2 General Performance Evaluation

In a first part, evaluation concerns low and intermediate level processing. We have chose maximum of gradient detection and Hough transform for the corresponding processing levels. Number of distributed windows and number of DSP modules are considered as parameters of the simulation. Windows size is fixed to 64x64 pixels.
We have simulated architecture for the processing of different number of windows. The following figures concern the processing of 64 windows :

- In figure 7, the simulation results show that running time decreases with the number of DSP modules but video rate is never reach. Three plots are presented
  - plot (a) : running time takes into account Tc (Communication time with one DSP)
  - plot (b) : running time does not take into account Tc
  - plot (c) : video rate

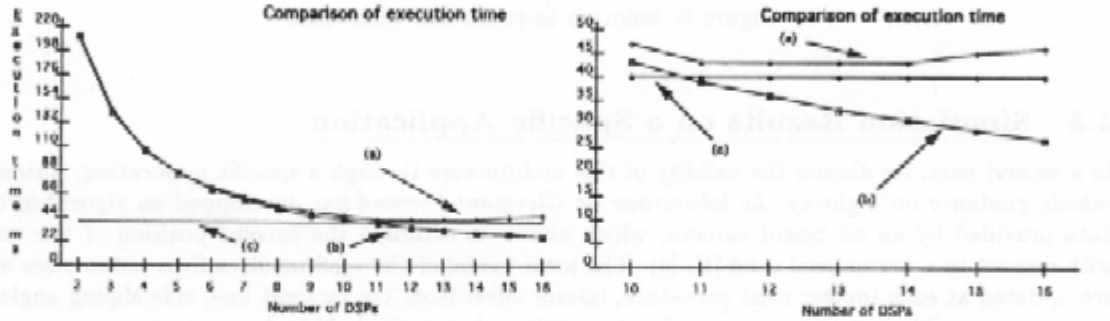


Figure 7: Distribution of 64 Windows 64x64

- figure 8 gives the evolution of the Efficiency (E) and of the Gain (G) (for the definitions of Efficiency and Gain, see [8])

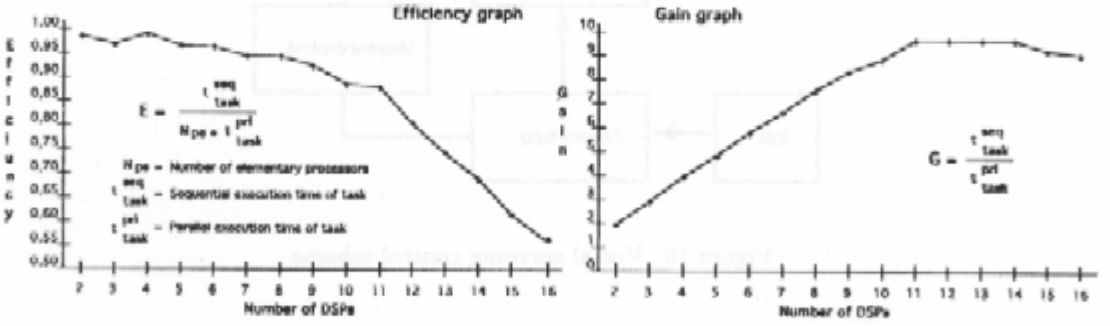


Figure 8: Efficiency and Gain of WINDIS architecture

Architecture limits appeared in the plots between 11 to 14 DSPs which is closed to theorical limit defined by Nopt = Te/Tc (Te : running time of one 64X64 window). In our case, Te = 6,5 ms and Tc = 0,5 ms then theorical limit is : Nopt = 13. This limit can be greater or equal to 16 if Te is greater or equal to 8 ms.

- figure 9 shows how to execute processing of 64 windows at video rate. To reach this aim, it is necessary to divide processing of 64 windows into 2 processings of 32 windows. As shown in figure 9, processing execution time has a value of 43 ms (point A) and become 37 ms (point B). In this case, two windows distributors are necessary and each of them is associated with 6 DSPs.

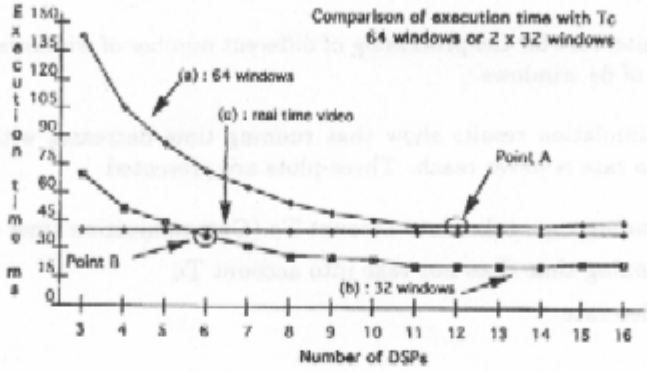


Figure 9: Solution to reach real time video

4.3 Simulation Results on a Specific Application

In a second part, we discuss the validity of this architecture through a specific application: automatic vehicle guidance on highway. *Le laboratoire de Clermont-Ferrand* has developped an algorithm using data provided by an on board camera, which allows to estimate the current position of the vehicle with respect to a normalized road [1], [6]. The local model of the road involves fives parameters which are updated at each image: road curvature, lateral offset from the straight line, side sliping angle and two parameters characterizing camera location (tilt and elevation). Vehicle parameters are given by a Kalman Filtering (figure 10):

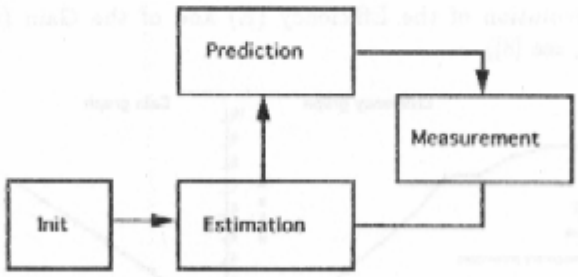


Figure 10: Visual servoing control scheme

- **Prediction step** : From the (k - 1) th position of the vehicle, high level module predicts in the image the locations of several areas of interest close to the road lines.
- **Measurement step** : Into each area, a gradient algorithm is performed row by row. It provides a list of maxima points which are merged by means of a Hough transform.
- **Estimation step** : From the measurements, the parameters of the road model are updated.

Simulation of this architecture has been performed and its performancies has been evaluated. In this simulation, we take into account the distribution algorithm involving several windows with various sizes and the processing times correponding to image algorithms and data transfert on the bus. Only the image processings correponding to measurements step has been considerated. For the simulation needs, we selected 30 simultaneous windows per image with different sizes (30x30, 40x40, 45x45, 50x50 and 64x64). The results obtained from simulation show that :

- All the windows are processed during only one video frame cycle (less than 30 ms)
- Only single distributor module and four DSPs modules are sufficient. The computer load is fully balanced over these four modules : 7 to 8 windows are processed by each modules.

In conclusion, WINDIS architecture has great capabilities in real time video processing. Modularity in low and intermediate level processing allows us to defined an Adapted Applications Configuration. When elementary processing execution time is too small, a limit appeared in this architecture due to the excess of communication time (Tc) compared to elementary processing execution time (Te).

5 Conclusion

Simulation of this architecture has been performed and its performances has been evaluated. In this simulation, we take into account the distribution algorithm involving several windows with various sizes and the processing times corresponding to image algorithms and data transfert on the bus. Only the image processings corresponding to the measurement step have been considerated.

Our project is now in its design phase and first results are expected next year. The cards satisfy to VME standard requirements. Two boards are presently in development : the first one implements all the low level processings as defined above, the second, based on DSP processors, is devoted to the intermediate level processings. Each intermediate level processing board can receive up to four DSP based modules.

References

[1] R. Chapuis *Suivi de primitives images, application à la conduite automatique sur route*, Ph-D Thesis, University of Clermont Ferrand, France, janvier 1991.

[2] F. Chaumette: *La relation vision-commande: théorie et applications à des tâches robotiques*, Ph-D Thesis,University of Rennes I, France, July 1990.

[3] J.P Derutin, B. Besserer, T. Tixier, A. Klickel: *A Parallel Vision Machine: TRANSVISION*, CAMP91, Computer and Machine Perception, Paris, December 1991.

[4] B. Espiau. *Le point sur la commande proximetrique*, Technique de la robotique, tome II, Hermes, Paris 1988

[5] H. Inoue, H. Mizoguchi: *A Flexible Multi Window Vision System for Robots*, 2nd International Symposium of Robotics Research, 1984.

[6] F. Jury, R. Chapuis, J. Gallice, J. Alizon: *Mobile Position Estimation in a Dynamic Environment*, RFIA-AFCET Congress, Lyon, November 1991.

[7] H. Kubota, K. Fukui, M. Ishikawa, H. Mizoguchi, Y. Kuno: *Advanced Vision Processor with an Overall Image Processing Unit and Multiple Local Image Processing Modules*,MVA'90 IAPR Workshop on Machine Vision Applications, Tokyo, 1990.

[8] G. Randall: *Conception et réalisation d'une machine de Stéréoscopie Trinoculaire Passive Temps Réel*, Ph-D Thesis, University of Paris 6, October 1991.

[9] P. Rives, F. Chaumette, B. Espiau: *Visual Servoing Based on a Task Function Approach*, First International Symposium on Experimental Robotics, Montréal, Canada, June 1989.

[10] C. Samson, B. Espiau, M. Le Borgne: *Robot Control: the Task Function Approach*, Oxford University Press, 1991.

[11] *SYMPATI 2: SYstème Multiprocesseur Adapté au Traitement d'Images* Document Technique, SIR/CEN de Saclay, 1991

[12] L. E. Weiss *Dynamic Visual Servo Control of Robots. An Adaptive Image based Approach*, Technical Report, CMU-RI-TR-84-16, Carnegie Mellon

[13] L. E. Weiss, A. C. Sanderson: *Dynamic Sensor-Based Control of Robots with Visual Feedback*, IEEE Journal of Robotics and Automation, Vol. RA-3, n. 5, Oct. 1987.