A Morphological Model-Driven Approach to Real-Time Road Boundary Detection for Vision-Based Automotive Systems*

Alberto Broggi and Simona Bertè
Dipartimento di Ingegneria dell'Informazione
Università di Parma
Parma, ITALY, I-43100

Abstract

This work presents a Computer Vision system for road boundary detection in automotive applications. Images are processed by a multiresolution algorithm, driven by a-priori knowledge through a top-down control. In order to face the hard real-time constraints of automotive tasks, a special purpose massively parallel computer architecture, PAPRICA, has been developed. The whole system is currently operative on MOB-LAB mobile laboratory: a land vehicle integrating the results of the activities of the Italian PROMETHEUS units.

The basis of the algorithm is discussed using the formal tools of mathematical morphology, while the choice of the computing architecture and of the computational paradigm is explained.

The generality of the presented approach allows its use also to solve similar problems, namely to detect features exploiting a long-distance correlation, such as the road boundaries in vehicular applications.

1 Introduction

The work presented in this paper has been developed within the Eureka PROMETHEUS project, aimed to the improvement of road traffic safety. The main target of this contribution to the project is the development of a Computer Vision system working in real-time on a moving vehicle: it is well-known, in fact, that the elaboration of images plays a basic role in vehicular applications. The information that must be selected and enhanced by the vision system are the boundaries of the road and/or the lane.

Due to the special field of application, the vision system must be able to process data and produce results in real-time. It is thus necessary to consider data structures, elaboration techniques, and computer architectures able to reduce the response time of the complete system.

*This work was partially supported by CNR Progetto Finalizzato Trasporti under contracts 93.01813.PF74 and 93.04759.ST74. The authors can be reached at the following e-mail address: broggi@CE.UniPR.IT

This work shows how the use of special-purpose hardware can be exploited in order to achieve a high computational power. In fact, a lot of different techniques has been implemented worldwide in the design of Computer Vision based automotive applications (road following), but the integration of a massively parallel architecture on-board has been seldom considered, due to problems related to cost and to the system physical size. The use of a general-purpose massively parallel architecture (as in the case of Carnegie Mellon's NAVLAB I [12, 13], a 16k Mas-Par MP-2) bears high costs not affordable by a large-scale use. This is the reason why generally the execution of low-level computations (efficiently performed by massively parallel systems) has been demanded to general purpose serial processors, as in the case of VaMoRs [8], developed at Universität der Bundeswehr, München. On the contrary, the design and implementation of special-purpose massively parallel architectures (like PAPRICA [5, 9]) keeps the production costs low, while delivers considerably high performances. As a consequence the traditional algorithms and heuristics used when facing real-time problems on sequential architectures must be carefully redesigned according to the specific data-parallel computational model.

The following section presents the complete Computer Vision system as well as the considered approach; section 3 justifies the choice of a multiresolution paradigm for feature extraction; section 4 presents the details of the whole algorithm; section 5 shows some performance figures together with some results; and finally section 6 ends the paper with some concluding remarks and presents the planned future developments.

2 The Computer Vision System

The complete system is composed of a camera and a frame-grabber device, for the acquisition and digitization of image sequences. Data are then pipelined to an on-board computer for processing. The output of the elaboration consists of: *(a)* a set of warnings to the driver displayed on a control-panel activated by a set of leds and/or *(b)* a representation

0-8186-6410-X/94 $4.00

on a on-board monitor (head-up display) of the enhanced features superimposed to the original image, as in figure 1. Since using massively parallel archi-

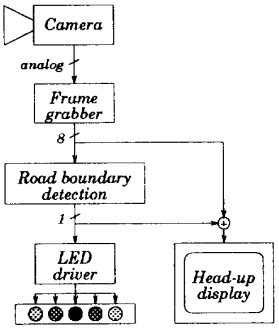


Figure 1: Block diagram of the whole system

tectures it is possible to reach a high computational efficiency for data-parallel algorithms, the vision system has been designed explicitly to exploit low-level processing. This work is in fact an effort toward the use of a *top-down* control (the feature extraction algorithm is based on a *model-driven* approach), instead of the traditional *data-driven* approach, which is generally used for data-parallel algorithms. In a previous work [4], the data-driven approach has been considered, but the approach described in this paper delivers better performances in terms of output quality.

Normally, an image acquired from a moving vehicle is a patch of different areas: a portion of the road region, obstacles, other vehicles, shadows, all surrounded by a specific background depending on the environment. Such a knowledge enables the development of a segmentation algorithm which is simpler and faster than the usual algorithms based on the *recognition* of the represented objects by means of traditional AI techniques. The approach presented in this paper restricts the search area in a neighborhood of standard positions, and uses information on the shape of the road region.

The model which contains the *a-priori knowledge* of the feature to be detected (road boundaries) is encoded in the traditional data structure used in low-level processing, namely in a two-dimensional array. In this specific case, as shown in figure 2, a binary image (hereinafter it will be referred to as *Synthetic Image*) representing two different regions (road and background) has been chosen.

The whole processing is performed by a special-purpose massively parallel SIMD architecture, PAPRICA, developed in collaboration with the Politecnico di Torino, Italy. PAPRICA system (PArallel PRocessor for Image Checking and Analysis) [5, 9] is based on a hierarchical morphology computational model and has been designed as a specialized coprocessor to be attached to a general purpose host workstation: in the current implementation it is connected to a Sparc-based workstation through a VME bus. PAPRICA is made up of 4 major functional parts: the Program Memory (storing up to 256k instructions), the Image Memory (up to 3 MBytes), the Processor Array (PA) and the Control Unit.

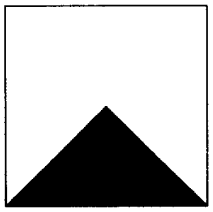


Figure 2: Synthetic image

The first prototype of the PA currently integrated on MOB-LAB is composed of an array of 4×4 full custom ICs, each of them containing a sub-array of 4×4 Processing Elements (PE). The PA is thus a 16×16 square matrix of 1-bit PEs each one with full 8-neighbors connectivity. Each PE has an internal memory composed of 64 bits; a single 16-bit memory fetch from the image memory and a single PA cycle take 350 ns. In the second PAPRICA prototype, which will be completely reengineered, the memory fetch time and the array cycle time will be reduced to 100 and 250 ns respectively, incrementing the performances by a factor from 3 to 4.

The characteristics that enable the integration of this architecture on a generic vehicle are:

- its low production cost,
- its low operative cost, and
- its small physical size.

These characteristics impose a relatively small number of PEs and a simple inter-processor interconnection topology, such as a 2D mesh.

3 Hierarchical Processing

Since the image coming from the camera (*Natural Image*) contains much more details than the synthetic image, the two images cannot be directly compared with local computations. The application of a low-pass filter to the natural image, such as an average filter would meet the requirement of reducing the presence of details, but it would also diminish the relevance of the road boundaries, causing a harder detection. On the other hand, since the road boundaries exploit a long-distance correlation, a subsampling of both the natural and the synthetic image would lead to a comparison less dependent on the detail content. More generally, it is much easier to detect large-sized

objects at a low resolution, where only their main characteristics are present, than at a high resolution, where the details of the specific instance of the represented object can cause its detection to become more difficult. Although the detection of objects works better at a low resolution, the complete recognition and description process can take place only at high resolutions, allowing the identification even of small details, thanks to the preliminary results obtained at a coarse resolution.

These considerations justify the use of a pyramidal data structure [1, 7], consisting of the same image at different resolutions. Furthermore, it is important to note that when the computing architecture contains a number of PEs smaller than the number of image pixels (namely when a PE virtualization mechanism [3] is needed), a useful side effect due to the reduction of the resolution is the decrement in the number of computations to be performed [7].

Thus, the choice of a pyramidal computational paradigm, beside being supported by theoretical reasons, offers an advantage in terms of computational efficiency.

4 The Iterative Process

Each subsampling of the natural image is preceded by a low-pass filter, as shown in figure 3, in order to avoid the *aliasing* [14] phenomenon introduced by the subsampling process. The image is partitioned into

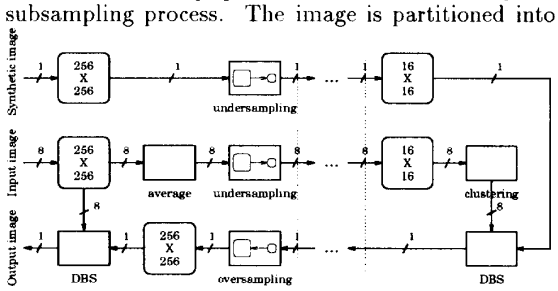


Figure 3: Block diagram of the whole algorithm

non-overlapping square subsets of 4 pixels each; the filter consists of a simple average among the values of the pixels belonging to the same subset. The set of resulting values forms the subsampled image. Once the minimum resolution has been reached (in the version working on MOB-LAB it is 16×16), the edges of the natural image are enhanced by the application of a few iterations of a clustering algorithm [6]. The synthetic image is then reshaped by an iterative algorithm (*Driven Binary Stretching*, DBS, described in detail in the next section) according to the features encoded into the natural subsampled image. The result is then oversampled, and further improved using the same DBS algorithm at a higher resolution. These steps are iterated until the original resolution is reached. Figure 3 presents a fragment of the block diagram of this algorithm, showing at each step the depth (in number of bits/pixel) of every image.

4.1 Driven Binary Stretching

The goal of the DBS algorithm, sketched in figure 4, is to reshape the input binary synthetic image according to the grey-tone natural image, which is used as a guide. The reshaped synthetic image is then oversampled and used at a higher resolution.

The features of interest in the natural grey-tone image are the road boundaries: usually the boundary of a generic object is represented by brightness discontinuities in the image reproducing it, thus the first step is accomplished by the application of a gradient-based filter to the input image. Then, as shown in figure 4, a threshold is applied to the gradient image, in order to keep only the most significant edges.

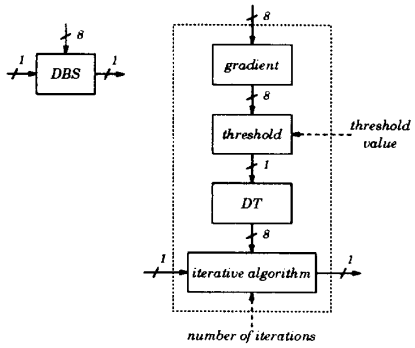


Figure 4: Block diagram of the DBS algorithm

Since a 2D mesh-connected massively parallel architecture is used, in order to stretch the synthetic image toward the positions encoded in the thresholded image, an iterative algorithm must be performed. Each border pixel of the synthetic image should move toward the position of the nearest foreground pixel of the thresholded image. To this purpose, a scalar field V is defined on E^2. The field V

$$V : E^2 \rightarrow E \ , \tag{1}$$

links the position of each pixel $p \in E^2$ to a scalar value $V(p) \in E$, which represents the *potential* associated to the pixel itself. This value is encoded in a *potential* image. The difference $V(p) - V(q)$ represents the *cost* of moving pixel p toward position q. The iterative process enables all the pixel movements associated to a negative cost. The scalar field should be such that a negative cost corresponds to the movement toward the nearest position of the foreground pixels t_i of the thresholded image.

Thus, the value $V(p)$ should depend on the minimum distance between pixel p and pixels t_i:

$$V(p) \triangleq -\min_i d(p, t_i) \ , \qquad (2)$$

where $d : E^2 \times E^2 \rightarrow E$ represents the distance between two pixels, measured with respect to a given metric. In this case, due to the special simplicity of the implementation, a *city block* (*Manhattan distance*) metric has been chosen (see figure 5). In fact,

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

Figure 5: *City block* or *Manhattan* distance

a very efficient method for the computation of the potential image using 2D mesh-connected architectures is based on the iterative application of morphological dilations [15, 10]:

1. a scalar counter is initialized to 0; for parallel architectures which can not run any fragment of scalar code, this counter is associated to every pixel in the image;
2. the counter is decremented;
3. the binary input image is dilated using a 4-connected structuring element N, formed by the following elements:

$$N = \Big\{(0,1);(0,-1);(0,0);(1,0);(-1,0)\Big\} \ ; \quad (3)$$

4. when a pixel, due to the previous dilation, changes its state from *background* to *foreground*, the value of the counter is assigned to the potential of that position;
5. the process is then repeated from step 2, until the output of the morphological dilation is equal to its input.

Thus, the potential image can be generated by the application of a sort of *Distance Transform*, DT [2] to the binary thresholded image. Due to a more efficient implementation-dependent data handling, the final version of the potential image DT is obtained adding a constant λ to every coefficient, in order to work with only positive values: λ represents the maximum value allowed for grey-tone images[1]. Thus the new definition of the scalar field V is:

$$V(p) \triangleq \lambda - \min_i d(p, t_i) \ , \qquad (4)$$

[1] In this specific case, since 8-bit images are considered, $\lambda = 255$.

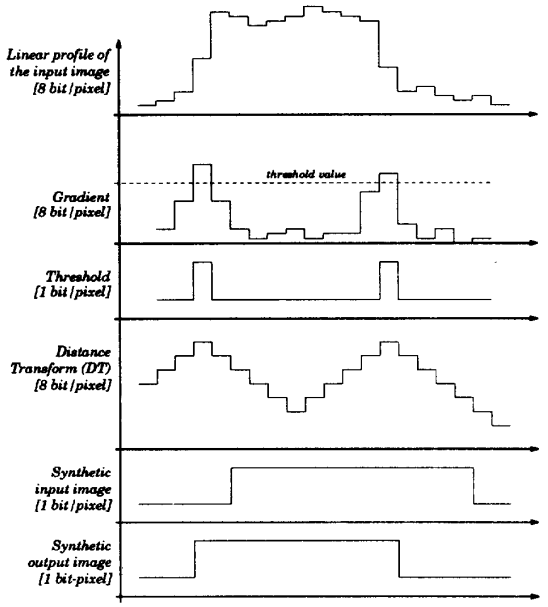


Figure 6: Example of 4 iterations of the DBS algorithm (monodimensional case)

Furthermore, since the 'distance' information will be used only for the pixels belonging to the border of the synthetic image and their neighbors, the iterative process may be stopped when the DT has been computed for such pixels, producing a performance improvement.

As already mentioned, the heart of the algorithm is constituted by an iterative process, whose goal is to move the edge pixels of the synthetic image in the directions in which the DT gradient has a maximum. Figure 6 shows an example of a monodimensional stretching.

4.1.1 The DBS morphological implementation

In order to describe the DBS algorithm, it is important to recall some concepts of Mathematical Morphology [15, 10]. A two-dimensional binary image S is represented as a sub-set of E^2, whose elements correspond to the *foreground* pixels of the image:

$$S = \Big\{s \in E^2 \mid s = (x, y), \ x, y \in E\Big\} \ , \qquad (5)$$

where vector (x, y) represents the coordinates of the generic element s.

The elements generated by the dilation of S by the 4-connected structuring element N shown in equation (3) are defined to be the *external edge* of S:

$$\mathcal{B}_e(S) = (S \oplus N) \cap \overline{S} \ . \qquad (6)$$

In a similar way, the elements removed from set S by its erosion using the same structuring element N are defined to be the *internal edge* of S:

$$\mathcal{B}_i(S) = \overline{(S \ominus N)} \cap S \ . \tag{7}$$

The logical union of $\mathcal{B}_e(S)$ and $\mathcal{B}_i(S)$ is the *edge* of S:

$$\mathcal{B}(S) = \mathcal{B}_e(S) \cup \mathcal{B}_i(S) \ . \tag{8}$$

The elements of $\mathcal{B}(S)$ are the only elements which can change their state (from *foreground* to *background* and viceversa) by the application of a single iteration of the DBS algorithm. More precisely, two different rules are applied to the two edges: the first, applied to the external edge, determines the elements to be included in set S; while the second, applied to the internal edge, determines the elements that must be removed from set S. The number of iterations required for a successful stretching is anyway low: at a coarse resolution few iterations are needed for a complete reshaping of the synthetic image, since the image size is small; and again, after the initial low-resolution stretching, only a few iterations are sufficient for a high-resolution refinement, since the shape encoded into the synthetic image already approximates the result.

1. **Rule for the external edge:**
   - each pixel of the external edge of S computes the minimum value of the DT associated to its 4-connected neighbors which belong to set S;
   - all the pixels, whose associated DT is greater than the value previously computed, will be inserted in set S.
2. **Rule for the internal edge:**
   - each pixel of the internal edge of S computes the maximum value of the DT associated to its 4-connected neighbors which does not belong to set S;
   - all the pixels, whose associated DT is greater than the value previously computed, will be removed from set S.

Note that rule 2 is the dual of rule 1: the latter tends to stretch the foreground onto the background, while the former acts in the opposite way, using the complement of the synthetic image.

4.1.2 Processing the external edge

In the following, the rule for the external edge will be considered, assuming step n in the iterative process. Recalling the Mathematical Morphology notations used to identify a grey-tone two-dimensional image, the DT image is a sub-set of E^3:

$$DT = \left\{ d \in E^3 \mid d = (u, v),\ v = V(u),\ \forall\, u \in E^2 \right\} , \tag{9}$$

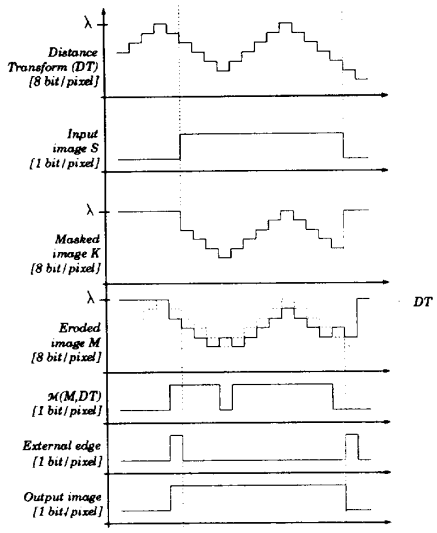


Figure 7: External edge monodimensional stretching

where u represents the position of element d in E^2, and v represents its *value*.

The *pixelwise masking* operation between a binary and a grey-tone image is here defined as a function

$$\odot : E^2 \times E^3 \rightarrow E^3 \tag{10}$$

such that $A \odot B$ represents a subset of B containing only the elements $b = (u, v)$ whose position vector $u \in E^2$ belongs also to A:

$$A \odot B \triangleq \left\{ x \in E^3 \mid x = (u, v) \in B,\ u \in A \right\} \tag{11}$$

In order to compute the minimum value of the DT in the specified neighborhood, let us consider image $K_e^{(n)}$

$$K_e^{(n)} = \left(S_e^{(n)} \odot DT \right) \cup \left(\overline{S_e^{(n)}} \odot L \right) , \tag{12}$$

where $S_e^{(n)}$ represents the binary image at step n, the subscript e indicates that the rule for the external edge is being considered, and finally

$$L = \left\{ l \in E^3 \mid l = (u, \lambda),\ \forall\, u \in E^2 \right\} . \tag{13}$$

As shown in [10], in order to compute the minimum value of a grey-tone image $K_e^{(n)}$ in a 4-connected neighborhood, the following grey-scale morphological erosion should be used:

$$M_e^{(n)} = K_e^{(n)} \ominus Q \ , \tag{14}$$

where

$$Q = \left\{ (1,0,0);\ (-1,0,0);\ (0,1,0);\ (0,-1,0) \right\} . \tag{15}$$

In order to determine the set of elements in which $M_e^{(n)}$ has a value smaller than DT, a new function $\mathcal{M}$ is required:

$$\mathcal{M} : E^3 \times E^3 \rightarrow E^2 \ . \tag{16}$$

Such a function is defined as

$$\mathcal{M}(A,B) \triangleq \left\{ x \in E^2 \mid \mathcal{V}(A,x) < \mathcal{V}(B,x) \right\} , \tag{17}$$

where $\mathcal{V} : E^3 \times E^2 \rightarrow E$ is defined as

$$\mathcal{V}(A,x) \triangleq \begin{cases} a & \text{if } \exists\, a \in E \mid (x,a) \in T(A) \\ -\infty & \text{otherwise} \end{cases} \tag{18}$$

In equation (18), $T(A)$ represents the *top* of A [10], here defined as

$$T(A) \triangleq \Big\{ t \in A,\ t = (u,v) \mid \nexists\, t' = (x,v') \in A \quad \text{for which } v' > v \Big\} . \tag{19}$$

The set of elements which will be included in set $S_e^{(n+1)}$ is given by the logical intersection between $\mathcal{M}(M_e^{(n)}, DT)$ and the set of elements belonging to the external edge of $S_e^{(n)}$:

$$E_e^{(n)} = \mathcal{M}\Big(M_e^{(n)}, DT\Big) \cap \mathcal{B}_e\Big(S_e^{(n)}\Big) \ . \tag{20}$$

Thus, the final result of iteration n is given by

$$S_e^{(n+1)} = S_e^{(n)} \cup E_e^{(n)} \ . \tag{21}$$

Figure 7 shows the execution of a single iteration of rule 1 on a monodimensional profile of an image.

4.1.3 Processing the internal edge

Following similar steps, let us formalize rule 2. In order to compute the maximum value of the DT in the specified neighborhood, let us consider image $K_i^{(n)}$

$$K_i^{(n)} = \overline{S_i^{(n)}} \odot DT \ , \tag{22}$$

where the subscript i indicates that the rule for the internal edge is being considered. As shown before, in order to compute the maximum value of a grey-tone image $K_i^{(n)}$ in a 4-connected neighborhood, the following morphological dilation should be used:

$$M_i^{(n)} = K_i^{(n)} \oplus Q \ , \tag{23}$$

where Q is defined in equation (15).

The set of elements which will be removed from set $S_i^{(n+1)}$ is given by the logical intersection between

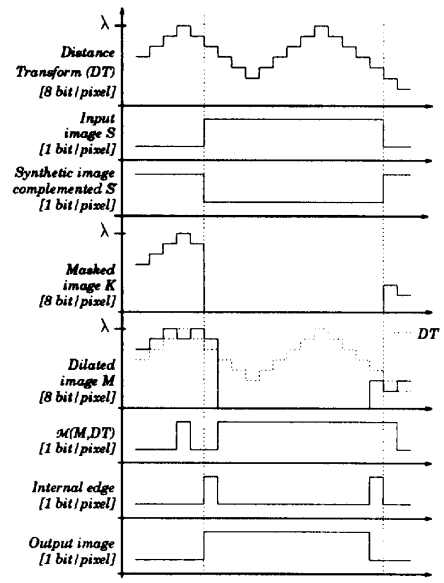


Figure 8: Internal edge monodimensional stretching

$\mathcal{M}(M_i^{(n)}, DT)$ and the set of elements belonging to the internal edge of $S_i^{(n)}$:

$$E_i^{(n)} = \mathcal{M}\Big(M_i^{(n)}, DT\Big) \cap \mathcal{B}_i\Big(S_i^{(n)}\Big) \ . \tag{24}$$

Thus, the final result of iteration n is given by

$$S_i^{(n+1)} = \overline{\Big(S_i^{(n)} \cup E_i^{(n)}\Big)} = S_i^{(n)} \cap \overline{E_i^{(n)}} \ . \tag{25}$$

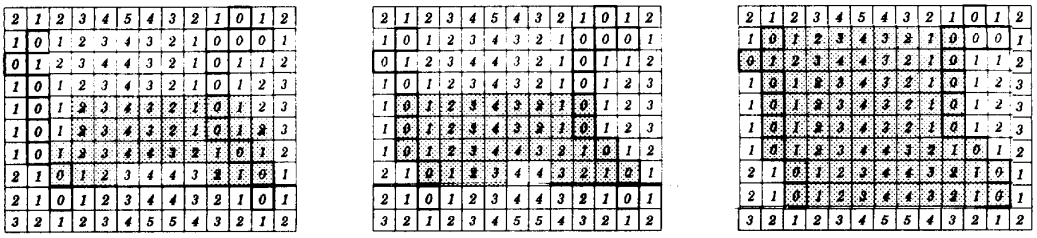
Figure 8 shows the execution of a single iteration of rule 2 on a monodimensional profile of an image.

4.1.4 Flat handling

Figures 7 and 8 refer to a monodimensional stretching; unfortunately, when dealing with 2D data structures, the DT does not present a strictly increasing or decreasing behavior, even locally. Thus, an extension to the previous rules must be considered for a correct flat-handling.

Figure 9.a shows the modulus of the DT coefficients (in the case of $\lambda = 0$), together with the input binary image (indicated with a grey texture). The output of the iterative DBS filter is presented in figure 9.b.

Since the movement of a generic pixel toward positions holding an equal DT coefficient is expressly disabled, the resulting binary image does not completely follow the shape encoded in the DT image. A minor revision to the definition of rule 1 is thus needed. Figure 9.c is obtained with the following rule applied to the external edge $\mathcal{B}_e(S)$.



a) Input binary image *b) Stretching without flat-handling* *c) Stretching with flat-handling*

Figure 9: Two-dimensional stretching: different line markings represent different states of input binary image; grey areas represent the result

1. **New rule for the external edge:**
   - each pixel of the external edge of S computes the minimum value of the DT associated to its 4-connected neighbors which belong to set S;
   - all the pixels, whose associated DT is greater than the value previously computed, and all the pixels not belonging to the thresholded image, whose associated DT is equal to the value previously computed will be inserted in set S.

The specific requirement for the pixels which are moving toward a flat region, not to belong to the thresholded image, ensures that the binary image does not follow the chain of maxima of the DT. With such a requirement, in the specific case of figure 9 the maxima in the upper-right hand side are not included in the resulting binary image.

5 System Performance Analysis

The complete processing presented in this work has been first implemented on a 4k processors Connection Machine CM-2 [11], and then on the target architecture PAPRICA. The goal of the CM-2 implementation was the verification of the correctness of the algorithm. Nevertheless, it presented highly encouraging results, and the code was ported to PAPRICA system.

Moreover, since a 256×256 image presents a sufficient detail content for this specific application, the processing can begin with a 256×256 image. Since the current output device integrated on MOB-LAB control-panel consists of a set of only 5 leds, this high quantization allows to stop the pyramidal processing at a medium resolution (e.g. 64×64), furthermore decreasing the processing time.

The performances obtained through a direct Assembly implementation on PAPRICA architecture are shown in table 1, where the timings are obtained averaging the processing of different frames of a sequence. The complete image acquisition, processing, and output stages require about 6, 510, and 2 ms respectively. The current system is capable of processing about 2 frames per second.

Operation	Image size	Time
Resolution Reduction	$256^2 \rightarrow 16^2$	448 ms
DT, DBS, Oversampl.	$16^2 \rightarrow 32^2$	11 ms
DT, DBS, Oversampl.	$32^2 \rightarrow 64^2$	51 ms
TOTAL	$256^2 \rightarrow 16^2 \rightarrow 64^2$	510 ms

Table 1: Performance of PAPRICA system

Figure 11 presents the result obtained by the processing of the image shown in figure 10.

The generality of the approach enables the detection of other sufficiently large-sized features: for example, using a different synthetic image it is possible to detect the boundaries of the lane, instead of the road boundaries.

6 Conclusions

In this paper a novel approach aimed to the detection of the road boundaries for automotive applications has been considered. The multiresolution approach together with a top-down control allow to achieve remarkable performances in terms of both computational time (when mapped on massively parallel architectures) and output quality. Moreover, the tuning of the algorithm is extremely simple, since the only input parameter is a threshold value, whose automatic determination is under testing.

The presented algorithm is currently being extended to handle image sequences. In this case two main advantages can be achieved by the substitution of the synthetic image with the binary image obtained as output in the processing of the previous frame.

- Due to the high correlation between two successive frames, the number of subsamplings can be reduced, since the expected result should be similar to the input binary image. Thus, the shorter the time interval between two successive frames

Figure 10: Original image

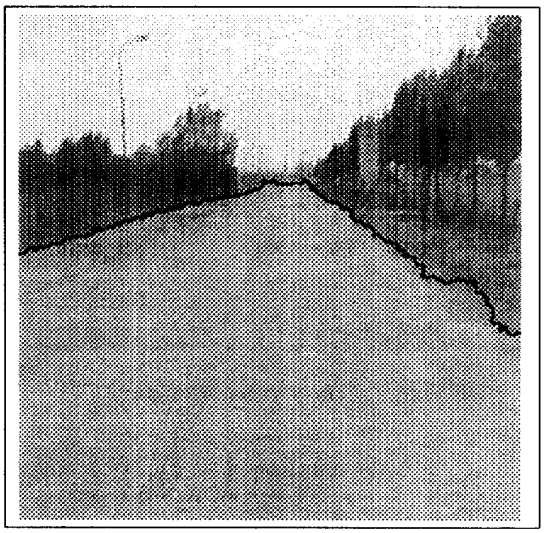


Figure 11: Output image

of the sequence, the more the number of subsamplings can be reduced, and thus the shorter the computational time.

- The number of iterations of the DBS filter can be reduced for the same reasons explained above.

These considerations lead to an extension of the algorithm (for the elaboration of image sequences), whose performances should be better than the ones obtained by the independent processing of each frame of the sequence. The proposed algorithm extension is currently under final testing and tuning and will be integrated on the second version of PAPRICA architecture. Software simulations have shown that the use of the second version of PAPRICA system running the algorithm extension mentioned above should give a performance improvement of more than one order of magnitude. In these conditions, the processing of image sequences will be performed nearly at video rate.

Acknowledgments

The authors would gratefully thank Andrea Folli and Stefano Mora for their help in the development of the whole project.

References

[1] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice Hall, 1982.

[2] G. Borgefors. Distance Transformations in Digital Images. In *CVGIP*, vol.34, pages 344–371, 1986.

[3] A. Broggi. Performance Optimization on Low-Cost Cellular Array Processors. In *Proc. IEEE - MPCS*, Ischia, Italy, 1994. IEEE CS. In press.

[4] A. Broggi. Parallel and Local Feature Extraction: a Real-Time Approach to Road Boundary Detection. *IEEE Trans. on Image Processing*, Feb. 1995. In press.

[5] A. Broggi, G. Conte, F. Gregoretti, C. Sansoè, and L. M. Reyneri. The PAPRICA Massively Parallel Processor. In *Proc. IEEE - MPCS*, Ischia, Italy, May 2-6 1994. IEEE CS. In press.

[6] A. Broggi and A. Gandini. Parallel Image Clustering: A Real-Time Application for a Special-Purpose Architecture. In *Proc. 8th SCIA*, volume 1, pages 297–304, Tromsoe, Norway, May 25-28 1993.

[7] T. Fountain. *Processor Arrays: Architectures and applications.* Academic-Press, London, 1987.

[8] V. Graefe and K.-D. Kuhnert. Vision-based Autonomous Road Vehicles. In *Vision-based Vehicle Guidance*, pages 1–29. Springer Verlag, 1991.

[9] F. Gregoretti, L. M. Reyneri, C. Sansoè, A. Broggi, and G. Conte. The PAPRICA SIMD array: critical reviews and perspectives. In *Proc. IEEE - ASAP'93*, pages 309–320, Venezia, Italy, Oct. 1993. IEEE CS.

[10] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Trans. on PAMI*, 9(4):532–550, 1987.

[11] W. D. Hillis. *The Connection Machine.* MIT Press, Cambridge, Ma., 1985.

[12] T. M. Jochem and S. Baluja. A Massively Parallel Road Follower. In *Proc. CAMP'93*, pages 2–12, New Orleans, December 15-17 1993. IEEE CS.

[13] T. M. Jochem and S. Baluja. Massively Parallel, Adaptive, Color Image Processing for Autonomous Road Following. In *Massively Parallel Artificial Intelligence.* AIII Publishers and MIT Press, 1993.

[14] W. K. Pratt. *Digital Image Processing.* John Wiley & Sons, 1978.

[15] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, London, 1982.