

M. J. ATALLAH

C. C. RIBEIRO

S. LIFSCHITZ

**A linear time algorithm for the computation of some  
distance functions between convex polygons**

*Revue française d'automatique, d'informatique et de recherche  
opérationnelle. Recherche opérationnelle*, tome 25, n° 4 (1991),  
p. 413-424.

[http://www.numdam.org/item?id=RO\\_1991\\_\\_25\\_4\\_413\\_0](http://www.numdam.org/item?id=RO_1991__25_4_413_0)

© AFCET, 1991, tous droits réservés.

L'accès aux archives de la revue « Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/legal.php>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme  
Numérisation de documents anciens mathématiques  
<http://www.numdam.org/>

## A LINEAR TIME ALGORITHM FOR THE COMPUTATION OF SOME DISTANCE FUNCTIONS BETWEEN CONVEX POLYGONS (\*)

by M. J. ATALLAH <sup>(1)</sup>, C. C. RIBEIRO <sup>(2)</sup> and S. LIFSCHITZ <sup>(3)</sup>

---

Abstract. — *We present in this paper a linear time algorithm for the computation of some distance functions between convex polygons in  $\mathbb{R}^2$  in the general case where they can intersect. The interest for this problem comes from applications in robot vision, pattern recognition and contour fitting, see Cox, Maitre, Minoux and Ribeiro [1989]. The algorithm is based on previous work of Atallah [1983] for the computation of the Hausdorff distance between disjoint convex polygons. Computational results are presented.*

Keywords : Computational geometry; convex polygons; distance computation; contour fitting; robot vision; algorithms.

Résumé. — *Nous présentons dans ce papier un algorithme de complexité linéaire pour le calcul de quelques fonctions de distance entre polygones convexes dans  $\mathbb{R}^2$ , dans le cas général où leur intersection est non-vide. L'intérêt pour ce problème provient d'applications en vision par ordinateur, reconnaissance des formes et recalage de contours (voir Cox, Maitre, Minoux et Ribeiro [1989]). L'algorithme est basé sur le travail d'Atallah [1983] pour le calcul de la distance de Hausdorff entre polygones convexes disjoints. Des résultats numériques sont présentés.*

Mots clés : Géométrie algorithmique; polygones convexes; calcul de distances; recalage de contours; vision par ordinateur; algorithmes.

### 1. INTRODUCTION

The computation of the projection of a point in  $\mathbb{R}^2$  on a convex polygon is a basic problem in computational geometry. An extension of this problem consists in the computation of the distance between two convex polygons,

---

(\*) Received August 1990.

<sup>(1)</sup> Purdue University, Department of Computer Science, West Lafayette, IN 47907, U.S.A..

<sup>(2)</sup> Catholic University of Rio de Janeiro, Department of Computer Science, Rua Marquês de São Vicente 225, Rio de Janeiro 22453, Brazil. Work of this author was partially done during his sabbatical leave at GERAD and *École Polytechnique de Montréal*, and was partially supported by *Conselho Nacional de Desenvolvimento Científico e Tecnológico* under grant 202005/89.5.

<sup>(3)</sup> Catholic University of Rio de Janeiro, Department of Electrical Engineering, Caixa Postal 38063, Rio de Janeiro 22452, Brazil.

which is based on the computation of the projection of each vertex of each polygon on the other one.

An application of this problem is described in Cox, Maitre, Minoux and Ribeiro [1989], consisting in the determination of the optimal matching of two given convex polygons (not necessarily having the same number of vertices, and with a non-empty intersection in the general case) representing, for instance, two distinct series of observations of the same contour or planar object. There it is shown that finding the projection of any vertex of one of the polygons on the other is the basic subproblem to be solved at each step of the algorithm which obtains their optimal matching.

In that reference, the authors use a trivial  $O(n_1 n_2)$  algorithm for the computation of the distance between the two convex polygons, where  $n_1$  and  $n_2$  are their number of vertices. As logarithmic time algorithms are available for the computation of the projection of each vertex (see e.g. Edelsbrunner [1985]), they can be used for building an  $O(n_1 \log n_2 + n_2 \log n_1)$  algorithm for the computation of the same distance.

We present in this paper a linear time algorithm running in time  $O(n_1 + n_2)$  for the computation of some distance functions between convex polygons, among them the Hausdorff distance and that defined by Cox, Maitre, Minoux and Ribeiro [1989]. The algorithm is based on previous work of Atallah [1983] for the computation of the Hausdorff distance in the case where the two polygons do not intersect.

The paper is organized as follows. We present in Section 2 the problem statement, along with the basic notation. The algorithm of Atallah for disjoint polygons is studied in Section 3 with the necessary modifications to compute the distance defined by Cox, Maitre, Minoux and Ribeiro [1989]. Computational results are also presented, illustrating how better this algorithm behaves in practice, when compared to the trivial  $O(n_1 n_2)$  one. Following, we give in Section 4 the detailed description of our distance computation algorithm for the general case where the polygons can intersect. Finally, computational results and conclusions are presented in Section 5.

## 2. PROBLEM STATEMENT

The two convex polygons  $P_1$  and  $P_2$  are given by the following data:

- (a)  $n_1$  (resp.  $n_2$ ), the number of vertices of polygon  $P_1$  (resp.  $P_2$ );
  - (b) the coordinates  $(a_j^1, b_j^1)$  (resp.  $(a_j^2, b_j^2)$ ) of the  $j$ -th vertex  $v_j^1$  (resp.  $v_j^2$ ) of  $P_1$  (resp.  $P_2$ ), for any  $j \in \{0, 1, \dots, n_1 - 1\}$  (resp.  $j \in \{0, 1, \dots, n_2 - 1\}$ );
- and

(c) the numbering of the vertices of  $P_1$  (resp.  $P_2$ ) is such that for any  $j \in \{0, 1, \dots, n_1 - 1\}$  (resp.  $j \in \{0, 1, \dots, n_2 - 1\}$ ), then the segment joining vertices  $v_j^1$  and  $v_{j+1 \bmod n_1}^1$  (resp.  $v_j^2$  and  $v_{j+1 \bmod n_2}^2$ ) is the  $j$ -th edge of polygon  $P_1$  (resp.  $P_2$ ).

According to (c) above, suppose that the vertices of both  $P_1$  and  $P_2$  are numbered in counterclockwise cyclic order. For any point with coordinates  $(u, v)$ , we define its distance  $\delta_{P_l}(u, v)$  to polygon  $P_l$  ( $l = 1, 2$ ) as

$$\delta_{P_l}(u, v) = \begin{cases} \|(u, v) - Proj_{P_l}(u, v)\|, & \text{if } (u, v) \notin Interior(P_l), \\ 0, & \text{otherwise,} \end{cases}$$

where  $Proj_{P_l}(u, v)$  denotes the point of  $P_l$  achieving the minimum euclidean distance to  $(u, v)$ ,  $Interior(X)$  denotes the interior of polygon  $X$  (i.e.,  $X$  minus its boundary), and  $\|\cdot\|$  is the  $L_2$  norm.

With this notation, the distance defined by Cox, Maitre, Minoux and Ribeiro [1989] can be expressed as

$$D_{CMMR}(P_1, P_2) = D_{CMMR}(P_2, P_1) = \sum_{j=0}^{n_1-1} (\delta_{P_2}(a_j^1, b_j^1))^2 + \sum_{j=0}^{n_2-1} (\delta_{P_1}(a_j^2, b_j^2))^2,$$

i.e., the sum of the squares of the euclidean distances from each vertex of each polygon to its projection on the other polygon. The Hausdorff distance (see Grunbaum [1967]) considered in Atallah [1983] can also be expressed with this notation, as

$$\begin{aligned} D_H(P_1, P_2) = D_H(P_2, P_1) &= \max_{l=1, 2} \left\{ \max_{j=0, 1, \dots, n_l-1} \delta_{P_{3-l}}(a_j^l, b_j^l) \right\} \\ &= \max \left\{ \max_{j=0, 1, \dots, n_1-1} \delta_{P_2}(a_j^1, b_j^1), \max_{j=0, 1, \dots, n_2-1} \delta_{P_1}(a_j^2, b_j^2) \right\}, \end{aligned}$$

i.e., the maximum euclidean distance from any vertex of any polygon to its projection on the other one.

We consider in this paper the problem of the fast computation of  $D_{CMMR}(P_1, P_2)$  by a linear time algorithm. The algorithm discussed in Section 3 below for the case of disjoint polygons is an adaptation of that presented in Atallah [1983] and can be easily extended to other distance functions. It will also be useful in the development of the algorithm described in Section 4 for the general case of non-disjoint polygons.

3. DISJOINT POLYGONS

Throughout this section we assume that  $P_1$  and  $P_2$  are disjoint (*i. e.*, they do not intersect and none of them contains the other). A supporting line  $r$  of a convex polygon  $P$  is a straight line passing through one of its vertices and such that the interior of  $P$  lies entirely on one side of  $r$ . The following lemma holds:

LEMMA 1 (Atallah [1983]): Consider a vertex  $x=(u, v)$  of  $P_2$  and its projection  $y=Proj_{P_1}(u, v)$  on  $P_1$ . Then, the perpendicular to  $xy$  at  $y$  is a supporting line  $r$  of  $P_1$ , and  $x$  and  $P_1$  are on different sides relative to that line (see fig. 1).

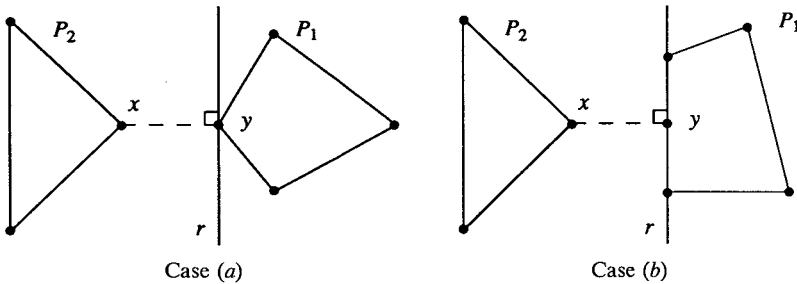


Figure 1. - Illustration of Lemma 1.

We address now the computation of the first part of  $D_{CMRR}(P_1, P_2)$ , *i. e.*, the sum of the squares of the euclidean distances from each vertex  $v_j^1$  of  $P_1$  to its projection on  $P_2$ . Let  $d_j = \delta_{P_2}(a_j^1, b_j^1)$  be the distance from vertex  $v_j^1$  to  $P_2$ . Computing every  $d_j$  separately may take  $O(n_2)$  for each one, resulting in a total of  $O(n_1 n_2)$  time. We show in the following how to compute all the  $d_j$ 's (and hence  $D_{CMRR}$ ) in  $O(n_1 + n_2)$  time.

For every  $j$ , let  $y_j$  be the point of  $P_2$  for which  $d_j = \|v_j^1 - y_j\|$ , *i. e.*,  $y_j = Proj_{P_2}(a_j^1, b_j^1)$ . Note that  $y_j$  is either a vertex of  $P_2$  or the foot of the perpendicular from  $v_j^1$  to one of the edges of  $P_2$ . Let  $\sigma$  denote the sequence  $y_0, y_1, \dots, y_{n_1-1}, y_0$ . We say that  $\sigma$  is *moving counterclockwise* at  $y_j$ , if  $v_{j+1}^1 \bmod n_1$  is to the left of  $y_j v_j^1$ . If  $v_{j+1}^1 \bmod n_1$  is to the right of  $y_j v_j^1$ , then we say that  $\sigma$  is *moving clockwise* at  $y_j$ . If  $v_{j+1}^1 \bmod n_1$  belongs to the line  $y_j v_j$ , then we say that  $\sigma$  is *stationary* at  $y_j$ .

LEMMA 2:  $v_{j+1}^1$  and  $y_{j+1}$  are on the same side relatively to the line  $y_j v_j^1$ .

We say that  $\sigma$  *changes direction* at  $y_j$  (i) if it is moving clockwise at  $y_{j-1}$  and it is moving counterclockwise at  $y_j$  or (ii) if it is stationary or moving counterclockwise at  $y_{j-1}$  and it is moving clockwise at  $y_j$ . Then, it follows

that:

LEMMA 3 (Atallah [1983]):  $\sigma$  changes direction at most twice.

Lemma 3 above ensures the linear time complexity of the algorithm described next. The first part of the algorithm is given below and consists in the computation of the sum of the squares of the distances from each vertex of  $P_1$  to its projection on  $P_2$ . Its second part is basically the same and is omitted here, consisting in the computation of the sum of the squares of the distances from each vertex of  $P_2$  to its projection on  $P_1$ . The algorithm starts by the computation of  $y_0 = Proj_{P_2}(a_0^1, b_0^1)$ , which can be easily done in  $O(n_2)$  time by a trivial algorithm. Following, Lemma 3 is used to compute  $y_{k+1}$  from  $y_k$ , for all  $k=0, 1, \dots, n_1-2$  in  $O(n_1+n_2)$  time.

Step 1 (finding  $y_0$ ):

Computed  $d_0^2$  and find  $y_0$  by a trivial linear time algorithm, set  $D_{CMMR}(P_1, P_2) \leftarrow d_0$  and  $k \leftarrow 0$ .

Step 2 (finding  $y_{k+1}$ ):

Case (i):  $v_{k+1}^1$  is to the left of  $y_k v_k^1$ : Search for  $y_{k+1}$  by traversing polygon  $P_2$  counterclockwise, starting at  $y_k$ , until one of the following conditions is satisfied (see fig. 2):

(a) The edge of  $P_2$  being considered (call it  $v_q^2 v_{q+1}^2$ ) is such that the foot (call it  $z$ ) of the perpendicular from  $v_{k+1}^1$  to  $v_q^2 v_{q+1}^2$  is between  $v_q^2$  and  $v_{q+1}^2$ . In this case, set  $y_{k+1} \leftarrow z$ .

(b) The vertex of  $P_2$  being considered (call it  $v_q^2$ ) is such that the perpendicular to  $v_{k+1}^1 v_q^2$  at  $v_q^2$  is a supporting line  $r$  of  $P_2$ . In this case, set  $y_{k+1} \leftarrow v_q^2$ .

Case (ii):  $v_{k+1}^1$  is to the right of  $y_k v_k^1$ : This case is similar to case (i), except that the search for  $y_{k+1}$  is done by traversing polygon  $P_2$  clockwise, starting at  $y_k$ .

Case (iii):  $v_{k+1}^1$  is on  $y_k v_k^1$ : Set  $y_{k+1} \leftarrow y_k$ .

Step 3 (next iteration):

Set  $D_{CMMR}(P_1, P_2) \leftarrow D_{CMMR} + \|v_{k+1}^1 - y_{k+1}\|^2$  and  $k \leftarrow k+1$ . If  $k = n_1 - 1$ , then stop. Otherwise, return to step 1.

The correctness proof of the above algorithm is straightforward, since the distances from all vertices are taken into account. Lemma 3 guarantees that polygon  $P_2$  will not be traversed more than twice. Then:

LEMMA 4 (Atallah [1983]): The algorithm presented above has time complexity  $O(n_1+n_2)$ .

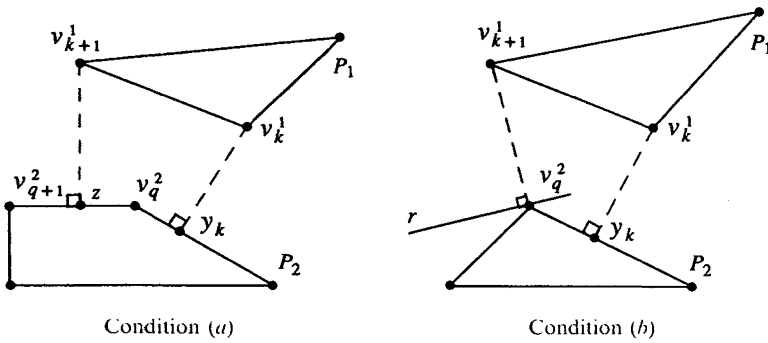


Figure 2. - Illustration of case (i).

We present in the following computational results obtained on randomly generated problems. For each value of  $n \in \{5, 10, 15, 20, 25, 30, 35\}$ , ten test problems are considered, each of which defined by two convex  $n$ -gons. The vertices of  $P_1$  are uniformly distributed in the unit square with diagonal  $((0, 0), (1, 1))$ , while those of  $P_2$  are in that with diagonal  $((0, 0), (1, -1))$ . The vertices of each  $n$ -gon are generated one-at-a-time, in order to ensure convexity. Table I gives the average computational times observed for each value of  $n$  on a 12 MHz IBM PC/AT microcomputer with an 80287 arithmetic processor, for both trivial  $O(n_1 n_2)$  and linear time algorithms.

TABLE I  
Computational times for the case of disjoint polygons (in seconds).

$n$	Alg. $O(n_1 n_2)$	Alg. $O(n_1 + n_2)$
5	0.203	0.099
10	0.763	0.226
15	1.688	0.345
20	2.989	0.490
25	4.632	0.615
30	6.675	0.714
35	9.081	0.855

4. A LINEAR TIME ALGORITHM FOR NON-DISJOINT POLYGONS

We address in this section the computation of  $D_{CMMR}(P_1, P_2)$  in the general case where  $P_1$  and  $P_2$  are not necessarily disjoint. We assume that neither polygon is entirely contained in the other, the problem is trivial

otherwise. (In this case, suppose, without loss of generality, that  $P_1 \subset P_2$ . Then, the distance from every point of  $P_1$  to  $P_2$  is zero. To compute the distances from all vertices of  $P_2$  to  $P_1$ , it suffices to traverse clockwise the vertices of  $P_1$  and  $P_2$ .) Given  $P_1$  and  $P_2$ , let

$$P = (P_1 \cup P_2) - \text{Interior}(P_1 \cap P_2),$$

*i. e.*, the region of the plane in  $P_1$  or  $P_2$  but not in the interior of their intersection. As both the union and the intersection of two convex polygons can be computed in linear time (see e.g. O'Rourke, Chien, Olson and Naddor [1982]), a description of region  $P$  can also be computed in linear time.

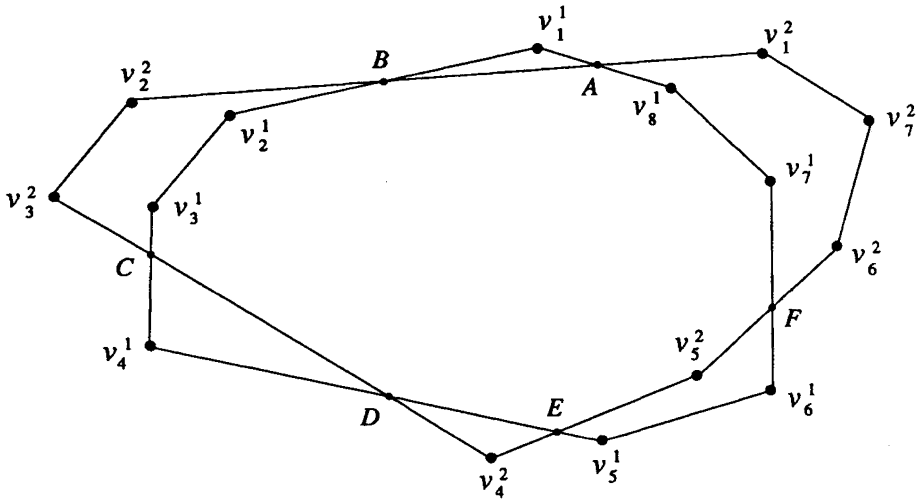
Region  $P$  consists of a number  $m = O(n)$  of pieces, where  $n = \max\{n_1, n_2\}$ , each of which is typically non-convex. The boundary of each piece consists of two polygonal chains: one coming from the boundary of  $P_1$  (which we call a  $P_1$ -chain), the other coming from that of  $P_2$  (which we call a  $P_2$ -chain); see *fig. 3* for an example. Suppose that the pieces appearing in the piece-decomposition of  $P$  are numbered counterclockwise, in the same way as the vertices of the polygons. Let us denote by  $P_1$ -chain $_k$  (resp.  $P_2$ -chain $_k$ ) the  $P_1$ -chain (resp.  $P_2$ -chain) in the  $k$ -th piece appearing in the piece-decomposition of  $P$  and by  $n_1^k$  (resp.  $n_2^k$ ) its number of vertices. The predecessor and the successor pieces of the  $k$ -th piece are, respectively, the  $k - 1 \bmod m$  and the  $k + 1 \bmod m$  pieces of the piece-decomposition of  $P$ .

Given the  $k$ -th piece, suppose that  $P_1$ -chain $_k$  (resp.  $P_2$ -chain $_k$ ) is the external (resp. internal) chain, *i. e.*, all vertices of  $P_2$ -chain $_k$  are in the interior or in the boundary of  $P_1$ . Hence, the distances from all vertices of  $P_2$ -chain $_k$  (*i. e.*, the internal chain in the general case) to  $P_1$  are null and do not need to be taken into account in the computation of  $D_{CMMR}(P_1, P_2)$ . It follows from convexity that, for any vertex  $v_j^1 \in P_1$ -chain $_k$ , the point  $y_j$  of  $P_2$  closest to  $v_j^1$  (*i. e.*,  $y_j = \text{Proj}_{P_2}(a_j^1, b_j^1)$ ) is in either one the  $P_2$ -chains indexed by  $k$  (which is an internal chain),  $k - 1 \bmod m$  or  $k + 1 \bmod m$  (which are external chains):

**THEOREM 1:** *Let  $v_j^1 \in P_1$ -chain $_k$  and  $y_j = \text{Proj}_{P_2}(a_j^1, b_j^1)$  be the point of  $P_2$  closest to  $v_j^1$ . Then, either (a)  $y_j \in P_2$ -chain $_k$  or (b)  $y_j \in P_2$ -chain $_{k-1 \bmod m}$  or  $y_j \in P_2$ -chain $_{k+1 \bmod m}$ .*

*Proof:* Case (a):  $y_j \in \text{Interior}(P_1)$ . Suppose that  $y_j \notin P_2$ -chain $_k$ . Then, the line segment  $v_j^1 y_j$  intersects  $P_2$ -chain $_k$ . Hence, there is a point of  $P_2$ -chain $_k$  which is closer to  $v_j^1$  than  $y_j$ . Therefore, if  $y_j$  lies in  $\text{Interior}(P_1)$ , it is necessarily a point of  $P_2$ -chain $_k$ .





$n_1 = 8$      $n_2 = 7$      $m = 6$   
 $P_1\text{-chain}_1 = \{A, v_1^1, B\}$   
 $P_1\text{-chain}_2 = \{B, v_2^1, v_3^1, C\}$   
 $P_1\text{-chain}_3 = \{C, v_4^1, D\}$   
 $P_1\text{-chain}_4 = \{D, E\}$   
 $P_1\text{-chain}_5 = \{E, v_5^1, v_6^1, F\}$   
 $P_1\text{-chain}_6 = \{F, v_7^1, v_8^1, A\}$   
 $\text{external}_1 \equiv P_1\text{-chain}_1$

$P_2\text{-chain}_1 = \{A, B\}$   
 $P_2\text{-chain}_2 = \{B, v_2^2, v_3^2, C\}$   
 $P_2\text{-chain}_3 = \{C, D\}$   
 $P_2\text{-chain}_4 = \{D, v_4^2, E\}$   
 $P_2\text{-chain}_5 = \{E, v_5^2, F\}$   
 $P_2\text{-chain}_6 = \{F, v_6^2, v_7^2, v_1^2, A\}$   
 $\text{internal}_1 \equiv P_2\text{-chain}_1$

$n_1^1 = 1$      $n_2^1 = 0$   
 $n_1^2 = 2$      $n_2^2 = 2$   
 $n_1^3 = 1$      $n_2^3 = 0$   
 $n_1^4 = 0$      $n_2^4 = 1$   
 $n_1^5 = 2$      $n_2^5 = 1$   
 $n_1^6 = 2$      $n_2^6 = 3$

Figure 3. - Piece-decomposition of region P.

Case (b):  $y_j \notin \text{Interior}(P_1)$ . Suppose that  $y_j$  belongs to an external  $P_2$ -chain other than  $P_2\text{-chain}_{k-1 \bmod m}$  or  $P_2\text{-chain}_{k+1 \bmod m}$ . Then, again, the line segment  $v_j^1 y_j$  intersects  $P_2\text{-chain}_k$ . Hence, there is a point of  $P_2\text{-chain}_k$  which is closer to  $v_j^1$  than  $y_j$ . Therefore, if  $y_j$  does not lie in  $\text{Interior}(P_1)$ , it is necessarily a point of either  $P_2\text{-chain}_{k-1 \bmod m}$  or  $P_2\text{-chain}_{k+1 \bmod m}$ . ■

In the general case, Theorem 1 amounts to say that, given the vertices of any of the external chains appearing in the piece-decomposition of P, their contributions to  $D_{\text{CMMR}}(P_1, P_2)$  can be evaluated just by looking at their distances to the internal chain of this same piece and to the external chains of their predecessor and successor pieces.

We show now how to compute these distances efficiently. Without loss of generality, suppose that we want to compute the distance from some vertex  $v_j^1$  of the external chain  $P_1\text{-chain}_k$  to  $P_2$  (i.e., to chains  $P_2\text{-chain}_k$ ,  $P_2\text{-chain}_{k-1 \bmod m}$  and  $P_2\text{-chain}_{k+1 \bmod m}$ ). We consider first the computation

of its distance to any of the external chains  $P_2$ -chain $_{k-1 \bmod m}$  or  $P_2$ -chain $_{k+1 \bmod m}$ , say the latter. To do so, suppose that we transform  $P_2$ -chain $_{k+1 \bmod m}$  into a convex polygon by creating an artificial edge joining its first and last vertices. Again from convexity, the distance we want to compute is the same distance from  $v_j^1$  to the polygon so formed. Suppose now that we proceed in the same way with  $P_1$ -chain $_k$ . Since  $P_1$ -chain $_k$  and  $P_2$ -chain $_{k+1 \bmod m}$  are disjoint, except for the last vertex of the former and the first one of the latter (which is a point of the intersection of the polygonal lines associated with  $P_1$  and  $P_2$ , hence not necessarily a vertex of either  $P_1$  or  $P_2$ ), the linear time algorithm given in Section 3 can now be used to compute in time  $O(n_1^k + n_2^{k+1 \bmod m})$  the distances from all vertices of  $P_1$ -chain $_k$  to  $P_2$ -chain $_{k+1 \bmod m}$ , and not only the distance from  $v_j^1$ . The same idea holds for the computation of the distances from all vertices of  $P_1$ -chain $_k$  to  $P_2$ -chain $_{k-1 \bmod m}$  in time  $O(n_1^k + n_2^{k-1 \bmod m})$ .

We consider now the computation of the distance from  $v_j^1$  to the internal chain  $P_2$ -chain $_k$ . Suppose that instead of computing just the distance from  $v_j^1$ , we compute the distances from all vertices of  $P_1$ -chain $_k$  to chain  $P_2$ -chain $_k$ , in the same way as in the previous paragraph. The idea here is to walk along  $P_1$ -chain $_k$  and  $P_2$ -chain $_k$  (like the way two sorted lists are merged), computing as we go along for each vertex of  $P_1$ -chain $_k$  that we encounter the point of  $P_2$ -chain $_k$  closest to it. What makes it work is the fact that, as we move monotonically along  $P_1$ -chain $_k$ , we also move monotonically along  $P_2$ -chain $_k$ . Computing the distances from all vertices of  $P_1$ -chain $_k$  to  $P_2$ -chain $_k$  takes hence  $O(n_1^k + n_2^k)$ . The algorithm follows:

**Begin**

Compute the union and the intersection of  $P_1$  and  $P_2$  using any linear time algorithm.

Compute and store the piece-decomposition of  $P$ .

Let  $m$  be the number of pieces in the piece-decomposition of  $P$  and  $n_1^k$  (resp.  $n_2^k$ ) be the number of vertices of  $P_1$  (resp.  $P_2$ ) in chain  $P_1$ -chain $_k$  (resp.  $P_2$ -chain $_k$ ), for  $k = 1, 2, \dots, m$ .

Let  $external_k$  (resp.  $internal_k$ ) be the external (resp. internal) chain of piece  $k$ , for  $k = 1, 2, \dots, m$ . Set  $n_{ext}^k \leftarrow n_1^k$  (resp.  $n_{int}^k \leftarrow n_2^k$ ) if  $P_1$ -chain $_k$  is an external chain;  $n_{ext}^k \leftarrow n_2^k$  (resp.  $n_{int}^k \leftarrow n_1^k$ ) otherwise.

Set  $D_{CMMR} \leftarrow 0$  and  $k \leftarrow 1$ .

**While**  $k \leq m$  **do**

**For each** node of  $external_k$  **do**

    Compute its distances  $d^{k-1 \bmod m}$ ,  $d^k$  and  $d^{k+1 \bmod m}$ , respectively to chains  $external_{k-1 \bmod m}$ ,  $internal_k$  and  $external_{k+1 \bmod m}$ .

    Set  $D_{CMMR} \leftarrow D_{CMMR} + (\min \{ d^{k-1 \bmod m}, d^k, d^{k+1 \bmod m} \})^2$ .

**end for each**

  Set  $k \leftarrow k + 1$ .

**end while**

**end begin**

The vertices of the two polygons are stored in two doubly chained circular lists. The pieces of the piece-decomposition of  $P$  are also stored in a doubly chained circular list, in such a way that whenever we need to compute the distances from all vertices of  $external_k$  to all those of  $external_{k-1 \bmod m}$  and  $external_{k+1 \bmod m}$ , the latter can both be easily accessed. The elements of this doubly chained circular list are themselves linear lists, whose first element corresponds to an intersection of two edges of  $P_1$  and  $P_2$  and the others are the vertices of the associated external chain originating at this intersection (see fig. 4). In fact, the internal chains do not need to be stored, since their vertices can easily be obtained by difference from their predecessor and successor external chains.

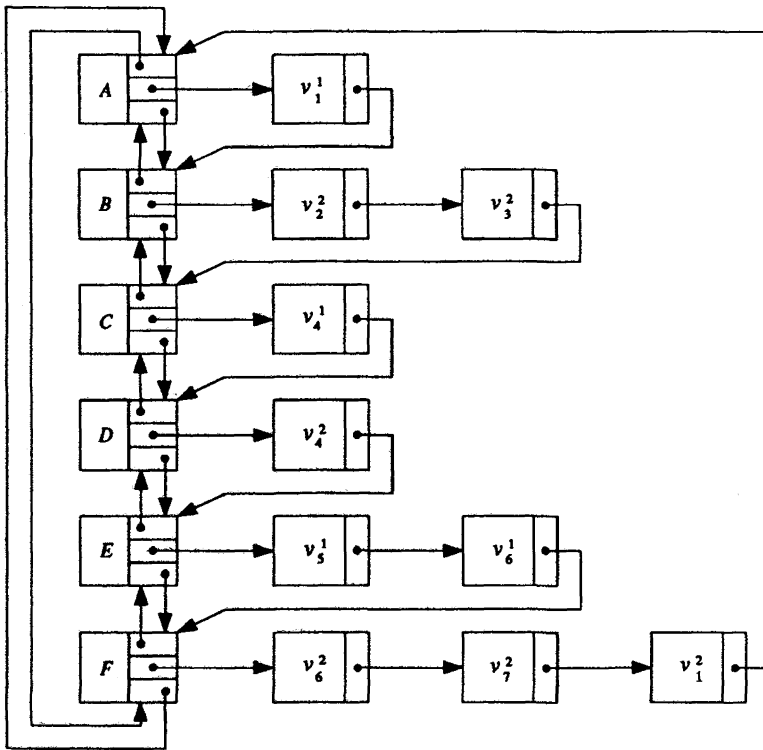


Figure 4. – Data structure for the example depicted in Figure 3.

**THEOREM 2:** *The algorithm described above has time complexity  $O(n)$ , where  $n = \max \{ n_1, n_2 \}$ .*

*Proof:* Denote by  $n_{ext}^k$  (resp.  $n_{int}^k$ ) the number of vertices of *external*<sub>k</sub> (resp. *internal*<sub>k</sub>). The computation of the distances from all vertices of *external*<sub>k</sub> to the other polygon takes  $O(n_{ext}^k + n_{int}^k + n_{ext}^{k-1 \bmod m} + n_{ext}^{k+1 \bmod m})$ . Hence, the overall complexity of the algorithm is

$$\begin{aligned} & \sum_{k=1}^m O(n_{ext}^k + n_{int}^k + n_{ext}^{k-1 \bmod m} + n_{ext}^{k+1 \bmod m}) \\ &= \sum_{k=1}^m O(n_{ext}^k + n_{int}^k) + \sum_{k=1}^m O(n_{ext}^{k-1 \bmod m}) + \sum_{k=1}^m O(n_{ext}^{k+1 \bmod m}) \\ &= O(n_1 + n_2) + O(n) + O(n) = O(n). \quad \blacksquare \end{aligned}$$

5. COMPUTATIONAL RESULTS AND CONCLUSIONS

The algorithm presented in Section 4 was applied to the same test problems described in Section 3, with the only modification that both polygons are now in the same unit square. The average computational times observed for each value of  $n$  (over ten test problems) on the same 12 MHz IBM PC/AT microcomputer with an 80287 arithmetic processor are presented in Table II, for a trivial  $O(n_1 n_2)$  algorithm and for the linear time algorithm presented in Section 4.

TABLE II  
Computational times for the case of non-disjoint polygons (in seconds).

$n$	Alg. $O(n_1 n_2)$	Alg. $O(n)$
5	0.252	0.254
10	0.786	0.468
15	1.436	0.615
20	2.591	0.853
25	3.587	0.951
30	5.174	1.121
35	6.532	1.593

It was noticed during the computational experiences reported in Cox, Maitre, Minoux and Ribeiro [1989] that most of the computational time observed for the solution of the problem of optimal matching of convex polygons corresponds to computations of the distance function between them. The linear time algorithm described here can then be used to considerably speed up the approach proposed in that paper. A very useful result would

be the extension of the algorithm presented in Section 4 for the computation of the distance between two convex polyhedra in  $\mathbb{R}^3$ .

#### REFERENCES

1. M. J. ATALLAH, A Linear Time Algorithm for the Hausdorff Distance between Convex Polygons, *Inform. Process. Lett.*, 1983, 17, pp. 207-209.
2. P. COX, H. MAITRE, M. MINOUX and C. C. RIBEIRO, Optimal Matching of Convex Polygons, *Pattern Recognition Lett.*, 1989, 9, pp. 327-334.
3. H. EDELSBRUNNER, Computing the Extreme Distances between Two Convex Polygons, *J. Algorithms*, 1985, 6, pp. 213-224.
4. B. GRUNBAUM, 1967, Convex Polytopes, *Wiley*, New York.
5. J. O'ROURKE, C.-B. CHIEN, T. OLSON and D. NADDOR, A New Linear Time Algorithm for Intersecting Convex Polygons, *Comput. Graph. Image Process.*, 1982, 19, pp. 384-391.