

Exact Euclidean Distance Function by Chain Propagations

Luc Vincent*

Division of Applied Sciences, Harvard University

[*Proc. IEEE Conf. on Comp. Vision and Pattern Recog.*, pp. 520–525, Maui, Hawaii, June 1991]

Abstract

Up to now, all the known Euclidean distance function algorithms are either excessively slow or inaccurate, and even Danielsson's method produces errors in some configurations. We show that they are due to the local way distances are propagated in images by this algorithm. To remedy these drawbacks, an algorithm is introduced, which encodes the objects boundaries as chains and propagates these structures in the image using rewriting rules. The chains convey Euclidean distances and can be written above one another, thus yielding *exact* results. In addition, the proposed algorithm is particularly efficient. Some of its applications to skeletons and neighborhood graphs are described.

1 Introduction

Given a set X in a metric space (E, d) , the corresponding distance function dist_X [8] associates with each point p of E its distance to X :

$$\text{dist}_X \begin{pmatrix} E & \longrightarrow & \mathbb{R}^+ \\ p & \longmapsto & d(p, X) = \inf\{d(p, q), q \in X\}. \end{pmatrix} \quad (1)$$

In this study, we restrict ourselves to the most common case where $E = \mathbb{Z}^2$. The distance function is an extremely interesting tool in the field of image analysis and mathematical morphology [10]. It is at the basis of many other transformations, like skeletons, skeletons by influence zones and ultimate erosions (see § 5), and it is thus particularly important to compute it at best. The purpose of this paper is to introduce an efficient algorithm for computing exact Euclidean distance functions.

After some reminders on the concepts and notations used in the sequel, § 3 recalls the principles of one

of the best known Euclidean distance function algorithms, namely Danielsson's sequential algorithm. It only produces approximate results, and the analysis of its error cases leads us in § 4 to propose a new method, based on the propagation of chains in the image under study. This results in a perfectly accurate algorithm which, in addition, runs faster than Danielsson's. Lastly, some examples and applications of this algorithm to the computation of skeletons, Delaunay triangulations and Gabriel graphs are provided.

2 Discrete Images and Grids

In this paper, we regard an image I as a mapping from a rectangular domain $D_I \subset \mathbb{Z} \times \mathbb{Z}$ into a set of grey-levels. The neighborhood relationships between pixels are provided by a digital grid G , subset of $\mathbb{Z}^2 \times \mathbb{Z}^2$ whose elements are called *edges*. The set of the neighbors of a pixel p with respect to G is given by $N_G(p) = \{q \in \mathbb{Z}^2 \mid (p, q) \in G\}$. G induces a discrete distance d_G in the plane \mathbb{Z}^2 : the distance $d_G(p, q)$ between two pixels p and q is defined as the minimal number of grid edges to cross to go from p to q . The most commonly used grids are the square one, in 4- or 8-connectivity, and the hexagonal one, according to which each pixel has six neighbors. Here, images will not be considered only as graphs, but also as subsets of the continuous plane \mathbb{R}^2 , equipped with the orthonormal coordinate system (O, \vec{i}, \vec{j}) . Accordingly, the 4 unit vectors of the 4-connected square grid are \vec{i} , $-\vec{i}$, \vec{j} and $-\vec{j}$, whereas the six unit vectors of the hexagonal grid are given by:

$$\begin{aligned} \vec{u}_0 &= \vec{i} & \vec{u}_3 &= -\vec{i} \\ \vec{u}_1 &= \frac{1}{2}\vec{i} + \frac{\sqrt{3}}{2}\vec{j} & \vec{u}_4 &= -\frac{1}{2}\vec{i} - \frac{\sqrt{3}}{2}\vec{j} \\ \vec{u}_2 &= -\frac{1}{2}\vec{i} + \frac{\sqrt{3}}{2}\vec{j} & \vec{u}_5 &= \frac{1}{2}\vec{i} - \frac{\sqrt{3}}{2}\vec{j} \end{aligned} \quad (2)$$

(see Fig. 1). Each pixel of a hexagonally sampled image has integer coordinates in the system $(O, \vec{u}_0, \vec{u}_1)$. In the sequel, both these grids are used, the latter being often

*This work was supported in part by *Dassault Electronique* and the *National Science Foundation* under Grant MIPS-86-58150, with matching funds from *DEC* and *Xerox*.

preferred in morphology because of its good behavior with respect to connectivity [12, chap. 1].

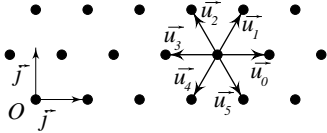


Figure 1: Elementary vectors of the hexagonal grid.

The discrete distances induced by the 4-connected square grid and the hexagonal grid are respectively denoted d_4 and d_6 . They are the most commonly used in practice, since their efficient implementation is easily achieved by various methods [13]. However, denoting by d the Euclidean distance in the space \mathbb{R}^2 , one can see that for certain orientations, $d_6(p, q) = (2/\sqrt{3}) d(p, q)$ [13]. Similarly, in some cases, $d_4(p, q) = \sqrt{2} d(p, q)$. This means that the distances provided by d_4 and d_6 are non-isotropic and induce errors of up to 40% and 15% respectively with respect to the Euclidean distance! Now, the efficient computation of Euclidean distance functions is a difficult task; the only simple algorithm, rather greedy, consists in entirely scanning the image under study and for each pixel p , *entirely* scanning the set of feature pixels in order to assign to p the smallest possible distance... For these reasons, many of the methods found in literature rely on compromises between computational efficiency and accuracy. Their principle is to add new edges to the used grid and to associate them the Euclidean distance between the two vertices they join [13]. They work in a sequential fashion and are detailed in [1]. Alternatively, the algorithm introduced in § 4 yields exact results.

3 Danielsson's Algorithm

The algorithm proposed in 1980 by Danielsson [2] to compute Euclidean distance functions is of *sequential* type [7], i.e., relies on the following two principles:

- The image pixels are scanned in a predefined order, generally video or anti-video,
- The new value of the current pixel, determined from the values of the pixels in its neighborhood, is written directly *in the same image*, so that it is taken into account to determine the new values of the not yet considered pixels.

Danielsson's algorithm differs from the techniques described in [1] in the sense that firstly, it requires four

image scanings instead of only two and secondly, it does not propagate distances, but vectors: it yields a *vector image* \vec{J} where each pixel p is assigned a vector \vec{v} such that $p - \vec{v}$ is (one of) its closest feature pixel in the original image I . To get an actual distance function from \vec{J} , it suffices to take its norm. Note that the square of the norm of every vector of \vec{J} is always an integer, so that no floating-point calculus is involved by the procedure. This algorithm is thus rather efficient, and its typical execution time on a *Sun Sparc Station 1* is of 4 seconds for a 256×256 image.

Unfortunately, the distance (vector) assigned to a pixel p by this algorithm is only determined from the distances associated to the pixels in $N_G(p)$, G being the underlying grid. As illustrated by Fig. 2 (in the case of the 4-connectivity), this is not sufficient to produce exact results in all cases. On this figure, the original image has only three feature pixels: a , b and c , and it is easy to see that the vector which will at last be assigned to p is either \vec{ap} or \vec{cp} , but not \vec{bp} , which would be the correct result!

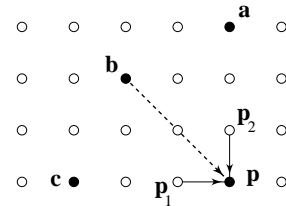


Figure 2: A typical configuration producing errors in Danielsson's algorithm.

For a closer analysis of the situation, we need to recall the concept of *influence zone*. Given a compact set $X \subset \mathbb{R}^2$ made of n connected components X_1, X_2, \dots, X_n , the influence zone $iz(X_i)$ of X_i is the set of the points of the plane which are closer to X_i than to any other component of X :

$$iz(X_i) = \{p \in \mathbb{R}^2 \mid \forall j \neq i, d(p, X_i) \leq d(p, X_j)\}. \quad (3)$$

Here, the typical difficulty is that, when restricted to the discrete plane equipped with grid G , the "continuous" influence zone of a connected component *may well be disconnected!* As illustrated by Fig. 3, this is the very reason why Danielsson's algorithm sometimes yields inexact results. In addition, when the influence zone exhibits sharp angles, the disconnection can be arbitrarily important. This remark is valid for any kind of discrete grid. To get exact Euclidean distance functions, it is therefore essential to introduce algorithms

working on a *non-local* basis and capable of propagating information over such disconnections.

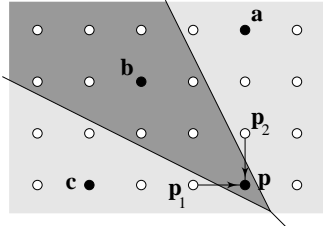


Figure 3: The errors produced by Danielsson’s algorithm come from the fact that some Euclidean influence zones are not connected with respect to the digital grid being used (according to the 4-connected square grid used here, p is disconnected from the rest of $\text{iz}(\{b\})$).

4 Proposed Algorithm

In this section, only hexagonal grids are considered, but the method easily extends to square grids. We shall get rid of the trick consisting in scanning each line in both directions and make use of a more relevant type of image scanning, based on the encoding of the object’s contours as chains and their propagation in the image. These types of algorithms were introduced by Schmitt in 1988 [9]. They are based on the fact that, given a chain C coding the boundary of a component X (Freeman chain [3]), one can determine the chain corresponding to the dilated set $\delta(X)$ by simply processing C via rewriting rules. Together with the algorithms based on queues of pixels [12], they provide one of the most efficient implementations of most morphological transformations. They work in 4-, 6- or 8-connectivity, and like the sequential methods described in [1], they can be generalized to yield better approximations of Euclidean distances, like octagonal and dodecagonal distances. However, to go further and to produce actual Euclidean distances, we need to attach to each chain element a vector information, which is propagated in the image together with the chains. In this sense, the algorithm proposed below is an extension of both Schmitt’s and Danielsson’s ideas.

More precisely, suppose that chain C_k has been obtained after k dilations of the initial chain C_0 . Each of the points p of C_k is then associated a vector $\vec{v}(p)$ such that $p - \vec{v}(p)$ is the pixel (one of the pixels) of C_0 which is the closest to p with respect to the Euclidean distance. In this section, we thus manipulate chains whose

elementary components are no longer chain directions, but pairs direction-vector that we shall call *spins*: A chain C is a data structure made of an *origin pixel* Or_C , a *length* $l(C)$, and an array $\text{Dir}_C = [s_0, s_2, \dots, s_{l(C)-1}]$ of *spins*, i.e., of pairs direction-vector

$$\forall i \in [0, l(C) - 1], s_i = \text{Dir}_C[i] = (d_i, \vec{v}_i).$$

Given a pixel p belonging to a chain C , $\vec{v}_C(p)$ refers to the vector associated by C to p . In practice, these vectors are manipulated via their integer coordinates in the system (\vec{u}_0, \vec{u}_1) . The square of the norm of a vector $\vec{w} = k_0\vec{u}_0 + k_1\vec{u}_1$ is given by $\|\vec{w}\|^2 = k_0^2 + k_1^2 + k_0k_1$.

Schmitt’s rewriting rules are now adapted to these new data structures in such a way that the vector information is also rewritten and propagated in the image. Here, there is however an additional difficulty: a pixel p of a chain C_k obtained after k dilations of a chain C_0 may well be equally distant (with respect to the Euclidean distance) to several pixels p_i of C_0 , as illustrated by Fig. 4. Assigning to p one of the vectors $\vec{p}_i\vec{p}$ could well yield important errors in the vectors determined at subsequent steps. One could thus associate with p all the vectors $\vec{p}_i\vec{p}$ and propagate them, but this would put a considerable burden on the algorithm. A much better solution is to account for the fact that such difficulties only occur for *concavities* of the original set. Hence, the initial loops (i.e., closed chains) are simply decomposed into several convex chains, as illustrated by Fig. 5.

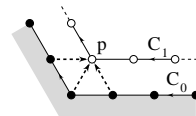


Figure 4: After dilation of a non-convex chain, which vector shall be assigned to pixel p ?

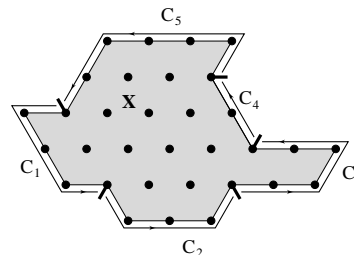


Figure 5: Decomposition of the boundary of a set X into convex chains C_i .

This decomposition is easily incorporated into the original contour tracking required by the algorithm. Its consistency is assured by the fact that, by dilation, a convex chain is transformed into a convex chain. When the boundaries of the initial image exhibit linear parts whose general orientation is not one of the grid directions, a large number of small chains is produced. This is not disturbing in practice, since the total number of spins remains unchanged. Now, even with this convex decomposition, a pixel p of a dilated chain may still be equally distant to two pixels p_1 and p_2 of the original chain [13]. However, in this case, the rewriting rules presented in Fig. 6 are designed in such a way that any of the two vectors $\vec{p}_1\vec{p}$ and $\vec{p}_2\vec{p}$ can be chosen without inducing errors in the ulterior propagation steps.

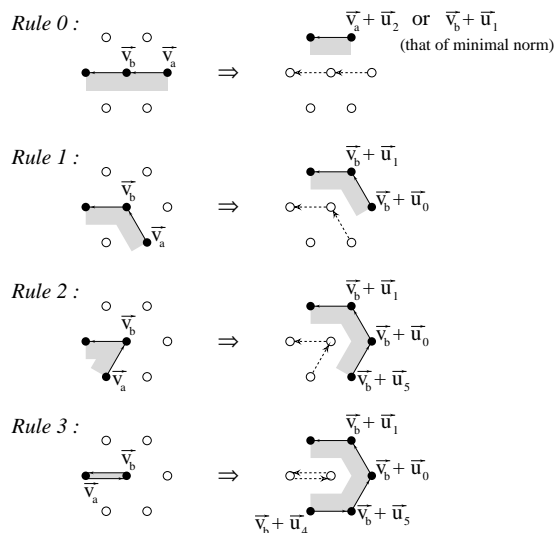


Figure 6: Rewriting rules for the vector conveying convex chains. Note that rules 2 and 3 only play a role in the first dilation step.

At this point, given a convex chain C , we know how to propagate it in such a way that it conveys Euclidean distance. Now, after each dilation step, the vector information has to be written in a vector image \vec{J} , similarly as in § 3. In the same time, so as not to dramatically increase the computation times, the chain parts which do not need to be propagated further are *eliminated*. These two operations are referred to as *adjustment* of the chains. The simplest adjustment technique one may think of consists in writing the Euclidean distance conveyed by the current chain whenever it is smaller than that already written in the image, while eliminating the remaining chain parts [13]. More precisely, denoting by C the current chain and p its current

pixel:

1. If $\|\vec{v}_C(p)\|^2 < \|\vec{J}(p)\|^2$ then $\vec{J}(p) \leftarrow \vec{v}_C(p)$;
2. otherwise do not modify $\vec{J}(p)$ and cut chain C in p (eliminate the part of C containing p).

Note that after each adjustment, a chain may be cut into several smaller ones... With the above rules, a chain part is eliminated as soon as the distance it conveys is larger than that already written in the image. The algorithm thus falls into the same traps as Danielsson's: when the influence zone of a component is not connected with respect to the hexagonal grid, the chains are not able to go over the disconnection, as illustrated by Fig. 7.

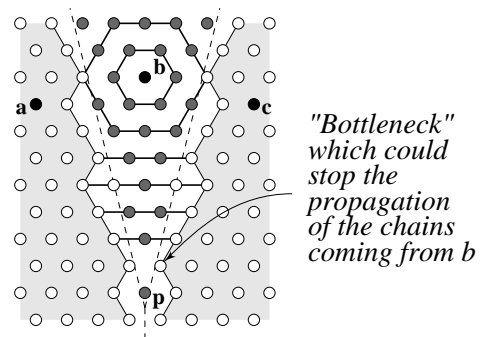


Figure 7: With an incorrect adjustment method, the chain propagation can be stopped too early and similar errors as with Danielsson's algorithm may occur.

Here, since the chains are propagated independently of any underlying vector image, the problem can now be solved as follows: let dist_I be the (exact) Euclidean distance function associated with I and let p and q be two neighboring pixels. It is easy to see that the following Lipschitzian property is fulfilled:

$$|\text{dist}_I(p) - \text{dist}_I(q)| \leq 1.$$

This holds even when p and q are located in different influence zones of I . Hence, in order for a chain to be able to jump over an influence zone "bottleneck", it suffices to carry on propagating it as long as the vector $\vec{v}_C(p)$ satisfies:

$$\|\vec{v}_C(p)\| \leq \|\vec{J}(p)\| + 1. \quad (4)$$

To summarize, we finally adopt the following adjustment rules, which allow the chains to keep being propagated along the boundaries of the Euclidean influence zones (C stands for the current chain and p for its current pixel):

1. If $\|\vec{v}_C(p)\| < \|\vec{J}(p)\|$ then $\vec{J}(p) \leftarrow \vec{v}_C(p)$;
2. otherwise, if $\|\vec{v}_C(p)\| \leq \|\vec{J}(p)\| + 1$ then, $\vec{J}(p)$ is unchanged, but **the chain part containing p is not eliminated**;
3. otherwise, leave $\vec{J}(p)$ unchanged and eliminate the part of C containing p .

According to the above adjustment rules and to the propagation (rewriting) rules of Fig. 6, we finally come up with the following algorithm, where \vec{o} stands for the null vector and \vec{v}_∞ represents a vector of “infinite” norm:

Algorithm: Euclidean distance function

- input: I , binary image,
output: \vec{J} , vector image defined on D_I ;
- For every pixel $p \in D_I$, {
 If $I(p) = 1$ then $\vec{J}(p) \leftarrow \vec{o}$;
 otherwise $\vec{J}(p) \leftarrow \vec{v}_\infty$;
}
- Assign value \vec{o} to the frame of \vec{J} ;
- Track the contours of I and decompose them into convex parts;
 Initialize the vectors $\vec{v}_C(p)$ of all chains with \vec{o} ;
- Repeat until there remain chains {
 Dilate (i.e. rewrite) the chains;
 Adjust them in image \vec{J} ;
}

This algorithm yields *exact* Euclidean distance functions in approximately half the time required by Danielsson’s algorithm. Its average execution time is of 2 seconds on a *Sun Sparc Station 1*, for a 256×256 image. Its average efficiency would probably be extremely difficult to estimate thoroughly, and give little useful information¹. We may only note that the chains are often written above one another, so that some image parts may be scanned several times. However in practice, the number of pixels scanned more than twice turns out to be rather low. In any case, on conventional computers, the algorithm introduced in this paper is much more efficient than the only other known algorithm yielding exact Euclidean distance functions [15], which requires a very large number of complete image scannings.

¹For example, most morphological algorithms run in linear time with respect to the number of pixels, but some of them take one second whereas some others may require one hour!

5 Examples, Application to Skeletons

On Fig. 8, the successive level lines of a Euclidean distance function produced by the present algorithm are displayed and opposed to a hexagonal distance function. The difference speaks for itself! Isotropic dilations and erosions (and consequently, openings and closings) [10]) can then be obtained via simple thresholdings. For these operations however, some even more efficient algorithms can be designed, which are described in [14]. Here, it is especially interesting to use our distance function algorithm to determine Euclidean skeletons. The *skeleton* or *medial axis* of a set can indeed be obtained as the set of crest-lines of the distance function. When the distance function has been obtained via the algorithm of this paper, this results in beautiful well-drawn skeletons [6]. In addition, Euclidean skeleton by influence zones (SKIZ) [5] can be derived. The best method consists in labelling the connected components of the binary image I under study and propagating these labels together with the chains in our algorithm. Extracting the boundaries of the resulting labelled zones yields the Euclidean SKIZ (see Fig. 9.a).

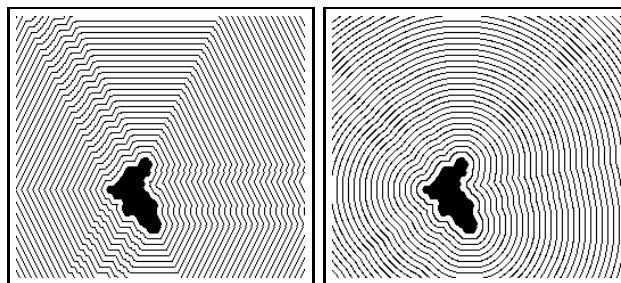


Figure 8: Hexagonal versus Euclidean distance function.

6 Concluding Remarks

An efficient and accurate algorithm for Euclidean distance function computation has been introduced. It was described for hexagonal grids, but is perfectly amenable to square ones. The original chain propagation technique on which it relies allows one to obtain Euclidean distances in computation times which are comparable to those needed to determine, e.g., square, hexagonal or octagonal distance functions with classical methods. The algorithm requires a random access to

the pixels and is therefore particularly suited to general purpose computers. However, it seems possible to parallelize the propagation of the different chains involved in the process. The resulting procedures would then be extremely fast on some architectures like massively parallel computers.

The interest of this algorithm to produce Euclidean skeletons and SKIZ has been mentioned. Additionally, it can serve as a basis to determine neighborhood graphs like Delaunay Triangulations, Gabriel Graphs [4] and Relative Neighborhood Graphs [11] in digital pictures. These graphs are known to provide interesting tools for clustering and spatial repartition analyses. Originally, they were defined for finite sets of points in the plane, but their definitions extend to finite sets of connected components [12, chap. 5]. Up to now, most algorithms to determine them were the concern of computational geometry techniques and only worked when the initial components were isolated points. Alternatively, in the present discrete case, algorithms using a contour tracking of the labelled Euclidean influence zones together with information taken from the vector image \vec{J} have been developed (see Fig. 9.b). They work for connected components of arbitrary size and shape and will constitute the topic of further publications.

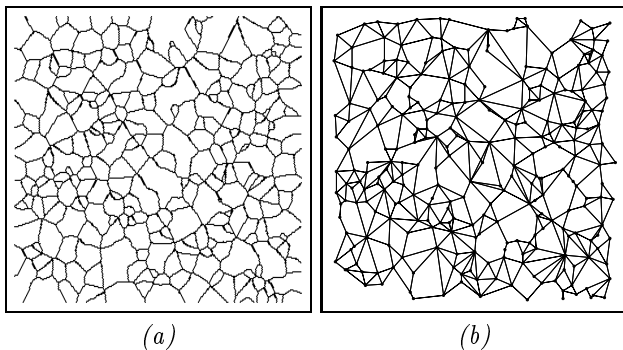


Figure 9: Euclidean SKIZ and the corresponding Gabriel Graph in a binary image.

References

[1] G. Borgefors. Distance transformations in digital images. *Comp. Vis., Graphics and Image Processing*, 34:334–371, 1986.

[2] P. Danielsson. Euclidean distance mapping. *Comp. Graphics and Image Processing*, 14:227–248, 1980.

[3] H. Freeman. On the encoding of arbitrary geometric configurations. *IEEE Transactions on Computers*, C10:260–268, 1961.

[4] K. Gabriel and R. Sokal. A new statistical approach to geographic variations analysis. *Systematic Zoology*, 18:259–278, 1969.

[5] C. Lantuéjoul. Issues of digital image processing. In R. M. Haralick and J.-C. Simon, editors, *Skeletonization in Quantitative Metallography*. Sijthoff and Noordhoff, Groningen, The Netherlands, 1980.

[6] F. Meyer. Digital Euclidean skeletons. In *SPIE Vol. 1360, Visual Communications and Image Processing*, pages 251–262, Lausanne, Switzerland, Oct. 1990.

[7] A. Rosenfeld and J. Pfaltz. Sequential operations in digital picture processing. *J. Assoc. Comp. Mach.*, 13(4):471–494, 1966.

[8] A. Rosenfeld and J. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[9] M. Schmitt. *Des Algorithmes Morphologiques à l'Intelligence Artificielle*. PhD thesis, Ecole des Mines, Paris, Feb. 1989.

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

[11] G. T. Toussaint. The Relative Neighborhood Graph of a finite planar set. *Pattern Recognition*, 12:1324–1347, 1980.

[12] L. Vincent. *Algorithmes Morphologiques à Base de Files d'Attente et de Lacets: Extension aux Graphes*. PhD thesis, Ecole des Mines, Paris, May 1990.

[13] L. Vincent. Exact euclidean distance function by chain propagations. Technical Report 91–4, Harvard Robotics Laboratory, 1991.

[14] L. Vincent. Morphological transformations of binary images with arbitrary structuring elements. *Signal Processing*, 22(1):3–23, Jan. 1991.

[15] H. Yamada. Complete euclidean distance transformation by parallel operations. In *7th International Conference on Pattern Recognition*, pages 69–71, Montreal, 1984.