

# Fast and accurate edge-based segmentation with no contour smoothing in 2-D real images

Giancarlo Iannizzotto, Lorenzo Vita

## Abstract

In this paper we propose an edge-based segmentation algorithm which is fast, has a low computational complexity and introduces no unwanted smoothing in the retrieved contours. Moreover, the contours are always given as closed chains of points, resulting in a very useful base for subsequent shape representation techniques.

## Keywords

2-D image segmentation, active contours, edge extraction.

## I. INTRODUCTION

**I**MAGE segmentation (that is, partitioning an image in distinct areas of interest) is a fundamental step in most of the applications of image analysis. In particular, in computer vision segmentation precludes the appropriate representation of the objects contained in an image and their classification according to specific features of interest [1]. One of the approaches most frequently adopted is based on the extraction of the *contours* of the objects: the main reason for this choice is that the most part of the information about the shape of a bidimensional object resides in its contour. So, the application of a method of segmentation based on the whole area of the object would be uselessly expensive. On the other hand, it is often said that contour-based image segmentation is error-prone due to the sensitivity of the operators used to detect object contours. Moreover, contour-based techniques usually need some additional instrument to obtain closed contours, a condition which is necessary to characterize a region in the image. Actually, the discussion about which approach (contour-based or area-based) is better is still open, and is not likely to end up.

The purpose of this work is to present a technique for contour-based segmentation of bidimensional images. This technique permits to quickly and accurately trace the contours of the objects, even if their shape is complex and they are nested at more than one level. The input of the system consists of a binary image, whose points are obtained through any algorithm of contour marking (edge extraction), based for example on the grey-level gradient. The output consists of a sequence of closed chains of points, each representing the contour of a single object. The Cartesian coordinates are given for each point of a contour. In the following paragraphs we will briefly focus our attention on the problem of contour-based image segmentation, will describe some related works and will examine the method that we developed, justifying our choices and presenting some experimental results. In the last paragraph we will draw some conclusions about the work we have done, and will disclose our future lines for its development.

## II. CONTOUR-BASED IMAGE SEGMENTATION

Contour-based image segmentation techniques usually start with some pre-processing steps whose purpose is to mark those points of the image that are more likely to belong to the contours of the objects (*edge detection*). The number and the computational complexity of such steps vary according to the techniques selected, but nowadays some hardware devices, which are based on the grey-level gradient, can obtain very good results of edge detection practically in real time. In any case, the gradient operator is a local operator, and so it is highly parallelizable, as well as not being too complex. Even the application of a threshold to the gradient image (in order to obtain a binary one) is not a computationally critical operation, and even though it can be easily made in hardware, it is highly parallelizable and therefore quite quick even if it is made

G. Iannizzotto is with the Department of Mathematics, University of Messina, C.da Papardo, Salita Sperone, 98166, Messina - Italy. E-mail: ianni@ingegneria.unime.it .

L. Vita is with the Istituto di Informatica e Telecomunicazioni, University of Catania, Viale A. Doria 6, 95025 Catania - Italy. E-mail: vita@iit.unict.it

through software. Reference can be made to the wide literature on this topic, for the selection of the edge detection operator, and of the specific thresholding strategy. See also [2] [1] [3]. At the end of the operation of edge detection, what we obtain is generally an image whose background is characterized by the black color, while the edge points of the objects take all the values of grey ranging from black to white, as a function of the level of *edgeness* that have been assigned by the operator. At this point, segmentation is referable to exploiting the values of edgeness to generate a contour which *links* the points characterized by a higher level of edgeness [4] [5] [6]. This linking operation must produce closed, one-pixel wide chains, in order to meet the requirements for a segmentation algorithm. In some specific situations, however, a segmentation algorithm must be able to identify some areas of arbitrary length but *1 pixel wide* that is, *open lines*. Such cases seem to be only apparently in contrast with the hypothesis of closed contour: in fact, an open line, considered as a region of the image, is always surrounded by a closed contour representing its edge with the rest of the image. In conclusion, we can accept that an open line 1 pixel wide is segmented by surrounding it with a closed contour 3 pixel wide (2 for the edge and 1 inner) provided that, according to some heuristic considerations, this line is correctly recognized as a line without inner points.

Very often, objects outlines are not represented by a single contour, but by several contours for each object. This is the case of an object with one or more holes, whose outline is represented by the outside contour and by the contours of each hole. In such occasions, the segmentation algorithm must be able to recognize the presence, within the outside contour, of other contours that are still part of the object and must be captured. The output of the algorithm will consist of two-level hierarchical structures, whose first level will contain the outside contour, and whose second level will contain the inside contours. In other occasions we have to identify the presence of objects inside other objects, often with several nesting levels: in this case the algorithm of segmentation will have to check, inside each contour identified, the presence of any new contour, and the output will consist of  $n$ -level hierarchical structures, with  $n$  unknown. An upper level of processing will have to determine how many and which objects there are on the scene, according to the base of the hierarchy of extracted contours. Of course, this level will no longer be part of the segmentation subsystem, but will belong to the upper layer, that is, *identification* subsystem.

### III. RELATED WORK

As described so far, segmentation is referable to an operation of *linking* of edge points in closed chains, operated in so that to meet the requirements fixed in the previous paragraph. The techniques generally used in this phase may be roughly subdivided into four categories:

1. Edge-tracking: starting from a given edge point, we move to all directions, looking for another one that could meet certain requirements: if we find it, the contour passes also for that point, and we continue looking for the following point; if we do not find it, the contour ends. A great number of variations of this technique are available in literature: the techniques based on hysteresis adopted by Canny [4] and those based on dynamic programming [7].
2. Fitting of parametric curves: by using several analytical and/or heuristic techniques, we look for the best composition of a certain number of curves, so to try and obtain that one curve corresponds to each object. Unless the nature of the objects in the scene is well known, this technique results to be computationally expensive and not very accurate [2]. However, its advantage is a direct analytical (and in general very compact) representation of the contours of the objects. This therefore makes the following steps of recognition much simpler.
3. Use of *active contours*: these are models representing open or closed curves that, having a very high capacity of morphological adaptation, can almost perfectly adapt to contours of any shape. These models are called *active* because they automatically respond to specific characteristics of the points of the image to which are applied, by changing their shape consequently. For example, an active contour can respond to the edgeness values of the points of the image, by changing its shape till it passes only for all the points characterized by values of edgeness locally higher (*edgeness local maxima*). A particular type of active contour is the *snake*, that does not only respond to the characteristics of the points of the image (through the minimization of a quantity called external energy), but also responds to specific internal laws ruling shape and way of deformation tending to minimize a quantity called internal energy [8] [9]. Some other variations of the snake are represented by

its parametric formulations, whose purpose is to change the problem of the minimization of internal/external energy into a problem of estimate of parameters (the control points of a spline, for example), much quicker and computationally less expensive (snake spline [10] [11]). Snakes are generally used as semi-automatic techniques of segmentation, because they are usually very sensitive to noise and to edginess local maxima. For this reason, on conditions of high noise, they tend to *lose contact* with their primary target, if they are not properly helped by interactive user operations. On the other hand, in some recent works with snakes very good results were obtained in the field of automatic segmentation, and even the problem of the *hierarchy of contours* [12] was dealt with. Nevertheless, they generally seem to be suffering from an intrinsic high computational complexity, and from an effect of *contours smoothing* which can be undesired. Concerning this, let us consider the case in which the most part of the useful information lies in the characteristics of unevenness of the contour rather than in its overall shape: an important example is represented by the diagnosis of tumors through images, for which the characteristics of unevenness and the way the contours are fringed and irregular are the basic information for the diagnosis [13]. In the light of such needs the development of a segmentation technique, which maintains the greatest accuracy in the recovery of contours, seems definitely more convenient. Then we can apply, only if appropriate, a fitting algorithm.

4. Use of Hough's Transform or of one of its many variations: it was proved [14] that Hough's transform can be considered a generalization of the concept of *deformable contour* that is the basis of active contours.

The technique we propose in this work can be classified in the third category, if we deem it to be also extended to non-analytical but *algorithmic* models. A first work about it, which some time ago was still at a preliminary stage of experimentation, was presented in ICIP '96 [15].

#### IV. THE PROPOSED METHOD: AMOEBEA

The problems that we were going to face when we started this project were substantially two:

- To obtain an accurate reconstruction of the contour, which did not introduce undesired smoothings, and which could exactly follow the contour of the object;
- To develop a quick and effective algorithm, easily parallelizable and whose computational complexity is quite low.

We started from the assumption that each contour should have been represented by a closed chain consisting of the points of the contour itself, possibly interpolated if some holes had appeared. We liked the idea of the snake, but we wanted to avoid the need for fitting the points of each chain with curves or sequences of curves (splines), that we should re-calculate at any iteration of adaptation of the chain to the contour. Besides, there not seem to be, at present, curves that do not introduce any smoothing effect.

The biological approach gave us the key for dealing with the problem correctly: modeling the chain as a sequence of points each having its own capacity of movement, the same way as a living tissue consists of a set of independent cells, strictly connected at the same time. Such points, that we will call *MOVels* (*MOVing ELeMents*), reproduce, move and die following a very simple set of rules which are all based on strictly local information. They therefore meet the need for parallelizability of the algorithm, and at the same time maintain continuity and coherence in the movement of the chain as a whole. The chain needs (as the only starting assumption) that all the objects that must be revealed are at its inside: it is therefore obvious to initialize it always as a frame surrounding the whole portion of interest of the image, so that all the objects of interest are originally internal to it. With a quick sequence of steps, the chain adapts its shape and dimensions until it follows exactly those of the objects in its inside. As we will see, the chain can also reveal the presence of several objects, both at the same level and nested each other even at more levels.

The characteristics and behaviors of the described entity closely recalls the one of an amoeba, and this name was given to the whole algorithm.

##### A. The life cycle of Amoeba.

The life cycle of Amoeba starts when the MOVels it is made of are laid, in the form of a closed and continuous chain, on the binary image; this can take place in four ways:

1. Automatic laying of a chain on the outer edge of the image, so to include all the objects of the image at the interior of the chain (phase of initialization of unsupervised segmentation);

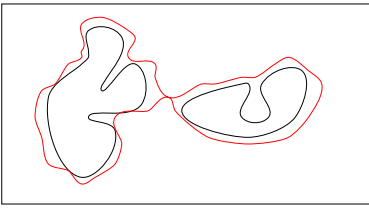


Fig. 1. Division of Amoeba

2. The user lays by mouse operations a closed and continuous chain, so to include at least one object in its inside (phase of initialization of the supervised segmentation);
3. A chain lays another one inside itself: this is one of the two ways of reproduction of Amoeba, and its purpose is to allow the revelation of nested objects; the outer one is revealed by a chain, then this chain lays a *son* chain inside the object revealed, and this son chain starts its life cycle.
4. A chain, during its life cycle, can surround two or more objects at the same time, not nested (see figure 1): in this case, the chain **splits up** in several twin chains, each of which surrounds only one object, and starts an independent life cycle. This is the second way of reproduction of Amoeba.

Amoeba, once it has been laid, starts moving by contracting and adapting its shape to the objects that it finds at its inside. The MOVels it consists of always move in such a direction that they make the chain contract evenly in all directions. When a MOVel finds a contour, it lies on the contour and stops moving. The MOVels adjacent to the chain continue moving (if they still can), and by doing so they can generate new points, in order not to leave any holes in the chain. The rules for the movement were designed so to avoid anomalous behaviors of Amoeba, that results to have (thanks to the mechanism of division and to a threshold control on the minimum dimension of a chain) a good rejection to punctiform noise.

If a chain does not find any object with significant dimensions at its inside, it closes in itself and disappears: this ends its life cycle.

### B. The life cycle of a MOVel

MOVels have their life cycle too, and the complex of life cycles of all MOVels of a chain forms the life cycle of the chain itself.

A MOVel starts when the chain that contains it is laid, or during the phase of movement of a MOVel that is departing from one of its two adjacent ones. In this case, a procedure of interpolation is activated, that closes any holes left by the departing MOVel. In conclusion, even this can be considered a reproductive phase, even if very rudimentary.

An active MOVel moves (following simple algorithmic rules based only on its positions and on the ones of its two adjacent MOVels in the chain) in such directions that the contraction behavior of the chain that includes it, is maintained. When a MOVel meets a contour point, it lays on it and becomes idle. This does not mean that it is dead: it only stops moving, causing the activation of interpolation mechanisms for any adjacent that can still move.

A MOVel dies when an adjacent one comes in contact with it, following the rules of the movement, or when the chain that contains it dies because it has become too small.

### C. How Amoeba moves

Amoeba always moves by contracting, and its motion is given by the composition of the motions of each MOVel. Each MOVel calculates the direction and the sense of its movement according to only two informations, that are the positions (with regards to it) of its antecedent and of its subsequent in the chain. Such informations are used as an index of access to a Look-Up Table (LUT) that gives the direction and the sense of motion for any configuration of the index. If the new position calculated for the MOVel is clear, the MOVel moves; if the position is occupied by another adjacent MOVel, the latter is deleted and replaced by the one that is moving. If the position is occupied by a not adjacent MOVel, the chain splits up in two parts, because there is evidently a narrowing indicating (see figure 1) that the chain is closing around two distinct objects. Finally,

if the new position is occupied by an edge point, the MOVel stops, and becomes idle. The complexity of the described algorithm is evidently  $O(n)$ , i.e. linear in the number of MOVels.

#### D. How Amoeba recognizes open contours

An open contour, that is, a line consisting of edge and not closed, is first surrounded by Amoeba as an object consisting of only edge points and not of internal points. This property is revealed in a second phase, when Amoeba, after finishing to include the contour, tries and lay a son chain *at its inside*: in this case, since there are no internal points, it cannot appen and the contour is marked as open.

### V. EXPERIMENTAL RESULTS

The algorithm described so far was implemented in C standard language and compiled with GNU compiler gcc version 2.7 on a Linux system, and we used libraries X11 for the graphic part. However, several tests were made on Intel-based, SUN SPARC with SOLARIS 2.5, SILICON GRAPHICS Indy platforms, with perfect portability. At present, we cannot report comparative tables of the performances of the algorithm proposed with reference to other ones present in literature, since their original implementations are not available. Nevertheless, according to the analysis of computational complexity, we can easily point out the higher ease and the intrinsic simplicity of the one proposed.

Some sets of photographs and some synthesis images were selected for the experimentation of the system carried out: the purpose of synthesis images was mainly to produce some sample pictures for this paper. The images first went through a gradient operator of pixel intensity (sobel operator), then through a threshold operator, for deleting the points whose gradient revealed lower than a certain threshold. This process corresponds to one of the most elementary and primitive techniques for the extraction of contours, and here was adopted for pointing out at the best the effects of the algorithm presented here. Any improvement in the technique of edge extraction adopted will involve an improvement in the final results of segmentation. The images of *edgeness* obtained then went through the Amoeba segmentation algorithm, whose result is a tree representing the hierarchy of nested objects whose root is represented by the primary chain, as it appears immediately after the initialization. Of course, such chain is not an integral part of the set of extracted objects, but represents the image area within which Amoeba moved. The chains represented by the nodes of the tree were saved on a file and then plotted with gnuplot.

In figure 2 we presented one of the sets of photographs used for the test, representing leaves of varous shapes: this choice was due to the extreme variability of their contours, ranging from very angular shapes to rather circular ones. Besides, since the analysis of this type of images often preludes the classificazione of the objects present in them, and in the specific case of leaves their classification requires a very high accuracy in pointing out the contours, we deemed this example useful for showing the characteristics of accuracy of Amoeba.

In figure 3 we can see the gradient images so as they are passed to Amoeba, and the outside contours pointed out by Amoeba. Since leaves are characterized by a high texture, we did not consider the need for activating the reproduction of Amoeba at their inside.

In figure 4, we can evaluate the results of the application of Amoeba to the gradient images of leaves: the contours were perfectly extracted, without any undesired effect of smoothing, and in a very short time. For the image in figure 5, a non-optimized prototype of Amoeba with an animated graphic interface required (on a PC Pentium 133 with Linux) about 1 second for terminating. The closed chains of points were saved on a file and then plotted with gnuplot.

The capacity of revealing multiple objects at the inside of an image is pointed out in figure 5: we can notice how Amoeba revealed the contours of each object in the image with great accuracy, by segmentating it perfectly.

The capacity of Amoeba to reproduce at the inside of the objects is shown in figures 6 and 7: in the first one we can see the departure shapes, while in the second one we can see the chains produced by Amoeba.

In this experiment we used a synthetic image only for pointing out the characteristics of Amoeba better: the algorithm works in the same way with photographs. As an example, in figure 8 are reproduced the photo of a young lady, the image of Canny contours and the Amoeba screen-shot. As it can be seen, even if the

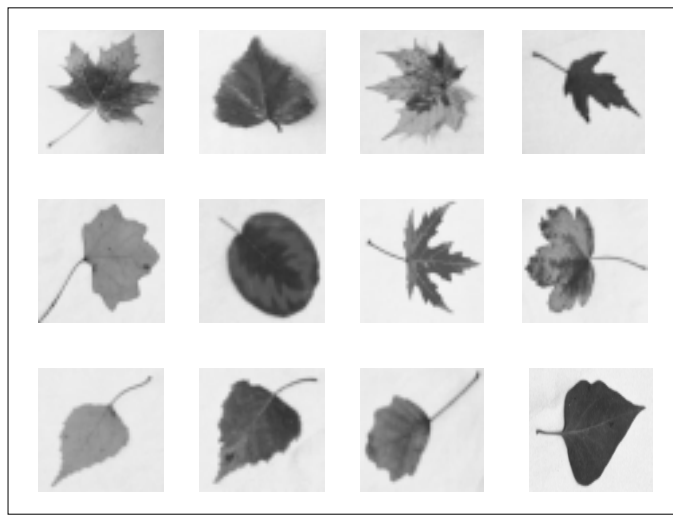


Fig. 2. The *leaves* set of images.

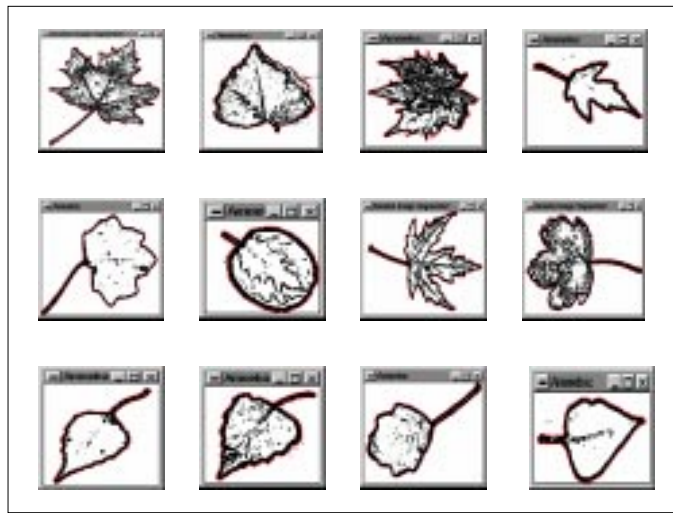


Fig. 3. Amoeba screen-shots for the images of the *leaves* set.

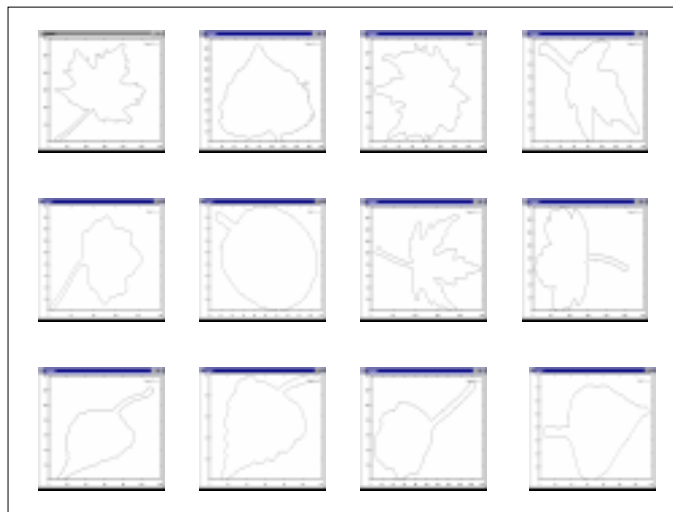


Fig. 4. Plotted chains resulting from Amoeba.

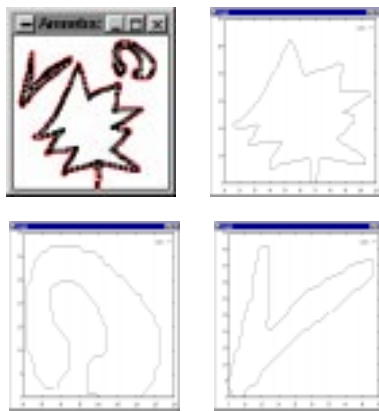


Fig. 5. The *Star* gradient image and the Amoeba resulting chains

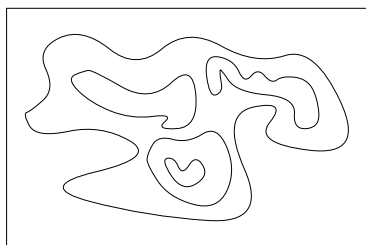


Fig. 6. The *Nesting* image

contours extracted with the Canny algorithm are more dense, those extracted by the Amoeba algorithm are all closed contours and are on the whole more meaningful.

## VI. CONCLUSIONS AND FURTHER WORK

In this paper we have presented a new algorithm for the segmentation of bidimensional photographic images and for the extraction of the contours of the objects that they contain. The algorithm differs from other ones present in literature for its accuracy in extracting the contours, for its low computational complexity and for the ease with which it can be optimized and parallelized. The characteristics and the performances in terms of accuracy of results are presented in a series of examples and tests, made on standard platforms (pc Pentium+Linux, SUN sparystation+Solaris, Silicon Graphics Indy). This project is continuing on several ways: optimization of the implementation of Amoeba with the purpose of passing from a prototype to a system with optimal performances, parallelization, introduction of the information regarding the gradient in the algorithm (Amoeba is currently working on binary images), extension of the algorithm to a multiscale model.

## REFERENCES

- [1] Dana H. Ballard and C.M. Brown, *Computer Vision*, Prentice-Hall, 1982.
- [2] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Addison-Wesley.
- [3] K. R. Castleman, *Digital Image Processing*, Prentice Hall, 1996.



Fig. 7. The chains resulting from the *Nesting* image



Fig. 8. The *young lady* image, its Canny contours and its Amoeba screen-shot.

- [4] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. on PAMI*, vol. 8(6):679-698, June 1986.
- [5] J. F. Canny, "Finding edges and lines in images," Tech. Rep. Tech. Rep. AI-TR-720, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1983.
- [6] J. R. Parker, *Algorithms for Image Processing and Computer Vision*, Wiley Computer Publishing, 1997.
- [7] Milan Sonka, Vaclav Hlavac, and Roger Boyle, *Image Processing, Analysis and Machine Vision*, Chapman & Hall Computing, 1993.
- [8] K. F. Lai and R. T. Chin, "Deformable contours: Modeling and extraction," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1994.
- [9] K. F. Lai and R. T. Chin, "On regularization, formulation, and initialization of the active contour models (snakes)," in *Asian Conference on Computer Vision*, 1993.
- [10] Myron Flickner, Jim Hafner, Eduardo Rodríguez, and Jorge Sanz, "Fast least-squares curve fitting using quasi-orthogonal splines," in *Proceedings of the 1st International Conference on Image Processing*, 1994, vol. I, pp. 686-690, IBM techreport RJ 9819.
- [11] Myron Flickner, James Hafner, Eduardo J. Rodríguez, and Jorge L. C. Sanz, "Periodic quasi-orthogonal spline bases and applications to least-squares curve fitting in digital images," *IEEE Transactions on Image Processing*, vol. 5, no. 1, pp. 71-88, Jan 1996.
- [12] R. Malladi, J. A. Sethian, and B.C. Vemuri, "Shape modelling with front propagation," *IEEE Trans. on PAMI*, vol. 17(2), Feb. 1995.
- [13] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas, "Fast nearest neighbor search in medical image databases," Tech. Rep., University of Maryland, Dept. of Computer Science, 1996.
- [14] K. F. Lai, "Deformable contours: Modeling, extraction, detection and classification," Tech. Rep. PhD. Thesis, University of Wisconsin-Madison, 1994.
- [15] G. Iannizzotto and L. Vita, "A fast, accurate method to segment and retrieve object contours in real images," in *International Conf. on Image Processing*, Lausanne, Switzerland, 1996.