OPTICAL COMPUTER VISION

(Some geometrical algorithms)

Y. B. KARASIK
School of Mathematical Sciences
Tel Aviv University
Tel Aviv, Israel

Abstract

We address one of the basic problems of computer vision – extracting vertices from a polygonal image. A constant time optical algorithm to solve the problem is proposed.

1 Introduction

Computer Vision aims to simulate the human visual system and as such deals with problems ranging from image acquisition and "interesting points" extraction to pattern recognition and scene analysis. The standard "interesting points" are corners or vertices of polygons. Therefore, there exist numerous algorithms for corner detection/ vertex extraction intended for standard electronic computers [KR], [FH], [RSB], etc.

However, even if an eye can be likened to a computer, it should be likened to an optical computer because an eye is an optical system including pupil, crystalline lense, photodetectors (in the form of rods and cones of a retina), etc. In addition, an eye performs some transformations inherent to optical computers [Sa]: the Fourier transform of an input image [G], log-polar change of coordinates [S] which turns out to be the corner stone of many optical algorithms (see e.g. [K91, KS92]), etc.

Therefore, it would be prudent to initiate developing algorithms of computer vision for optical computers. The first step in this direction could be optical algorithms for extracting edges and vertices from an image. In fact, we only need the latter algorithm because extracting edges (i.e. the boundary of an image) can be performed either by spatial differentiation of an image or by convolving an image with an edge detecting operator (i.e. Sobel operator). And both these operations – differentiation and convolution, are elementary operations in the optical computational model (see e.g. [KS92]).

Recall that this model includes as primitives the following operations:

(i) Minkowski sum and difference of two plane figures [K91];

(ii) Union, intersection and subtraction of plane figures [K91];

(iii) The standard duality transform that maps a point (a, b) to a line $y = -a \cdot x + b$ and a line $y = c \cdot x + d$ to a point (c, d) [K92, KS92]. It is well known that this duality preserves incidence between points and lines, and maps a point lying above (resp. below) a line l to a line lying above (resp. below) a dual point of l.

(iv) Hough duality transform that maps a point (p, θ) to a straight line $x \cdot \cos\theta + y \cdot \sin\theta = p$, and a straight line segment

$$x \cdot \cos\theta + y \cdot \sin\theta = p, \quad \theta \in [\theta_1, \theta_2]$$

to a point (p, θ) whose intensity is proportional to the length of a segment [ED];

(v) Semi-conformal changes of coordinate [B]:

$$\begin{cases} u = u(x, y); \\ v = v(x, y), \end{cases}$$

satisfying the condition

$$\frac{\partial u}{\partial y} = \frac{\partial v}{\partial x}.$$

(vi) Intensity inversion of an image: $g(x,y) = I_{max} - f(x,y)$, where $f(x,y)$ and $g(x,y)$ are intensity distributions of an image before and after inversion respectively, and $I_{max} \geq f(x,y)$ for all x, y [OHG];

(vii) Thresholding of an image at a given level of intensity (as well as extracting regions of its maximum/minimum intensity) [GCHHZY].

Remark: In what follows we use two types of thresholding: $Threshold_{\leq a}(S)$ which denotes that portion of an image S whose intensity is not greater than a, and $Threshold_a(S)$ which denotes that slice of S whose intensity is a exactly. Obviously,

$$Threshold_a(S) = Threshold_{\leq a}[I_{max} - Threshold_{\leq I_{max}-a}(I_{max} - Threshold_{\leq a}(S))].$$

Based on these operations as primitives, in the rest of the paper we propose a constant time optical algorithm to extract vertices from the boundary of a polygon.

2 How to extract vertices optically from the boundary of a polygon

Let the boundary ∂A of a polygonal image A be given. We assume that this boundary consists of rectilinear segments $\{S_i\}_{i=1}^N$ called edges. To extract vertices of the polygon from the boundary we can proceed as follows:

Step 1. Dualize the edges $\{S_i\}_{i=1}^N$ into points and then dualize these points back into the union of straight lines $L = \bigcup_{i=1}^N l_i$ supporting these edges (see Fig. 1).

Step 2. Extract the intersection points $Q = \{Q_i\}_{i=1}^M$ of the lines by thresholding the image obtained at level 2. Obviously, Q contains all vertices of A and intersection points of its boundary with lines in L and between pairs of lines in L. These two kinds of points can be considered as redundant vertices. To get rid of them we can proceed as follows:

Step 3. Compute $P = \partial A \cap Q$. Obviously, P contains all vertices of A and intersection points of its boundary with lines in L.

Step 4. Choose an $\epsilon > 0$ such that the disc of radius ϵ centered at any point $p \in P$ does not contain other points of P and does not intersect edges of the polygon except for the edge(s) that contain p (the details of performing this step are described below).

Step 5. Compute the following segments of boundary edges:

$$S = \partial A \cap (P + O_\epsilon),$$

where O_ϵ is the disc of radius ϵ centered at the origin of the coordinates (see Fig. 2).

It is easily seen that each segment $s \in S$ has length ϵ or 2ϵ, depending on whether or not one of its endpoints is a vertex of A.

Step 6. Construct all radial segments R of O_ϵ which are parallel to the edges of the polygon using the algorithm described in [KS93].

Step 7. Compute

$$Threshold_\infty(S + R).$$

It is easily seen that the image obtained consists of segments lying completely in the boundary of the polygon unless they pass through its vertices (see Fig 3). Hence,

$$Threshold_\infty(S + R) \quad \setminus \quad \partial A$$

is a set of exterior open segments, each of which has a vertex of A as an endpoint and lies on an extension of an incident edges. We refer to these segments as "pointers" at the vertices, and construct them by:

Step 8.

$$pointers\_at\_vertices =$$

$$Threshold_\infty(S + R) \quad \setminus \quad \partial A,$$

(see Fig. 4).

Step 9. Elongate the pointers by computing

$$lengthened\_pointers =$$

$Threshold_\infty(pointers\_at\_vertices+R)$,

(see Fig. 5).

Obviously, due to this choice of ϵ, *lengthened_pointers* contain all vertices of A and do not contain redundant vertices. Hence, we can complete the algorithm as follows:

Step 10. Compute

$$Vertices = P \cap lengthened\_pointers.$$

Computing the required ϵ at Step 4 can be performed as follows:

Step 1. Rotate the set L of lines obtained in Step 1 of the previous algorithm by the angle $\frac{\pi}{2}$. This can be done using the following semi-conformal coordinate transformation:

$$\begin{cases} u = y \\ v = -x. \end{cases}$$

As a result we obtain a set L_1 of lines, each of which is perpendicular to a corresponding line of L.

Step 2. Hough-dualize the lines of L_1 into points (p_i, θ_i), then transform these points into the corresponding points $(0, \theta_i)$, and finally Hough-dualize the new points back to set L_2 of lines which pass through the origin of the coordinates. Obviously, each line of L is perpendicular to at least one line of L_2 and vice versa.

Step 3. Compute the Minkowski sum $L_2 + P$. As a result we obtain the set of supporting lines of all the perpendiculars from the points of P to the edges $\{S_i\}_{i=1}^N$ of the polygon.

Step 4. Compute the set Q of all the intersection points between the edges of A and the perpendiculars to these edges from the points of P, constructed at the previous step.

Step 5. Compute a lower bound 2ϵ on the minimum distance between any pair of the points in $P \cup Q$. Obviously, such ϵ satisfies the requirements of Step 4 of the previous algorithm.

Hence, we obtain

Theorem 2.1 *The vertices of a polygonal image can be extracted optically in constant time by a single optical processor which works in the monochromatic mode of optical computation.*

3 Conclusion

What is the importance of the result obtained?

The point is that till now all evidence that an eye (and probably the brain as a whole) is an optical computer was mostly neuro-physiological [P] and not behaviorial. For example, nobody tried to explain why human visual perception solves geometrical problems, ranging from extracting feature points of images to motion planning amidst obstacles so efficiently with a speed which does not seem to depend on the complexity of the images viewed, whereas the running time of the most efficient algorithms for electronic computers to solve these problems is some polynomial of the input size.

Our result explains this phenomenon and, hence, can be viewed as new and important evidence in favour of the conjecture that an eye is an optical computer.

References

[B] O. Bryngdahl, Optical map transformations, *Optics Communication*, 10(2), 164-168 (1974).

[ED] G. Eichman and B. Z. Dong, Coherent optical production of the Hough Transform, *Applied Optics* 22(6), 830-834 (1983).

[FH] J. Q. Fang, T. S. Huang, A corner finding algorithm for image analysis and registration, *Proceedings of AAAI Conference*, pp. 46–49, 1982.

[G] N. Graham, The visual system does a crude Fourier analysis of patterns, *SIAM–AMS Proceedings*, vol. 13, pp. 1–16, 1981.

[GCHHZY] C. Gu, S. Campbell, J. Hong, Q. He, D. Zhang, P. Yeh, Optical thresholding and maximum operations, *Applied Optics*, 31(26), pp. 5661–5665, 1992.

[K91] Y. B. Karasik, Optical algorithms of computational geometry, Technical report

199/91, The Eskenasy Institute of Computer Science, Tel Aviv University (1991).

[K92] Y. B. Karasik, The optical algorithm for the inverse Hough transform, *Journal of Optical Computing and Processing,* 2(2), 127–136 (1992).

[KS92] Y. B. Karasik, M. Sharir, Optical Computational Geometry, *Proc. 8th ACM Symposium on Computational Geometry,* ACM Press, pp. 232–241, 1992.

[KS93] Y. B. Karasik, M. Sharir, The power of geometric duality and Minkowski sums in optical computational geometry, *Proceedings 9th ACM Symposium on Computational Geometry,* ACM Press, pp. 379 – 388, 1993.

[KR] L. Kitchen, A. Rosenfeld, Gray level corner detection, *Pattern Recognition Letters,* 1, pp. 95–102, 1982.

[OHG] E. Ochoa, L. Hesselink, J. Goodman, Real-time intensity inversion using two-wave and four-wave mixing in photorefractive $Bi_{12}SiO_{20}$, *Applied Optics,* 24(12), 1826 – 1832 (1985).

[P] K. H. Pribram, Languages of the brain (Prentice-Hall inc., Englewood cliffs, New Jersey, 1971).

[RSB] K. Rangarajan, M. Shah, D. V. Brackle, Optimal corner detector, *Proceedings of the 2nd International Conference on Computer Vision,* pp. 90–94, 1988.

[Sa] B. E. A. Saleh, Optical information processing and the human visual system, *in Applications of Optical Fourier Transforms, ed. H. Stark,* pp. 431–463, 1982.

[S] H. Szu, Holographic coordinate transformations and optical computing, *in Optical and Hybrid Computing,* Proceedings SPIE, vol. 634, pp. 480–484, 1986.

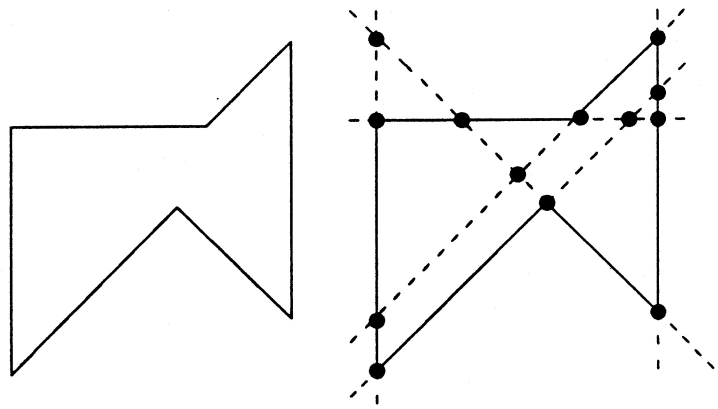

Figure 1: A polygon, extensions of its edges and the intersection points of these extensions

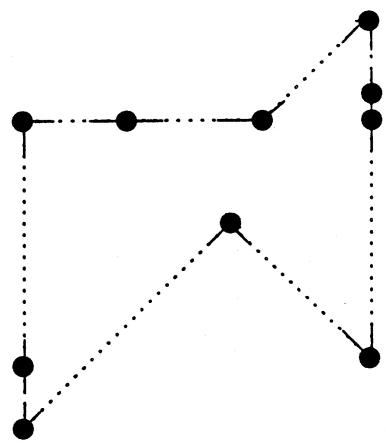

Figure 2: Real and spurious vertices of the polygon, and the segments $S = \partial A \cap (P + O_\epsilon)$

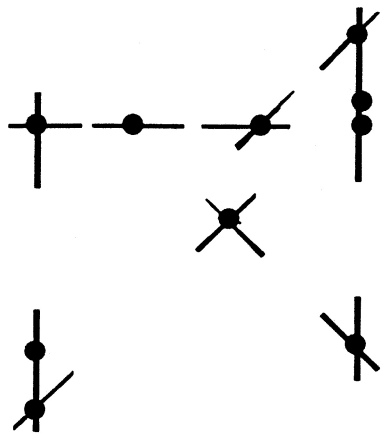

Figure 3: The image $Threshold_{\infty}(S + R)$

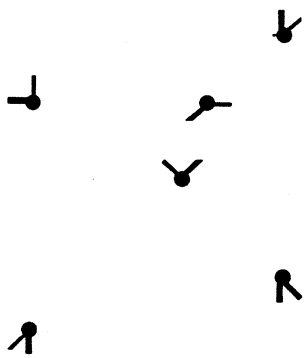

Figure 4: Vertices and pointers at them

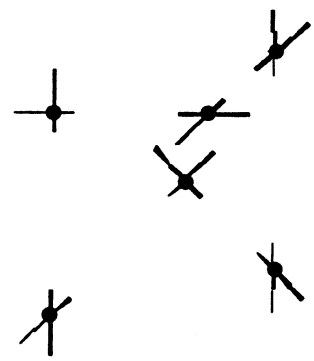

Figure 5: Vertices and lengthened pointers at them