MVA '92 IAPR Workshop on Machine Vision Applications Dec. 7-9,1992, Tokyo

Contour Based Image Segmentation Process on a Parallel Vision Machine

P. Legrand, J.P. Dérutin

Laboratoire d'Electronique, U.R.A 830 du CNRS,
Université Blaise Pascal, 63177 Aubière Cedex, FRANCE
email : legrand@lesmti.univ-bpclermont.fr

Abstract

Image processing applications need shorter processing times. This requires the parallelization of low and mid levels sequential process chains on specific machines. In this paper we present the implementation of a contour based image segmentation process on our parallel vision machine *Transvision*. Two approaches for the contour coding steps are discussed.

1 Introduction

Edge extraction is an important step in a system of image processing. High and mid-level processings require short processing times and accurate results from the edge extraction process (good localisation, one reponse to one edge...). The different approaches based on optimisation of criteria seem to fulfil qualitative requirements. This improvement leads to an increase of the complexity of calculations. On traditional sequential computers the use of egde extractor based on optimisation of criteria is too much time consuming for applications requiring short response times.
This paper deals with the implementation of a contour based image segmentation process on a heterogeneous machine based on two architecture levels, *pipe-line* and *distributed memory MIMD* models.
This segmentation process can be split in five different stages :

- directional gradients processing (R. Deriche),
- local maxima detection (R. Deriche),
- hysteresis thresholding (R. Deriche),
- contour coding (we developped this algorithm),
- polygonal approximation of coded countours (Duda and Hart).

2 Sequential algorithms

2.1 Edge detection and extraction

The computation of the directional derivative images is achieved by using the Deriche filter [1], [3]. The next step consists in a local maxima detection. Then the hysteresis thresholding is used. During this last step, the pixels detected as local maxima are tagged according to the value of their gradient magnitude :

- tag=TRUE if $\| G \| > t_h$,
- tag=FALSE if $t_l < \| G \| < t_h$

with t_l : low threshold and t_h : high threshold

2.2 Contour coding

This process is necessary to link points belonging to the same part of each contour. The connected contour pixels are linked, and only their coordinates are stored. By this process, we ouput a set of lists. During the step of local maxima detection and hysteresis thresholding, the coordinates of each contour pixel were stored in a structured list. To organize this list, the abscissa will be considered first and organized from the lowest value to the highest one. When there is more than one point abscissa of the same value, the ordinates will be organized in the same manner. We associate an index in order to increase the reading time of this list. This index contains the suffixes from which the differents points of researched abscissa can be found. In this method, we arbitrary give a higher priority level to 4-connected pixels, and look only for "the future", according to the structure of the list. Consider the figure 1.

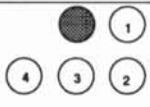

Figure 1: Neighbourhood.

The first unlinked point of the list will be called *current point*. We will then test the next point of the list to determine if it is 4-connected to the current point (point 1 in the figure 1). If it is 4-connected, it will become current point. Otherwise, we will scan the next set of points of the next row of the image using the index. If a 4-connected point (point 3) is found it becomes current point. Otherwise, we look for a 8-connected point (point 2 or 4) which if found will become current point. If not, the process on this specific chain will stop. This operation is reiterated on the points that has not been previously identified as part of a chain. At this stage we may eliminate isolated points, or small chains of points. A logical OR between the set of connected points obtained indicates if the chain of points must be stored or deleted (hysteresis thresholding principle). At the end of this process, we obtain different lists of connected points.

2.3 Polygonal approximation

This algorithm was first introduced by Duda and Hart [6]. It consists in a recursive segmentation of a curve, and an approximation of the curve by straight lines.

3 Parallelization

3.1 The Transvision machine

A required major feature for the "On-board Mobile Vision systems" is to solve very hard temporal constraints :

- one or several on-line sensors deliver data at real-time video rate,
- a real-time control process (which includes a full chain of image processing : digitalization, low, intermediate and high level image treatments),
- control process.

These systems which allow us to solve this temporal constraints,

require the design of parallel architectures of dedicated machines. In this paper, we use a parallel vision machine concept based on two combined architecture models (*pipe-line* and *Distributed Memory MIMD*) and a "data-bridge" between them, called *Video Node* [4]. The name of this project is *TRANSVISION*.
The pipe-line level is upstream from the MIMD structure and can be reduced to a video digitalization module. From this concept, we create a "test machine" which allows us to define the best suited target machine to a specific application.
The "test-machine" links three principles together :

- The Pipeline model is composed of real-time video rate modules. These heterogeneous modules are designed with DSP's or ASIC's chips, which fit well the low-level image processing. These operations are generaly regular using local data.
- The Distributed Memory MIMD model which computes the object-list created before, is composed of processing elements (Pe) like the Transputer Txxx chips. The Pe's network can be reorganized with program. This model is really adapted to the intermediate and high levels of image processing.
- The "data-bridge" achieves the synchronisation and the communication between the two models above. It is based on :
  - a T800 transputer,
  - a dual ported video RAM,
  - a special hardware to realize the synchronization between the two worlds (this synchronization is based on a *OCCAM rendez-vous*).

In fact, these two modules realize different transformations on different kinds of data, in an asynchronous way.

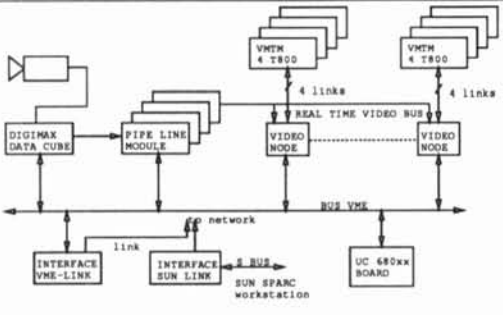

Figure 2: The Transvision architecture.

In order to solve accurately the temporal constraints of the embedded applications from this test machine, the target module is defined by establishing the type and the number of pipeline modules, the number of elements and the topology of the Pe's network, the communication modes and the number of Video Nodes, which could resolve the temporal constraints of the application.
To illustrate this approach, we present a real-time application from an image sequence : a contour based image segmentation.

3.2 Problem position

This segmentation process is implemented at the stage MIMD of the Transvision machine allowing a coarse grain parallelism. Because the goal of this work is mainly to evaluate a parallel scheme on a MIMD structure, we don't use the pipe-line level of the Transvision machine.
This segmentation process can be divided into various sequential or independant tasks as shown in the figure 3.

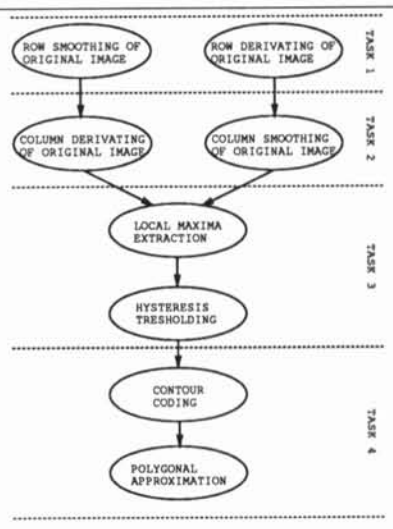

Figure 3: Tasks graph.

The sequential process is split in coarse grain tasks which are implemented according to a pipe-line method on the MIMD structure. The global processing time will be equal to the greatest task processing time ; but the results are available with a delay due to the number of pipe-line stages.

The input parameter of *task 1* is the original image. Because the Video Nodes have access to the original image data flow at the video rate, we assign them *task 1*. The different calculations are row dependent ; so we choose an horizontal data sharing between the different Video Nodes. They have to be synchronized. In order to balance the communications, we choose to connect the processors in a ring(refer to figure 8).

The input parameters of *task 2* are the images resulting from *task 1* (smoothed and derivated). During this process, we have to proceed to a column filtering (vertical data dependency). To do so, the result images have to be split in vertical bands. Consider the figure 4.

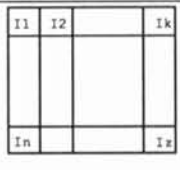

Figure 4: Image sharing.

The first horizontal band of the original image (I_1, I_2,.., I_k) is filtered by NV_1, the second band by NV_2 and so on. In order to perform the vertical filtering of the vertical band (I_1,..,I_n) on the second level of the pipe-line, the processors P_{11},..,P_{1n} used at this stage, must have in their own memory the smoothed and derivated image bands (I_1,..,I_n). We then require communications with the transputers NV_1,..,NV_n which have in memory the horizontal bands of results. The topology of the transputers network of the second level (P_{1i}) has to be a ring (as shown on the figure 8), because they have to exchange with each other different parts of the horizontal bands of images. Each processor P_{1i} is linked to the NV_i. The communications between NV_i and P_{1i} will occur in two steps :

1. NV_i send the horizontal bands of results to P_{1i},
2. the transputers P_{1i} exchange the portions of bands with each other.

The use of transputers allows us to hide the communications. A transputer is able to manage the parallel execution of several processes by time sharing ; this way it is possible to establish a communication process with another process simultaneously (*parallel*). We proceed as follow to compute the directional derivative images :

Processors NV_i :
PARALLEL
horizontal filtering of I(t+dt)
send R(t)

Processors P_{1i} :
PARALLEL
receive R(t)
vertical filtering of R(t-dt)
send $G_{x,y}(t-2dt)$

R : smoothed and derivated image bands.
dt : highest calculation time.

As the communication time between NV_i and P_{1i} is lower than the two filtering times, the communications will be hidden (according to the principle of pipe-line). The directional derivative images G_x and G_y are available at a rate equal to the longuest processing time , with a delay due to the propagation of data trough the network.

Task 3 is shared into two local tasks (local maxima detection and first step of hysteresis thresholding). It takes place at the next level of the pipe-line. The processors $P_{21},..,P_{2n}$ used at this stage receive the G_x and G_y images from $P_{11},..,P_{1n}$. The extraction of the local maxima needs to know the 8 neighbourhood points of the result image from the second level. This way, a data sharing is also possible. But it is necessary to create an overlapping between the bands of the directional derivatives G_x and G_y of one column for each side of the vertical bands. This exchange needs a network of processors organized as a ring (refer to figure 8).
During this process, once a local maxima is detected, the corresponding point is tagged according to the hysteresis thresholding method (refer section 2.1). Storing or deleting points is achieved during the contour coding step of *task 4*. At the end of this process, we have determined the list of contour points tagged *TRUE* or *FALSE*. For each vertical band of the image, we obtain a list and its associated index.
In order to shorten the processing time we can reduce the number of tests and the number of operations. To reduce the number of tests we only will threshold the local maxima pixels. To reduce the number of operations, we will first have to estimate the gradient magnitude of the pixel, and determine if the pixel can be a local maxima by comparing its gradient magnitude to the low threshold.

process of $\| P(i,j) \|$
IF $\| P(i,j) \| >$ *low threshold*
process of local maxima
IF P(i,j) local maxima
IF $\| P(i,j) \| >$ *high treshold*
store (i,j) in LIST
tag=TRUE
ELSE
store (i,j) in LIST
tag=FALSE
END IF
ELSE
don't care
END IF
ELSE
don't care
END IF

Note :
The amount of operations is greatly reduced, but the processing time is strongly dependant of data.
We will hide the communications the same way we did during the directional derivatives task. At the end of this task, the transputers output the structured list and the associated index.

Task 4 (second step of hysteresis thresholding, coding contour and polygonal approximation) is executed at the last level of the pipe-line, using the transputers $P_{31},..,P_{3n}$. Each processor P_{3i} operates from the list of points sent by its father P_{2i}. The second step of the hysteresis thresholding takes place during the contour coding. This way, chains of points can be broken as shown in the figure 5 a, and a loss of data may appear.

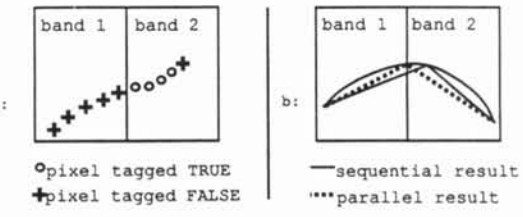

Figure 5: Chains broken by data sharing.

The section of the chain within a band (band 1 on figure 5 a) will be deleted if all of its points within this band are tagged *FALSE*. Even though one or more points of this chain are tagged *TRUE* in another band. Moreover the results output by the sequential polygonal approximation process can be different from those obtained with a sequential process, as shown on the figure 5 b. We have presented in [7] a simplified approach ; each processor works on its associated image band. In order to obtained parallel results closed to the sequential results we have to proceed to a merging step of the broken chains. This merging process will need the production of a new set of lists. Consider a ring composed with N processors. The image is divided into N bands. Each processor P_{3i} works on the list of contour points of the associated image band. Many cases of broken chains may appear as shown on the figure 6.

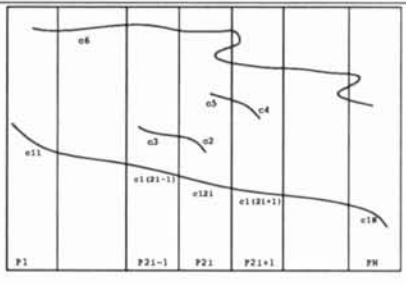

Figure 6: Various broken chains.

The chains of type $c1 = c11 \cup c12 \cup ... \cup c1N$ or $c6$ will require the maximum number of merging steps. This can be done with $log_2 N$ steps by using a merging strategy based on a bintree structure. The merging of the broken chains of the whole image will occur in two stages :

First stage :
P_i send broken chains to P_{i-1} (for $i \in [2,n]$), according to the following rules, (refer to figure 6) :

- P_{2i} sends broken chains of type $c_{1,2i}$ and c_2 to P_{2i-1},
- P_{2i+1} sends broken chains of type c_4 to P_{2i}

To obtain this bintree structure, P_{2i-1} stores in its memory the chains of type $c_{1,2i-1}$ et c_3 for merging with the chains received from P_{2i}.

Then each processor $P_1, .., P_{N-1}$ merges the broken chains ; P_N has finished its work.

Second stage :
This process allows the merging of chains which are dispatched on more than two bands. The merging of a chain belonging to the N image bands will need $log_2 N - 1$ merging steps. This can be described by :

step $\leftarrow$ 1
WHILE $(\frac{N}{4*step} - 1) \geq 0$
FOR $i \in [0, \frac{N}{4*step} - 1]$

$P_{step(4i+1)}$ sends broken chains to $P_{step(4i+2)}$

$P_{step(4i+3)}$ sends broken chains to $P_{step(4i+2)}$

$P_{step(4i+2)}$ merges the broken chains.

$step \leftarrow step + 1$

If we take $N = 16$, the graph of merging steps is shown on the figure 7.

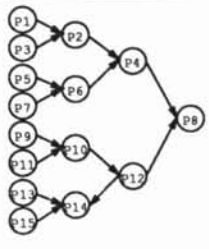

Figure 7: Merging steps.

At the end of this process each processor has in memory its set of lists of chains of connected points. The polygonal approximation process can start. We can note that the ring of processors P_{3i} is not well load balanced because the chains of points are not fearly distributed between the processors, for the contour coding and the polygonal approximation steps.
In order to have a well load balanced ring of P_{3i} for the polygonal approximation process based on a split method, we can use the processors farm model [5] to implement a parallel version of this algorithm. To do this, we have to manage a recursive generation of tasks which is discribed in [10].
On the other hand, the first step of *task 4* (second step of hysteresis thresholding, contour coding step and merging step) needs another parallel approach to obtain a best tasks mapping on the processors P_{3i}. But this will induce a largest processing time and will not be efficient enough.

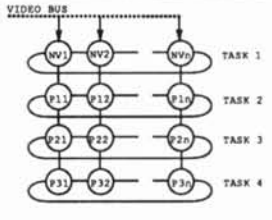

Figure 8: Network of transputers.

3.3 Network extension

Part of our problem is to evaluate the maximum number of transputers we can assign to each stage of the network. We reach this limit when communication time becomes greater than the calculation time, as the communications are hidden. We face a second restriction ; the number of transputers can not exceed $min\{N_{row}, N_{column}\}$.
The communication rate is the highest between the first and the second stages of the torus (between (NV_i) and (P_{1i})).

The communication time on a transputer link is equal to :

$$T_c = \alpha + \beta\{number\ of\ bytes\} \quad (1)$$

First, the Video Nodes send two bands of results to the P_{1i}. If there are N processors NV_i, the size of a band is equal to $\frac{n^2}{N}$ (for an image of size n x n). Because the data is coded as *real 32 bits* the communication time is equal to :

$$t_{c1} = 2 * \{\alpha + 4\beta\frac{n^2}{N}\} \quad (2)$$

Then, the processors P_{1i} interchange parts (of size $\frac{n^2}{N^2}$) from the two result bands. $\frac{N}{2}$ steps are required for this exchange ; that is why the communication time is equal to :

$$t_{c2} = \frac{N}{2} * 2\{\alpha + 4\beta\frac{n^2}{N^2}\} \quad (3)$$

And the global communication time is :

$$T_c = t_{c1} + t_{c2} = \alpha(N+2) + 12\beta\frac{n^2}{N} \quad (4)$$

The sequential processing time of one of the tasks (*task 1 or task 2*) is equal to T_s. As we use a simple data sharing parllelization method for each of those tasks, the processing time for N processors is :

$$T_N \approx \frac{T_S}{N} \quad (5)$$

We are now able to evaluate the number of processors of stage 1 (stage 2 and stage 3), for a required processing time.
As we use transputers, the communications between the processors may be hidden. That is to say, we must always have a communication time lower than the processing time. For a given image size and a given processing time, the use of the formula (4) and (5) allows us to verify if the communication time is lower than the processing time.
For instance take an image of size 256x256. The results here under are expressed in milliseconds (T_c : communication time, T_N :calculation time).

N	4	8	16	32	64	128	256
T_c	220	110	50	30	15	9	7
T_N	530	270	130	70	30	20	8

With an image of size 32x32 :

N	4	8	16	32
T_c	6.8	1.9	1.2	1
T_N	8.4	4.1	2.1	1.1

So we can note that the communication time never exceeds the processing time. The maximum numbers of transputers of stage 1, stage 2 and stage 3 of the pipe-line is equal to the minimum size of the image.

4 Experimental results

4.1 Temporal results

For our tests, we use reference images (french CNRS GRECO-GDR134 database).
The network of transputers is composed by 16 T800 20Mhz

image size	t1	t2	t3	t4
32x32	7.1	7.1	8	3.2
64x64	31	31	35	22
96x96	72	72	75	56
128x128	129	129	130	84
256x256	532	532	480	360

ti : task i time

All the temporal values are expressed in milliseconds. t3 and t4 are average temporal values ; the processing times of the associated tasks are data dependent.
The results are output with a rate equal to :

$$\mathbf{rate=max\{t1,t2,t3,t4+t5\}}$$

We will now work on the new Transvision machine. We will use the new generation of transputers *T9000*. The processing time will be divided in the worst case by a factor 10 [8].

4.2 Results

The results are obtained with : $\alpha = 1.5$, *low threshold* = 15.0, *high threshold* = 45.0, *minimum length of a chain* = 5, *polygonal approximation threshold* = 6.

Figure 9 : Sequential result.

Figure 10 : Parallel result.

5 Conclusion

Many real time processors are able to extract the contours of an image (based on DSP or ASIC technology), but a few can realize the full segmentation process [9].
This approach we selected, gives us usefull results for the whole chain of segmentation.
However, this solution seems more suitable for an ASIC implementation, a perspective we will tackle in collaboration with the the Laboratoire GERE, Université de Bourgogne, France [2].

This work is supported by the CNRS PRC AMN project and by CNRS GRECO-GDR134.

Acknowledgment :

The authors would like to thank Nicole BARTHOMEUF for her help.

References

[1] J. Canny : *A computational approach to edge detection.*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No 6, november 1986, pp 679-698.

[2] E. Bourennane, F. Truchetet and M. Paindavoine : *An Improvement of Canny-Deriche Filter for Ramp Edge Detection.*, IMACS Symposium on Signal Processing and Neural Networke, SPANN 93, Montreal may 10-12 1993.

[3] R. Deriche : *Fast algorithms for low level vision.*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.12, No1, january 1990 pp78-86.

[4] J. P. Derutin, B. Besserer, J. Gallice : *A parallel vision machine : TRANSVISION.*, Proc. of Computer Architecture for Machine Perception CAMP9I, Ed B. ZAVIDOVIQUE and P.L. WENDEL december 1991-Paris, pp 241-251.

[5] P. Dew, H. Wang : *Data parallelism and the processor farm model for image processing and syntethis on a transputer array.*, SPIE, vol. 997, Real Time Signal Processing XI, 1988 pp 179-186.

[6] R.O. Duda and P.E. Hart : *Pattern Classification and Scene Analysis.*, New York, ED Wiley, 1973.

[7] P. Legrand and J.P. Dérutin : *Implementation of a contour based image segmentation process on a parallel vision machine.*, Proceedings of ICSPAT, Boston, November 1992.

[8] D. May : *The next generation transputer and beyond.*, Proceeedings of the 2nd Conference EDMCC2, april 1991, pp 7-22.

[9] R. Massen, E. Herre, M. Halschka : *A single-board VMEbus image preprocessor for real-time edge extraction and thinned contour computation*, Proc. of EURASIP 88, Signal Processing IV : Theories and applications, Ed North-Holland, pp 607-610.

[10] S. Nicolle, P. Legrand, J.P Derutin : *Managing the recursive generation of tasks in a transputer network*, Proc. of 2nd Europeean Conference EDMCC2, Springer-Verlag, avril 1991, pp 432-439.