# MORPHOLOGICAL SEGMENTATION ON THE PROGRAMMABLE RETINA: TOWARDS MIXED SYNCHRONOUS/ASYNCHRONOUS ALGORITHMS

ANTOINE MANZANERA*
*Ecole Nat. Sup. de Techniques Avancées*
*32, Bd Victor - 75015 Paris - FRANCE*
*Antoine.Manzanera@ensta.fr*
*Tel: 33 1 45 52 44 42*

**Abstract.** We present a new algorithm to compute the watershed transform on a massively parallel machine. Logic minimization, together with a novel approach to the notion of dynamics, allow the implementation of the watershed algorithm on a cellular automata grid, such as the programmable retina. We then investigate the profit acquired from computing some parts of the algorithm in an asynchronous way, in terms of time and energy consumption.

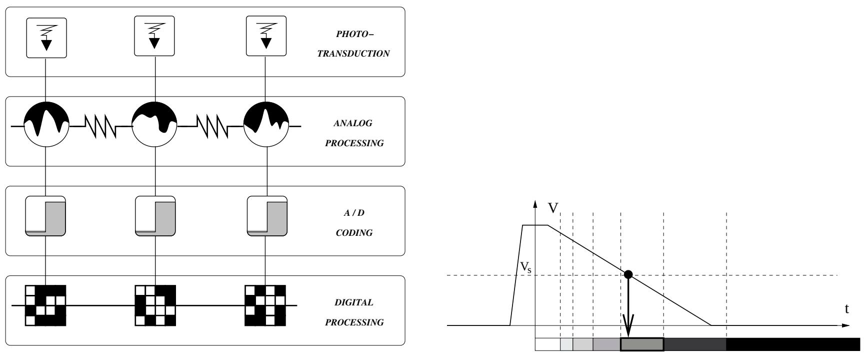
**Key words:** watershed, programmable retina, dynamics, Buffon's method, synchronous/asynchronous parallelism, SIMD, FIFO

## 1. Introduction

Implementation of morphological algorithms on massively parallel machines is not a challenging issue in itself, since the translation invariance and local knowledge had already been set as principles of the morphological operators by its creators [10]. But on the one hand, the deceiving success of large size array SIMD machines, due to conception difficulties, cost of the I/O flow, and lack of efficiency of certain operations, and on the other hand, the great production of the algorithmicists [9] to optimize the implementation on sequential architectures, have pushed the affinities between mathematical morphology and SIMD into the background.

Today, the constant improvement of CMOS technologies allows to introduce a rough intelligence within every pixel of an electronic camera, thus leading to the integration of more and more powerful programmable retinas (Section 2). But unfortunately (resp. fortunately), the algorithmic problems remain difficult (resp. interesting), as we wish to illustrate in this paper through the example of the watershed (Section 3). First, because of the harsh constraints due to the limited capability of the processors, that force to a constant effort of logic minimization (Section 4). And second, because of the inherent limitation of the SIMD programming, that incite to investigate the possibilities offered by architectural sophistications of the SIMD mesh (Section 5).

* This work has been partially realized at CTA/GIP and INT/ARTEMIS, with the financial support of EADS.




(1) (2)

Fig. 1. (1) The four functional layers of the programmable retina. (2) The NSIP digitizing process: time-amplitude coding for level-sets processing and anamorphoses computation.

## 2. The programmable retina

An artificial retina is a circuit combining image acquisition and processing functions. The interest of such device is to reduce dramatically the flow of information within a vision system, by performing the image processing computations in the focal plane. The programmable retina [5], [2], [8] is a CMOS sensor and a cellular SIMD machine at the same time, with a very small digital processor integrated into every pixel. Such a circuit is adapted to real-time constrained image processing system, thanks to its high degree of parallelism, as well as embarkable or autonomous systems, thanks to its performances in terms of compactness and low-power.

Figure 1(1) shows (in 1D) the organization of the acquisition and processing functions in the programmable retina. The analog processing layer is considered here as the simplest possible and dedicated to very early image processing, such as spatio-temporal filtering. In fact, we will suppose that the signal processed by the digital part can be the luminance image itself, but also a temporal differenciation, or a spatial gradient intensity, both functions being computable with small analog circuitry [7].

Our algorithmic interest actually begins with the third layer: analog/digital conversion (ADC). Through the NSIP (near sensor image processing) principle [4], a very compact ADC can be performed under the form of a capacity which unloading time is inversely proportional to the value to measure. The analog value obtained from phototransduction and analog processing can then be digitized through a multiple thresholding (Figure 1(2)) that will produce a set of binary images. From a morphological point of view, these images represent the level sets of the signal to process, that can be coded in the digital memory, or if possible, processed during the ADC. Furthermore, modifying the number and position of the thresholds along the time axis correspond to the application of

**procedure** $dilate_4(p)$ {
Boolean plane $p_1$;
$p_1 = p^N \vee p^E$;
$p = p \vee p_1 \vee p_1^{SW}$;
}

**procedure** $dilate_8(p)$ {
Boolean plane $p_1$;
$p_1 = p \vee p^N \vee p^S$;
$p = p_1 \vee p_1^E \vee p_1^W$;
}

| | | |
|---|---|---|
| $p^{NW}$ | $p^N$ | $p^{NE}$ |
| $p^W$ | $p$ | $p^E$ |
| $p^{SW}$ | $p^S$ | $p^{SE}$ |

TABLE I
Retinal code of the basic operators in 4- and 8-connectivity. On the right, the notations used for the translations. The erosions are obtained by turning the $\vee$ (OR) into $\wedge$ (AND).

a morphological anamorphose.

The digital layer, which is the main matter in this paper, is made of a mesh of digital processors. It is an SIMD machine, with an important originality related to a limited computation capability, due to a tiny memory (only a few bits per pixel). The data we operate on are a few binary images, called Boolean planes (which can represent the grey levels of the same image). The only hard-wired operations on these data are: (1) One-pixel translation of one image. (2) Boolean operation between two images (OR, AND, XOR, ...).

With its Boolean language, the programmable retina is naturally fitted for the non linear processing. Table I shows the computation of the elementary morphological operations, which are computed in a few elementary operations and use 2 bits of memory: 1 bit of data, and 1 bit for the computation.

More generally, the time complexity of a retinal operator is the number of elementary operations that compose it, and the space complexity the number of bits needed to perform it. In our very constrained context of memory, the space complexity is a critical factor for the implementation of algorithms. The consequence is a constant effort to reduce the amount of data memory coded in the digital processors, in order to maximize the computation power.

The computation of the watershed [3] is a significant example of this effort. Indeed, the watershed produces a binary image, which can intrinsically represent a wide-dynamics signal, and thanks to time-to-amplitude coding, it can be processed "on the fly" on the incident signal of the digital layer.

## 3. Computation of the watershed transform

From now on, we will suppose that the computation is made on a digital bidimensional signal $I$ of range value $[0, n]$, known by its $n$ binary thresholds, or level sets, $\{I_i\}_{i \in [1,n]}$, with $I_i = \{x \in \mathbb{Z}^2; I(x) < i\}$, which are either read on the fly from the ADC, or produced from an inner digital coding, thus occupying $\log_2(n+1)$ bits of memory.

In the retinal algorithms presented below, the operations are performed within a fully parallel framework over the pixels. The instructions are either elementary operations, or procedures presented earlier. The operators used are $\wedge$, $\vee$ and $\neg$, respectively AND, OR and NOT. The space complexity of each

```
procedure reconstruction(p, p') {
Repeat until stability, {
        dilation(p);
        p = p ∧ p';
        }
    }
```

```
procedure opening_by_reconstruction_r(p) {
Boolean plane p1;
p1 = p;
Repeat r times {
        erosion(p);
        }
reconstruction(p, p1);
    }
```

TABLE II
Retinal code of the geodesic reconstruction (left), and opening by reconstruction (right).

```
procedure homotopic_dilation_EAST(p) {
Boolean plane p1, p2;
    p1 = ¬p ∧ p^E;
    p2 = p1 ∨ p1^N ∨ p1^S;
    p1 = ¬p2 ∧ p^W;
    p = p ∨ p1;
    }
```

```
procedure geodesic_SKIZ(p, p') {
Repeat until stability,
and for X in {EAST, SOUTH, WEST, NORTH}
(periodically repeated) {
        homotopic_dilation_X(p);
        p = p ∧ p';
        }
    }
```

TABLE III
Retinal code of the homotopic dilation (left, the other directions are deduced by rotation), and of the geodesic SKIZ (right).

procedure will be measured, as the sum of the data bits (the parameters) and the computation bits (the "local" variables of the procedure, including the hidden variables used by sub-procedures).

The procedures in this section can be presented briefly, as it is a classical implementation of the watershed by immersion simulation. It is based on two binary functions: the geodesic reconstruction and the geodesic SKIZ.

The geodesic reconstruction is computed by the left procedure of Table II. It uses 3 bits of memory. The right procedure of Table II shows a direct application in morphological filtering: the opening by reconstruction by a ball of radius $r$. This procedure, that needs 3 bits of memory too, will be used in the next section.

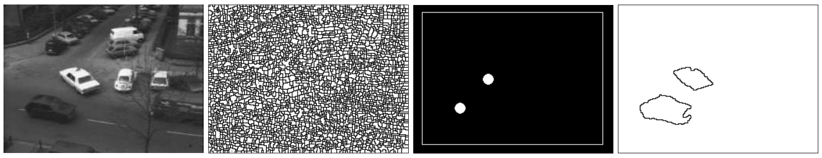
The classical approximation of the SKIZ by the dual homotopic kernel is computed thanks to the relaxation of the homotopic dilation operator, shown on Table III (left), that uses 3 bits of memory. The geodesic SKIZ is simply obtained by computing a logical AND with the image of reference, so it uses 4 bits of memory.

Finally, the watershed of the digital signal $I$ of range value $[0, n]$ is computed thanks to the procedure of Table IV (left), where $p$ is the binary output. This procedure uses 4 bits of memory. To overcome the well-known problem of over-segmentation (Figure 2 (2)), a classical solution consists in using an image of markers (Figure 2 (3)) to constrain the topology of the final watershed (Figure 2 (4)). The constrained procedure is shown on Table IV (right), where $m$ is the image of markers. Its space complexity is: 5 bits of memory.

```
procedure watershed(I, p) {
Boolean plane p1, p2;
p = I[1];
For k = 2 to n {
        p1 = I[i];
        p2 = p;
        reconstruction(p2, p1);
        p2 = ¬p2 ∧ p1;
        p = p ∨ p2;
        geodesic_SKIZ(p, p1);
        }
}
```

```
procedure watershed_bis(I, m, p) {
Boolean plane p1;
p = m;
For k = 1 to n {
        p1 = I[i];
        geodesic_SKIZ(p, p1);
        p = p ∨ m;
        }
}
```

TABLE IV
Retinal code of the watershed transform (left), and of the watershed with markers (right).



(1) (2) (3) (4)

Fig. 2. Results of the watershed transform. The computed signal in this example is the gradient intensity of Image (1). The raw watershed transform is shown on image (2). The markers are the (white) connected components of image (3): one disk marks the white taxi at the center, another disk marks the dark car on the left, the rectangle marks the background. The watershed computed with these markers is shown on Image (4) (a post-process is used to eliminate the curve between the two catchment basins).

It can be seen that the computation of the watershed lends itself naturally to a compact implementation, well adapted to our architecture. Nevertheless, the markers need to be computed before, and in most cases of segmentation, they are not easy to find. Another way to avoid over-segmentation is to discard non-significant minima. This is done thanks to filtering and dynamics setting.

## 4. Filtering and dynamics setting

Indeed, non-significant minima can appear for two reasons: (1) noise on the signal, and (2) poor-dynamics catchment basins. The noise problem is addressed through morphological filtering (see Figure 3): let $\rho$ be the minimal thickness of a basin admissible, the filtering consists in performing an opening by reconstruction by a ball of size $\rho$ on every level set of $I$. This can be done without extra-memory.

The dynamics problem is more critical. The classical solution [6] to remove minima of dynamics inferior to $h$ (basins of depth inferior to $h$) consists in reconstructing $I$ over $I + h$. This can be done in the retina by performing the

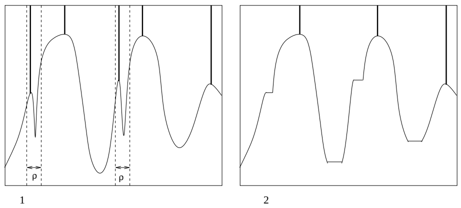


Fig. 3. The first cause of over-segmentation in the watershed: noise on the signal. Basins of thickness inferior to $\rho$ (1) are suppressed thanks to morphological filtering (2).

binary reconstruction of $\neg I_i$ in $\neg I_{i-h}$, for every level set such that $i > h$. But this method implies that two level sets distant of $h$ must be available in the memory at the same time. If the signal is not already coded in the digital memory, this costs $h - 1$ bits of memory. The solution proposed in this paper addresses differently the dynamics issue, and uses only 1 extra bit of memory.

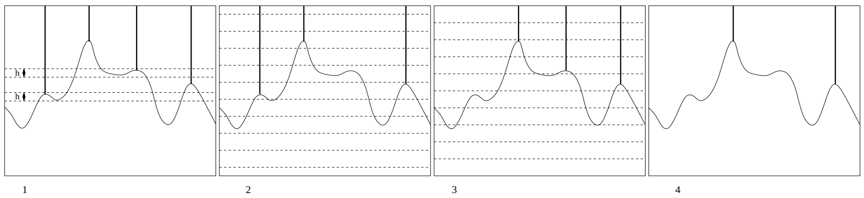


Fig. 4. The second cause of over-segmentation in the watershed: poor dynamics minima. Basins of depth inferior to $h$ (1) are discarded thanks to the Buffon's method (2), (3), (4).

The principle of our solution (see Figure 4) is to compute two watersheds of the same signal. The level sets of the signal are sub-sampled with a regular step of $h$, and two consecutive level sets are used for computing two distinct watersheds. So the two watersheds are computed over a sub-sample of the level sets, with a regular step of $2h$, in such a way that the two sub-samples are in phase quadrature (see Figure 4 (2) and (3)). We then postulate that the only significant curves of the segmentation are those that appear in both watersheds, and then the results is computed by performing a logical "AND" between the two watersheds(see Figure 4 (4)) .

This "Buffon's" method is indeed based on the monodimensional version of the needle experience of the french naturalist. Throwing randomly a segment of length $l$ (the needle) over a line marked by samples regularly spaced of $h$, what is the probability that the segment hits two samples ? It is easy to see that it is the continuous piece-wise affine function of $l$ that is equal to 0 if $l < h$, and 1 if $l > 2h$. For the watershed computed by the procedure of Table V, it

```
procedure watershed_ter(ρ,h,σ)(I, p) {
Boolean plane p1, p2, p3;
p = I[h];
opening_by_reconstruction_ρ(p);
p1 = I[2h];
opening_by_reconstruction_ρ(p1);
for i = 3 to ⌈n/h⌉ {
        p2 = I[i × h];
        opening_by_reconstruction_ρ(p2);
        if (i%2 == 0) {
                p3 = p;
                reconstruction(p3, p2);
                p3 = ¬p3 ∧ p2;
                p = p ∨ p3;
                geodesic_SKIZ(p, p2);
        } else {
                p3 = p1;
                reconstruction(p3, p2);
                p3 = ¬p3 ∧ p2;
                p1 = p1 ∨ p3;
                geodesic_SKIZ(p1, p2);
                }
        }
for i = 1 to σ {
        erode4(p);
        }
p = p ∨ p1;
SKIZ(p);
}
```

TABLE V
Retinal code of the watershed with parameters $(\rho, h, \sigma)$.

means that:

- No minimum of dynamics $\delta < h$ can appear in the watershed.
- All minima of dynamics $\delta > 2h$ will appear in the watershed.
- A mimimum of dynamics $h < \delta < 2h$ will appear in the watershed with a probability $\frac{\delta - h}{h}$.

In fact, the actual combination of the two watersheds is a bit more difficult than a simple AND. Indeed, the quantization of the level sets discriminates minima of poor dynamics, but also induces a shift on the frontiers of the other minima, i.e. their positions can vary in the two watersheds. If $\sigma$ is the maximal shift admitted, $W_1$ and $W_2$ the two "semi-watersheds", the intersection is computed as: $W_1 \wedge (W_2 \oplus B_\sigma)$, where $\oplus$ is the dilation and $B_\sigma$ a digital ball of radius $\sigma$. The value of the shift depends on (1) the quantization step $h$, (2) the local sharpness of the peak in the signal. So the parameter $\sigma$ will be choosen according to the minimal sharpness of a contour desired, and proportionnally to $h$. Furthermore, the right combination is not the common points, but the common closed curves of the two watersheds, so the final watershed is $W = HK(W_1 \wedge (W_2 \oplus B_\sigma))$, where $HK$ is the homotopic kernel. See end of procedure of Table V: as we are working on complementary watersheds, we compute in fact $W = SKIZ(W_1 \vee (W_2 \ominus B_\sigma))$.

A result of computation is shown on Figure 5. The important achievement of this method is that computing the watershed with filtering and dynamics setting can be performed at a constant memory cost, independant of the parameters: 5 bits.

## 5. Benefits of asynchronism

We have shown that the watershed transform, with its refinements used in morphological segmentation, can be implemented at a very low material cost (AND, OR, NOT, translation, 5 bits of memory), on a cellular automata grid. This means that, today, it can be computed within the pixels of a digital camera, at the output of the ADC. But what about the efficiency of this computation ?

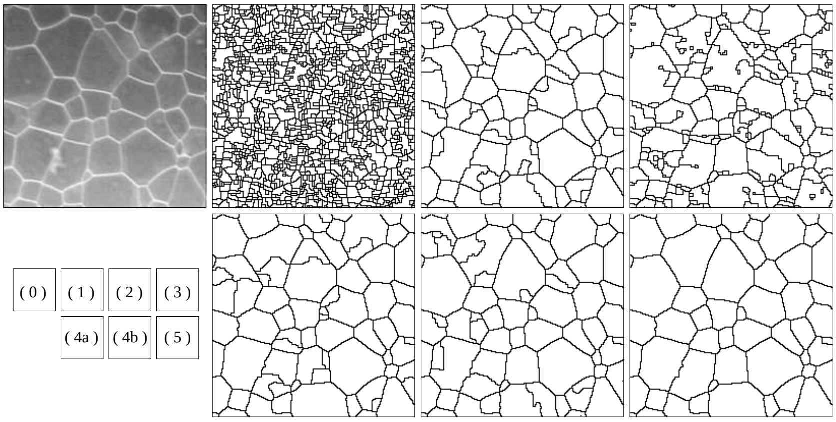



Fig. 5. Results of the watershed with parameters. (0) Original image (200 x 200 x 8 bits). (1) Raw watershed ($\rho = 0, h = 0$). (2) Filtering with no dynamics setting ($\rho = 2, h = 0$). (3) Dynamics setting with no filtering ($\rho = 0, h = 8$). (4a) and (4b) Dynamics setting by Buffon's method: two filtered watersheds ($\rho = 2$) are computed by sub-sampling the greylevels with a step $\Delta = 16$, such that the two sub-samples are in phase-quadrature. (5) The result is obtained by combination of the two watersheds ($\rho = 2, h = 8, \sigma = 2$).

Indeed, by performing the segmentation directly in the sensor, we aim at saving time and energy, thanks to the in-site massively parallel computation. However, if we examine the efficiency of the SIMD parallelism used in the previous procedure, it turns out that the computation of relaxation operators like the geodesic reconstruction can be very inefficient. In the typical case of the reconstruction of a curve from one of its points, only a few pixels change their value at every iteration, then the SIMD computation speed is not better than the sequential one, and the energy waste is much greater !

This drawback of the SIMD parallelism can be minimized within the digital layer of the programmable retina, thanks to asynchronous propagation. The use of semi-static shift registers as digital memories [1] makes possible the following asynchronous phenomenon: a set of "active" pixels copy their value on their neighbors, that become active too, and so on until stability. This simple asynchronous function become very interesting if the connections are programmable. Indeed, it allows the computation of the geodesic reconstruction in one relaxed asynchronous propagation (see Figure 6(1)). This computation is faster, because the propagation takes much less time than the synchronous (dilation+intersection): the asynchronous propagation time has been estimated [1] to 1.5 ns per pixel, which correspond to the (unrealistic) control frequence of 12 GHz in synchronous. And it is much more efficient, because during the propagation, only the active pixels/processors consume energy.

What are the algorithmic implications of this asynchronous parallelism ? In fact, it is more than a simple economical way to get the reconstruction, because, thanks to the synchronous parallelism, the topology of the programmable mesh

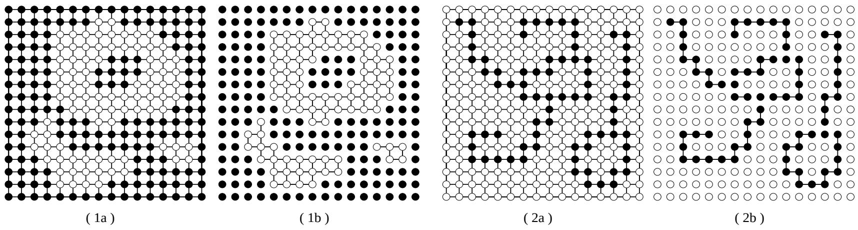



Fig. 6. Use of an asynchronous parallelism on a programmable topology. (1a) The geodesic reconstruction on a static mesh is obtained through iterated dilation/intersection. (1b) On a programmable mesh, after disabling the connections to background (black) pixels, the reconstruction is obtained through one asynchronous propagation from the marker pixels. (2a) When computing the SKIZ on a static mesh, the synchronous algorithm become very inefficient as soon as the background (black) become one pixel thick. (2b) On a programmable mesh, after disabling the connections to foreground (white) pixels, and to multiple points of the background, the reconstruction from the extremities of the background allows to remove the open curves in one asynchronous propagation.

can be constrained not only by the image itself, but by the result of an SIMD operator over the image. Figure 6(2) shows an example for the SKIZ, which is the other relaxation operator used in the watershed computation. In our system, the SKIZ is then computed in two phases: one synchronous, and one asynchronous.

More generally, the asynchronous operations can be algorithmically seen as an implementation of a limited class of FIFOs. The FIFO data structure and algorithmics [11] is of special interest for us, because its computing capabilities are, in some measure, diametrally opposed to the capabilities of the SIMD machine, and then the coexistence of the two systems on the same circuit is very promising.

## 6. Conclusions

On the base of the two last circuits design [2], [8], the next generation of programmable retina will have between 15 and 20 bits of digital memory per pixel, a control frequence adjustable between 1 and 100 MHz, and a resolution of 200x200 pixels. The expected computation time has been estimated (see Table VI) on the image of Figure 5(0), which is an 8 bits image of size 200x200, for a control frequence of 10 MHz, and for the two modes: synchronous, and mixed synchronous/asynchronous. Obviously, the mixed mode is more efficient, and in addition, unlike the synchronous mode, whose computation time is strongly dependent of the data, the time of the mixed mode is not much different in the worst case, because the asynchronous propagation time is almost negligible with respect to the synchronous period.

These results show the interest of a mixed synchronous/asynchronous parallelism in order to enhance the efficiency of cellular SIMD machines, by extend-

| | $watershed(I,p)$ | $watershed\_ter_{(2,8,2)}(I,p)$ |
|---|---|---|
| Synchronous | 1.4 ms | 1.9 ms |
| Mixed Sync/Async | 0.3 ms | 0.5 ms |

TABLE VI
Estimation of the computation time.

ing their computation capabilities, from the local to the regional. Such mixity is already possible at a low material cost and will probably exist in the next generation of programmable retinas, in which the algorithms presented here will be easily implemented. Now, other ways to exploit the asynchronous propagations should be explored. In particular, how to constrain the propagations by something "smarter" than the simple topology, typically local configurations that appear during the propagation, thus obtaining the implementations of more classes of FIFOs ? Future works will address this issue.

## Acknowledgments

The author wish to thank Thierry M. Bernard and Michel Schmitt for their contribution to this work.

## References

1. T. M. Bernard. Multi-purpose semi-static shift registers for digital programmable retinas. In *Phot. West Conf. on SCSSIA*, volume 3965. SPIE, 2000.
2. T. M. Bernard, B. Y. Zavidovique, and F. Devos. A programmable Artificial Retina. *IEEE JSSC*, 28-7:789–798, 1993.
3. S. Beucher and F. Meyer. *The Morphological Approach to Segmentation : The Watershed Transformation*. Edward Dougherty ed., 1991.
4. J.-E. Eklund, C. Svensson, and A. åström. VLSI implementation of a focal plane image processor - a realization of the NSIP concept. *IEEE VLSI*, 4-3:322–335, 1996.
5. P. Garda, A. Reichart, H. Rodriguez, F. Devos, and B. Y. Zavidovique. Une rétine électronique automate cellulaire. *Traitement du signal*, 5-6:435–449, 1988.
6. M. Grimaud. A new measure of contrast: the dynamics. In *Image Algebra and Morphological Processing III*, volume 1769, pages 292–305. SPIE, 1992.
7. A. Moini. *Vision Chips*. Kluwer Academics, 1999.
8. F. Paillet, D. Mercier, and T. Bernard. Second generation programmable artificial retina. In *Proc. IEEE ASIC/SOC Conf.*, pages 304–309, 1999.
9. M. Schmitt and L. Vincent. *Morphological image analysis. A practical and algorithmic handbook*. Cambridge University Press, to appear.
10. J. Serra. *Image analysis and mathematical morphology*. Academic Press, London, 1982.
11. L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE PAMI*, 13(6):583–598, 1991.