Smart retina as a contour-based visual interface

Paul Nadrag, Antoine Manzanera, Nicolas Burrus

Smart retina as a contour-based visual interface

Paul Nadrag
Politehnica University
Bucharest - ROMANIA
paul.nadrag@ensta.fr

Antoine Manzanera
ENSTA - LEI
Paris - FRANCE
antoine.manzanera@ensta.fr

Nicolas Burrus
ENSTA - LEI
Paris - FRANCE
nicolas.burrus@ensta.fr

Abstract

The purpose of this work is to provide a robust vision-based input device. In our system, a programmable retina is looking at the user who sends commands by moving his hand. The fusion between the acquisition and the processing functions of the retina allows a close adaptation to the lighting conditions and to the dynamic range of the scene. Thanks to its optical input and massive parallelism, the retina computes efficiently the contours of the moving objects. This feature has nice properties in terms of motion detection capabilities and allows a dramatic reduction in the volume of data to be output of the retina. An external low-power processor then performs global computations on the output data, such as extreme points or geometric moments, which are temporally filtered to generate a command.

1 Introduction

A vision-based interface is a system allowing to send symbolic commands to a computer just as a mouse or a keyboard does, but with visual interaction only. Using a camera as an input device is an attractive issue for numerous applications, from disabled persons assistance to intelligent home environments. To be really useful, such systems must not be invasive at all: it must neither impose to the user the wearing of any equipment, nor induce any constraint in the decoration of the scene observed by the camera. But detecting or recognizing the visual features independently of the scene content and its variation is a difficult task.

We are using in this work a vision system composed of a smart camera (programmable retina) and an external low-power processor (ARM). The interest of this approach is twofold:

1. The intimate combination of acquisition and processing functions of the retina allows to maintain a high fitness between the scene lighting properties and the further processing: in our case, the acquisition is adapted according to a local contrast criterion, in order to enhance the stability of the contours, which are the visual features chosen to detect the interaction.
2. The massive data reduction which is made between the optical input of the retina and the contour descriptor manipulated by the external processor is in perfect adequation with the I/O flow of the visual interface: image in, symbol out. By minimizing data transfers thanks to the massive parallelism and to the global measure output, the system is designed to optimize this flow from an energetic point of view.

Section 2 presents the Programmable retina within the vision system, its instruction set, acquisition and programming paradigms. Section 3 deals with image processing algorithms: the moving contour detection is detailed, and its properties are discussed. In Section 4, the higher level processing is presented and our first experiments in a drawing interface application are exposed. Finally, Section 5 outlines the contributions and presents some perspectives of this work.

2 Programmable retina

The programmable retina concept originates from the NCP (Neighborhood Combinatorial Processing) retinas [12], which were SIMD Boolean machines. The NSIP (Near Sensor Image Processing) concept [7] then allowed to process graylevel images. Now, the highly submicronic level of CMOS technology allows to put more and more powerful processing circuitry aside the photoreceptors while preserving good acquisition performance and resolution. The most recent circuit that has been fabricated and validated is a 200×200 retina called *Pvlsar 34*, with an elementary digital processor and 48 bits of memory within every pixel.

The algorithm presented in this paper was implemented on the architecture presented in Figure 1). The retina *Pvlsar 34* is both a CMOS sensor and a parallel machine. It is a grid of 200×200 pixels/processors connected by a regular 4-connected rectangular mesh. The processors operate synchronously, on their local data, a sequence of instructions being sent by the controller, which is the NIOS processor IP core of an Excalibur FPGA chip. The host processor or cortex is the ARM processor hardware core of the Excalibur. It can exchange data with the retina, modify the program sent by the NIOS to the retina, and is in charge of the higher levels of computation (i.e. non-image processing).

Every pixel/processor p of the grid G is composed of:

- one photoreceptor,
- one Analog to Digital Converter,
- 48 bits of digital memory $\{p_i\}, 1 \leq i \leq 48$.
- one Boolean Unit (BU), which can read some bits of the digital memory, compute a Boolean operation and write its output on one bit of the digital memory.

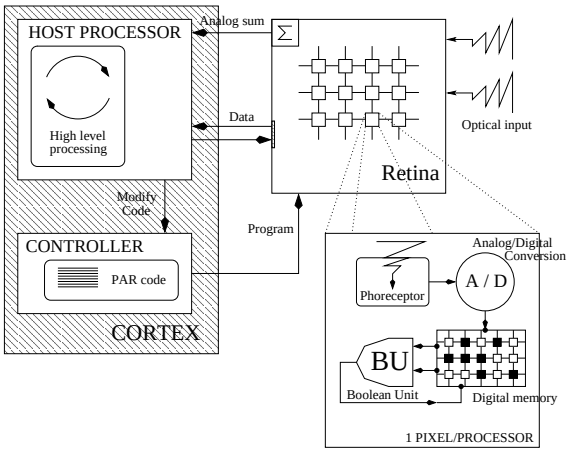

Figure 1. Architecture of the system composed of a retina and a cortex, with focus on one elementary processor.

For a given memory index i, the set $X_i = \bigcup_{p \in G} p_i$ is a 200×200 binary image called "bit plane". The elementary operations of the retina are massively parallel instructions of the form $Y = OP(X_1, X_2)$, where X_1, X_2, Y are 3 bit planes (not necessarily distinct) of the digital memory, and OP is any binary Boolean function (e.g. AND, XOR, AND NOT, etc). Every pixel of the retina shares 1 bit of its memory with each one of its 4 closest neighbours. This enables spatial interactions and image translations.

Regarding data extraction, there are two ways to output information from the retina:

- by transfering sequentially (translating) the image, to get the exact content of one or more bit planes of the digital memory.
- by using the Analog Global Summer, which provides in constant time an approximate measure of the number of 1s in any bit plane of the digital memory.

Although simple, this last feature is important as it provides efficiently global measures that are very useful to get spatial statistics or to detect the convergence of relaxation algorithms.

The Analog to Digital Conversion is performed during the acquisition, and is a plain part of the retinal program. Indeed, the acquisition of a n graylevels image is done by multiple readings of the photoreceptor which provide $(n-1)$ binary images (level sets) along the time (Figure 2). The photoreceptor (photodiode) is loaded to tension V_{max} and then unloads linearly with time, with a slope proportional to the amount of photons received. The graylevel is then computed in every pixel by counting (on $\log_2(n)$ bits), the number of times the tension at the bounds of the photodiode is below a threshold tension V_s. Thus, an acquisition is defined by the set of time indices $\{x_i\}_{i=0..n-1}$, corresponding to the reading instants of the photodiode.

The acquisition and digital coding of a n graylevels image is performed in $n + \log(n)$ operations, corresponding to the sequence of Boolean operations performed to add the $(n-1)$ level sets. Of course, the resulting digital image occupies $\log(n)$ bits of memory in every pixel. Then, the retinal program applies a sequence of arithmetic and logic operations that are completely written in software, at the bit level. However, if the number of operations (and thus the energy consumption), is not critical, the possibility of processing during acquisition allows a lot of elegant computations of image operators. In particular, many operators of mathematical morphology are naturally computed by combining their binary counterparts over the level sets of the image (this is a consequence of the extension scheme of the mathematical morphology from sets to functions) [11]. In our application for example, the morphological gradient (see Section 3) can be computed by summing the level lines (i.e. the contours of every level set) instead of the level sets.

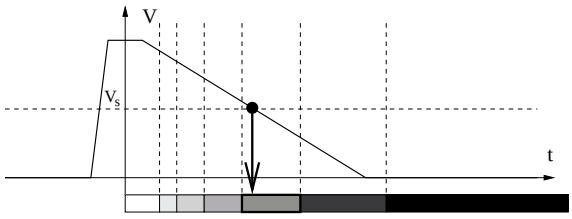

Figure 2. Graylevel acquisition by multiple reading of the photodiode.

But a more immediate consequence of this mode of acquisition is that it allows to control very simply the histogram of the image, by modifying the reading instants of the photoreceptor (e.g. logarithmic compression, gamma correction). Furthermore, it can be coupled with the analog global summer to adapt the reading instants to the graylevels repartition (e.g. histogram equalization). In the following of this section, we describe the adaptive acquisition we actually perform in our system.

As we wish to describe the static background of the scene (and then, by discrimination, the moving objects) by their contours, we need to compute contours that are both *rich* (to get all the structure from the objects, without regard to their contrast) and temporally *stable*. Meeting these two requirements is very hard, in particular when coding a high dynamic scene with a limited number of bits, because in this case, the contrast within the extreme (i.e. very light or very dark) regions is inexistent (saturation effect).

On the retina, a triple acquisition is performed: at each frame time t, 3 graylevels images are coded B_t, M_t and D_t corresponding respectively to bright, medium and dark regions of the graylevel distribution. Then $3(n-1)$ level sets are computed instead of $n-1$, corresponding to the 3 sets of time indices $\{b_i^t\}_{i=1..n-1}$, $\{m_i^t\}_{i=1..n-1}$, and $\{d_i^t\}_{i=1..n-1}$ respectively. These time indices are related as follows. First, the first index of the medium acquisition is temporally updated by a small increment:

$$\begin{array}{lll} \text{if } |M_t^{n/2}| < (w \times h)/2 & \text{then} & m_1^{t+1} = m_1^t + \varepsilon \\ & \text{else} & m_1^{t+1} = m_1^t - \varepsilon \end{array}$$

where $M_t^{n/2}$ is the (binary) $n/2$-level set of M_t (or equivalently, the most significant bit of the binary code of M_t), $|X|$ is the cardinality (the number of 1s) of X, w and h are the

dimensions of the image (here $w = h = 200$). This first condition corresponds to adapting incrementally the time reference such that the median value of the medium image M_t is half of the n graylevels dynamics. $|M_t^{n/2}|$, which is the only global measure needed by this algorithm, is computed thanks to the analog summer.

Second, the first indices of the two other acquisitions are placed in the time axis according to the relation:

$$b_1^t + \delta_{bm} = m_1^t = d_1^t - \delta_{md}$$

where δ_{bm} (resp. δ_{md}) represents the time offset - manually fixed - between the bright and medium (resp. between the medium and dark) regions.

Finally, every image is coded according to a logarithmic representation of the graylevels:

$$x_{i+1}^t = \alpha x_i^t$$

with $x = b$, m, and d respectively.

Figure 3(1-3) shows the 3 resulting images B_t, M_t and D_t. Recomposing a high dynamic range image from these 3 images, or reducing the representation of the image (e.g. from $3log(n)$ to $log(n)$ bits) can be done using different techniques that have been addressed in the literature [4] [6]. But in our application, we do not wish to recompose an image, but only to estimate the local contrast to compute the contour. We then simply compute a reduced representation of the contrast by computing the maximum of the local contrast in the 3 images:

$$G_B^t = \max(g_B(B_t), g_B(M_t), g_B(D_t))$$

where g_B is the morphological gradient (see Section 3).

Some other results that have been achieved by using the artificial retina can be found in the [2] and [10] papers.

3 Moving Contour detection

The static part of the algorithm used in the detection of the moving contours is based on the morphological edge detector described by Lee et al in [9]. This non-linear framework, using essentially min and max operations, is well suited to the Boolean computational model of the programmable retina. The morphological edge detection provides a measure of local contrast within every pixel, thanks to the dilation residue operator: let $f(x,y)$ be a grayscale image and b the structuring element, which is a set containing the origin $(0,0)$. The dilation of f by b if defined as:

$$(f \oplus b)(x,y) = \max_{(i,j)\in b} f(x-i, y-j)$$

The edges obtained by using the dilation residue, or (dilation-) gradient, noted by $g_b(x,y)$, will be:

$$g_b(f)(x,y) = (f \oplus b)(x,y) - f(x,y)$$

Like [9], we combine different gradients obtained using several structuring elements: $g_B(f) = \min(g_{b_1}(f), ..., g_{b_n}(f))$, in order to be less sensitive to noise and edge orientation. And according to Section 2, we maximize the gradient over the three acquired images: $G_B^t = \max(g_B(B_t), g_B(M_t), g_B(D_t))$: see Figure 3(4).

Once the (grayscale) edge map computed, we get the (binary) contours by employing a non-maximal suppresion technique, adapted from [3] to suit the massive parallelism of the retina. The method of suppresion is the following: let p be the pixel that is currently being processed (do not forget that the other 39,999 pixels of the retina are simultaneously undergoing the same sequence of operations) and his twelve neighbours, denoted using cardinal directions indices as follows:

$$\begin{array}{ccccc} & & p_{NN} & & \\ & p_{NW} & p_N & p_{NE} & \\ p_{WW} & p_W & p & p_E & p_{EE} \\ & p_{SW} & p_S & p_{SE} & \\ & & p_{SS} & & \end{array}$$

p will be selected as a contour point if at least one of the following four conditions is true:

1. $p_{NW} + p_W + p_{SW} < p_N + p + p_S > p_{NE} + p_E + p_{SE}$ (corresponding to a horizontal ridge)
2. $p_{NW} + p_N + p_{NE} < p_W + p + p_E > p_{SW} + p_N + p_{SE}$ (corresponding to a vertical ridge)
3. $p_{NN} + p_{NW} + p_{WW} < p_{NE} + p + p_{SW} > p_{EE} + p_{SE} + p_{SS}$ (corresponding to a ridge with an orientation of 135°)
4. $p_{WW} + p_{SW} + p_{SS} < p_{NW} + p + p_{SE} > p_{NN} + p_{NE} + p_{EE}$ (corresponding to a ridge with an orientation of 45°)

After this local maxima selection two last operations are completed to compute the contour image C_t: first, the contour with a gradient value of 1 are discarded to avoid the fake edges induced by the quantization effect on a grayscale ramp (those for example, due to the vignetting effect of the lens). Then, to eliminate salt-and-pepper-like detections, we also impose the condition that a pixel can be taken into account as a contour pixel only if it has at least two neighbours classified as contours too. This gives a lower rate of false detections, although it lowers the quantity of information theoretically obtainable from an image. Figure 3(5) shows an example of contours image.

To discriminate the moving contours from those of the static scene, we need to compare the current occurrences with the past edges. The basic approach is the following one: we keep track of the number of times when a pixel was a contour, by computing, for any instant t, two descriptors attached to every pixel, and related to two different timescales: the short term history (STH) and the long term history (LTH). At every frame, if the pixel belongs to an edge, then STH is increased. If not, it is decreased. The increment/decrement is adjusted to not exceed the interval allowed by the dynamics of STH. The update for the LTH is almost the same, except that instead of the contour C_t, the most significant bit of STH is used, and the timescale (i.e. the update period) is larger.

The computation of the moving contours is obtained from a logical AND between the current contour C_t and the pixels with a low LTH. As, obviously, a pixel can belong to both the contour of a moving object and the contour of the static scene, one iteration of reconstruction (or geodesic dilation) is performed: a dilation is computed on the intersection set, followed by a logical AND with C_t. This step allows to limit the disconnection on the resulting moving contours image E_t. The relevance of this basic approach is based on the fact that - by construction - the contours are one-pixel-thick, and then an object must be very still to let his contours be taken into account by the LTH. To enhance this behavior, however, and avoid the blurry, low-value disturbances which may be caused by the contours of a moving object in STH, we lower the increment frequency in STH for the pixels belonging to

E_t.

Another problem in the basic approach is generated by the fact that, generally, the moving objects are not transparent, which makes possible for a part of the background contours to be erased by the overlapping objects. The solution has been provided by the introduction of discrimination between the increment and the decrement of the LTH, the latter being discouraged. Thus, even if an object is obstructing the background scenery for a large amount of time (adjustable through parameterisation), it is possible for the object's contours to integrate themselves in the background (if the object is standing still) and, at the same time, for the background contours to stay longer in the scene, as if the new object was transparent. This property, together with the intrinsic thinness of the contours, significantly reduces the "ghost effect" (part of the static scene appearing as moving contours after being covered for a long time by a moving object). See Figure 3(6) for an example of moving contours image.

The measured times of execution for the different parts of the algorithm are the following: 10ms are needed by the retina to do its job (determination of the gradients, update of STH and LTH, extraction of moving contours, etc.), 8.6ms are needed for the ARM to compute the new trace (see Section 4) and 50ms are necessary to acquire the three initial images. The time needed to capture the three images is largely scene-dependent, a darker scene will demand a longer exposure time. As one can observe, the bottleneck is not induced by the processors in the retina themselves, but by the images' capture time. From an energetic point of view, the power consumption of the system is only a few tens milliwatts, 3 to 4 orders of magnitude as small as that of a PC.

Because of the high sensitivity induced by this algorithm, we get some false detections in addition to the true contours of the moving objects. Since some local filtering has already been applied, those false detections are gathered into small connected components. As objects of interest generally have important areas, we reduce the number of false alarms by filtering out small regions. This requires non-local computations on the image, surpassing the present possibilities of the retina. Thus, this step is performed by the host (ARM) processor. The further analysis, consisting in extracting and exploiting regional and global quantities on the moving contours is discussed in the next section.

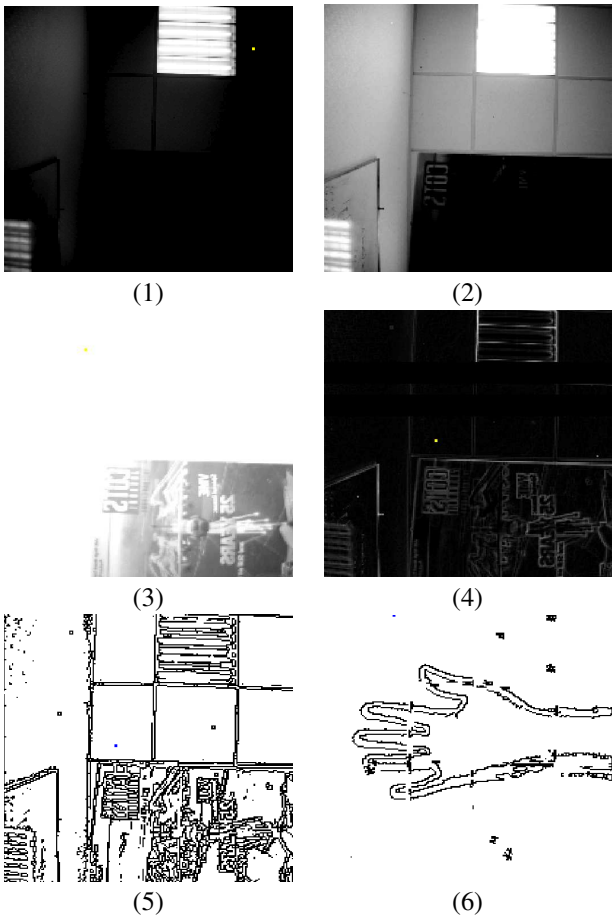

Figure 3. Acquisition and processing within the programmable retina: (1) short exposure acquisition B_t. (2) medium exposure acquisition M_t. (3) long exposure acquisition D_t. (4) maximum of the morphological gradient over the three acquisitions G_t^B. (5) binary contours C_t. (6) moving contours detected E_t.

4 Vision-based interface

The purpose of the image processing techniques presented above is to allow a stable and complete extraction of the contours of the moving objects, without imposing constraint of uniformity in the background. With respect to the input data flow, this feature represents a small amount of data which can be quickly extracted from the retina and processed by the external processor. We are currently beginning the exploitation of this visual feature for higher level processing. We present in this section the first experiments of visual interaction we have performed.

In these experiments, we are testing a drawing interface: the user draws a bidimensional curve by moving his finger or a pen above the retina, which is oriented toward the ceiling. Such interface can be used to input hand drawn characters or sketches [5], but also some simple hand gestures [8], for which the curve represents a 2d model to be further processed and recognized.

One difficulty in developping visual interface is the necessity of several modalities in the interaction, just like the action of a mouse stops if it is dropped, or changes if its button is depressed. We wish to use the visual interface alone, without interaction of the mouse or keyboard. This implies that several parameters of the visual feature must be exploited, in order to define not only the position of the "tool", but also the action modality. In our example, the 3 modalities are (1) Draw, (2) Do nothing, (3) Erase.

So the first parameter computed from the visual feature is a global one ; it is the orientation of the moving object. Like [1], we compute the second order moments (v_x, v_y), i.e. the variances of the two coordinates of the moving contour points. The rough orientation is then used to define the modality: if v_x/v_y is superior to a given threshold σ_D, then the "Draw" action is selected. If v_y/v_x is superior to a given threshold σ_E, then the "Erase" action is selected. Otherwise,

the "Do nothing" action is selected.

The second parameter extracted from the moving contour is a local one: the extreme left position of the moving objects, if any (there must be a connected curve with at least N_{min} points) is used as the candidate position of the tool. This candidate position is checked according to a predictive filtering (it must not be too far from the previous position of the same tool). If the position is confirmed, then the following action is applied on the drawing widget: if "draw" is selected then create a line between previous and current position ; if "erase" is selected, then erase all points situated at the right of the current position.

On Figure 4, we can see the tracking system in action. Special attention should be paid to the transitions over the very textured areas (the neon grid, for instance). Because we work with contours only, we could expect some difficulties when crossing these areas, but by using a wisely chosen set of filters (enclosure and selection based on the elements' size), we are able to obtain a very high detection rate of the fingertip.

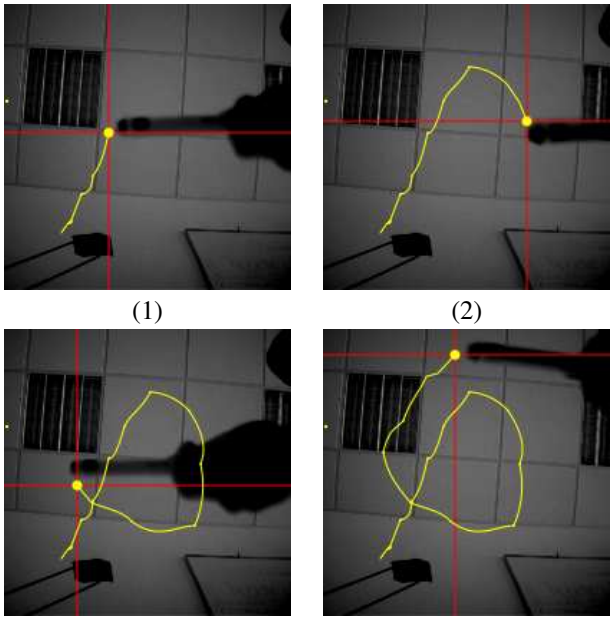

Figure 4. Drawing example using the visual mouse

5 Conclusion

We have presented in this paper a project of visual interface based on the extraction and processing of the contours of the moving objects. At the moment, we have mainly developped the low-level part of the project, which corresponds to the extraction of the moving contours, using a programmable retina, both as camera and parallel computer. The combination of acquisition and image processing allows a smart adaptation to the lighting properties of the scene. The visual interaction can be seen as a huge data reduction, turning video flow into symbols. The computational model of the programmable retina limits the data transfers to the minimum, and then reduces the energy consumed by this data reduction.

The higher level (interpretation of the moving contours) is being investigated, and will be pursued in the following months. We have presented our first experiments, based on single point/multiple actions interface. The perspectives of this work are also: global hand posture estimation for controlling a 3d mouse, and traking a 3d model to recover the complete posture of the arm/hand/fingers.

6 References

[1] S. Ahmad. A usable real-time 3d hand tracker. In *28th Asilomar Conference on Signals, Systems and Computers*, pages 1257–1261, 1995.

[2] N. Burrus and T.M. Bernard. Adaptive vision leveraging digital retinas: Extracting meaningful segments. In *Advanced Concepts for Intelligent Vision Systems*, pages 220–231, 2006.

[3] B. Chanda, M.K. Kundu, and Y.V. Padmaja. A multiscale morphologic edge detector. *Pattern Recognition*, 31(10):1469–1478, 1998.

[4] P. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH 97*, pages 369–378, aug 1997.

[5] I.A. Erdem, M.E. Erdem, V. Atalay, and A.E. Çetin. Vision-based continuous graffiti-like text entry system. *Optical Engineering*, 43:553–558, 2004.

[6] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. In *ACM SIGGRAPH 2002*, pages 249–256, july 2002.

[7] R. Forchheimer and A. Astrøm. Near-sensor image processing: a new paradigm. *IEEE trans. on Image Processing*, 3:736–746, 1994.

[8] T. Huang and V. Pavlovic. Hand gesture modeling, analysis, and synthesis. In *International Workshop on Automatic Face and Gesture Recognition*, pages 73–79, 1995.

[9] J. Lee, R. Haralick, and L. Shapiro. Morphologic edge detection. *IEEE Journal of Robotics and Automation*, 3:142–156, 1987.

[10] J. Richefeu and A. Manzanera. A morphological dominant points detection and its cellular implementation. In *ISSPA 2003 Proceedings*, volume 2, pages 181–184, 2003.

[11] J. Serra. *Image analysis and mathematical morphology*. London Academic, 1982.

[12] B. Zavidovique and G. Stamon. Bilevel processing of multilevel images. In *Pattern Recognition and Image Processing*, Dallas, TX, August 1981.