An Efficient Algorithm for the Computation of the Metric Average of Two Intersecting Convex Polygons, with Application to Morphing

Evgeny Lipovetsky[†] and Nira Dyn[‡]
School of Mathematical Sciences, Tel-Aviv University, Israel

Abstract. Motivated by the method for the reconstruction of 3D objects from a set of parallel cross sections, based on the binary operation between 2D sets termed "metric average", we developed an algorithm for the computation of the metric average between two intersecting convex polygons in 2D. For two 1D sets there is an algorithm for the computation of the metric average, with linear time in the number of intervals in the two 1D sets. The proposed algorithm has linear computation time in the number of vertices of the two polygons. As an application of this algorithm, a new technique for morphing between two convex polygons is developed. The new algorithm performs morphing in a non-intuitive way.

1. Introduction. In this work an algorithm for the computation of the metric average between two intersecting convex polygons in 2D is developed and studied. The metric average of two compact sets is a union of the weighted averages between any point from any set of the two, and the subset of all the closest points to it from the other set.

The original application of the metric average is for "piecewise linear" approximation of set-valued functions [2]. It is applied in spline subdivision schemes for compact sets, a procedure which is motivated by the problem of the reconstruction of a 3D smooth object from its parallel cross-sections [4]. An algorithm for the computation of the metric average of 1D compact sets with computation time which is linear in the number of connected subsets in the two 1D sets is introduced in [3].

The metric average of two sets is a subset of the Minkowski average of the sets and generally is a non-convex set (even for two convex sets). The Minkowski average is much bigger than the metric average. For example the Minkowski average of a non-convex set with itself is a larger set containing it, while the metric average is the set itself. The metric property of the metric average is that its Hausdorff distance from any one of the averaged sets changes linearly with the weight parameter in the average.

For the reconstruction problem of a smooth 3D object from its parallel cross-sections, the assumption that the projections of two consecutive cross-sections into a parallel plane intersect significantly, is natural. The choice of convex polygons was made although for such polygons the reconstruction from parallel cross-sections can be done by using Minkowski sums [5]. Yet the algorithm developed in this work is regarded as a first step in developing an efficient algorithm for the computation of the metric average of two general polygons. The authors have recently developed a general algorithm for the computation of the metric average of two intersecting regular polygons.

An additional outcome of this work is a new morphing technique between two convex polygons, based on the metric average. The morphing of shapes or objects is a common task in producing animations and visual effects. The goal of a morphing process is to

---

[†]eug@post.tau.ac.il
[‡]niradyn@post.tau.ac.il

change “smoothly” a source shape or object into the target shape or object. An intuitive way to do this is to find a correspondence between elements in the two shapes, and to map these elements into each other. For example in [1] methods based on correspondence between meshes are reviewed.

We suggest a new non-intuitive approach to the morphing of 2D shapes, which does not require a search for corresponding parts. We produce the intermediate objects as the metric average computed with suitable weights between the source shape and the target shape, mapped by a rigid motion to intersect significantly the source. The performance of the new technique is demonstrated by an example. Note that the intermediate shapes are non-convex, although the source and the target shapes are convex polygons.

Here is the outline of the work: Section 2 gives the background, it introduces the metric average and studies the internal structure of the computed set. Section 3 presents the algorithm. First it describes how to split the original problem into several simple ones, and presents several sub-algorithms, each one solving a simple sub-problem. Then all the sub-algorithms are integrated into the final algorithm. Complexity analysis of the algorithm is performed in Section 4. An example demonstrates the performance of the algorithm in Section 5. Our morphing technique based on the algorithm from Section 3 is studied in Section 6. Some important properties of this technique are observed and proved, and an example demonstrating its performance is given in Section 7.

2. Preliminaries. The metric average is a binary operation defined on two sets, which depends on a parameter $t \in [0,1]$. To define this binary operation we first introduce some notations.

For two sets A and B in R^n we define

$$A \setminus B = \{ a: a \in A,\ a \notin B \}.$$

$\prod_A(b) \subset A$ is the set of all closest points to some $b \in R^n$ from the set A, termed the projection of b on the set A.

An addition of two sets is the Minkowski sum of these sets

$$A + B = \{ a + b: a \in A, b \in B \}.$$

A multiplication of a set by a scalar is the set

$$tA = \{ ta: a \in A \}.$$

| ab | is the Euclidean distance between the points a, b.

[a,b] is the segment with boundary points a, b.

The metric average of two compact sets $A, B \in R^n$ and a weight $t \in [0,1]$ is defined as the set:

$$A \oplus_t B = \{ t\{ a \} + (1-t) \prod_B(a): a \in A \} \cup \{ t\prod_A(b) + (1-t)\{ b \}: b \in B \}.$$

It is easy to verify that $A \oplus_t A = A$, $A \oplus_0 B = B$, $A \oplus_1 B = A$. An example of the metric average of two non-convex sets in 1D is given in Fig.1. An example of the metric average of two intersecting convex polygons is given in Fig. 6.

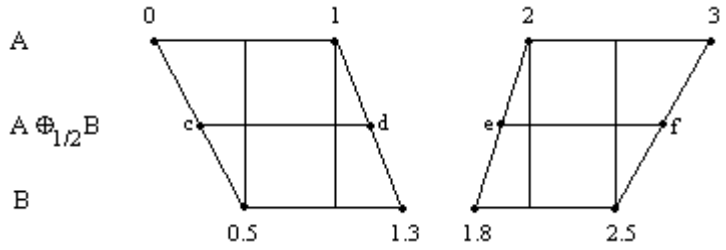

Fig.1.The metric average of two non-convex sets in 1D.
$A=[0,1]\cup[2,3]$, $B=[0.5,1.3]\cup[1.8,2.5]$, $A\oplus_{1/2}B=[c,d]\cup[e,f]$.

Introducing the non-symmetric operation between two sets,

$$M_t(A, B) = \{ t\{ a \} + (1-t) \prod_B(a): a\in A \},$$

and taking into consideration the fact that for all $p\in A\cap B$, $p\in A\oplus_t B$ (since for $p\in A\cap B$, $p = \prod_A(p) = \prod_B(p)$),the metric average is a union of three sets:

$$A \oplus_t B = M_t(A\backslash B, B) \cup M_{(1-t)}(B\backslash A, A) \cup (A\cap B).$$

In the following we limit the discussion to the case where the two sets are two intersecting convex polygons A, $B\in R^2$. We compute the set $A \oplus_t B$ by computing its boundary.

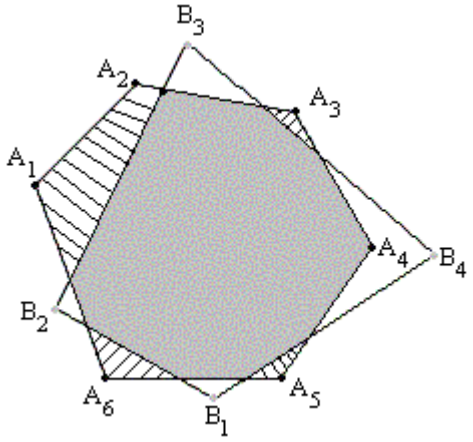

Fig. 2. Two intersecting convex polygons and their pockets.
≡ - A-pockets, - B-pockets, intersection polygon in gray

As we can see in Fig. 2, there is an intersection polygon $A\cap B$ (the gray area) which is convex and there are "pockets" (the white area). We define a pocket as a connected component of the symmetric difference between the two polygons. A disjoint union of all the pockets is the symmetric difference between the two polygons A and B.

A pocket is a simple polygon, not necessarily a convex polygon. A simple polygon is a simply connected polygon without holes and self-intersections of the boundary.

The boundary of a pocket consists of two polylines. One of these polylines is a part of A's boundary, and is denoted by p_A, the second is a part of B's boundary and is denoted by p_B. Each pocket has a common polyline with the boundary of the intersection polygon $A\cap B$. We call a pocket an "A-pocket" if it is a subset of A. Similarly, we call a pocket a "B-pocket" if it is a subset of B.

The computation of boundary($A \oplus_t B$) is done in terms of an operation termed the *exterior boundary average* of a pocket. Here we introduce this operation. For a pocket P, $Extb_t(P)$ is a polyline defined as

$$Extb_t(P) = \begin{cases} M_t(p_A, p_B), & P \subset A, \\ M_t(p_B, p_A), & P \subset B, \end{cases}$$

Observe that for A, B convex polygons in R^2 A\B is a disjoint union of simple polygons which are A-pockets $P_1 \ldots P_N$ and B\A is a disjoint union of simple polygons which are B-pockets $Q_1 \ldots Q_M$.

With this notation the boundary of the metric average $A \oplus_t B$, $t \in [0,1]$ is given by

$$\text{boundary}(A \oplus_t B) = \bigcup_{i=1}^{N} Extb_t(P_i) \quad \cup \quad \bigcup_{j=1}^{M} Extb_{(1-t)}(Q_j).$$

The algorithm we present in the next section is based on the above representation of boundary($A \oplus_t B$).

3. The algorithm. We have seen in the previous section that it is possible to reduce the original problem of calculating the boundary of $A \oplus_t B$ to a smaller one – namely of calculating the exterior boundary average of pockets. We limit the following discussion to the case of an A-pocket since the processing of a B-pocket is similar.

The boundary of an A-pocket P consists of two polylines p_A and p_B. p_B denotes the common boundary of P and $A \cap B$, and is termed the interior boundary. p_A is termed the exterior boundary.

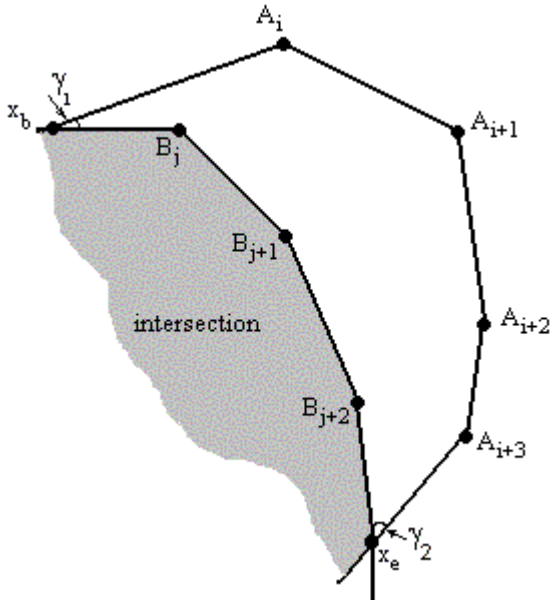

Fig. 3. An example of an A-pocket.
P_A is the polyline through $\{x_b, A_i, A_{i+1}, A_{i+2}, A_{i+3}, x_e\}$. P_B is the polyline through $\{x_b, B_j, B_{j+1}, B_{j+2}, x_e\}$.

Denote by $p_A^1, \ldots, p_A^n$ the ordered vertices of p_A, $p_B^1, \ldots, p_B^m$ the ordered vertices of p_B, x_b and x_e the intersection points between p_A and p_B and γ_1 and γ_2 the intersection angles between p_A and p_B at x_b and x_e respectively (See Fig. 3). We define a pocket to be

"acute" if both angles γ_1 and γ_2 are less than $\pi/2$, otherwise the pocket is called "non-acute". The closest points to a point $p \in p_A$ from the boundary of B in an acute pocket are in p_B (see Proposition 3.2), which is not necessarily the case for a non-acute pocket. This observation is in the basis of the proof of the linear complexity of the algorithm (see Section 4).

The algorithm for calculating $Extb_t(P)$ processes the points on the exterior boundary of the A-pocket P looking for the nearest points on the boundary of the polygon B. The geometric property stated in Proposition 3.1 allows us to calculate projections only for the breakpoints of the exterior boundary.

Proposition 3.1 Let $q^j = \prod p_B(p_A^j)$, $j = 1,\ldots,n$. Then $q^1,\ldots,q^n$ are ordered along p_B. Moreover, for any point v in the segment $[p_A^j, p_A^{j+1}]$, $j = 1,\ldots,n-1$, $\prod p_B(v)$ is in the part of p_B between q^j and q^{j+1}.

Proof: First note that for any point u in a segment of the form $[v, \prod p_B(v)]$, with v a point on p_A, we have $\prod p_B(u) = \prod p_B(v)$. Also, since p_B is convex, the set $\prod p_B(v)$ for any point v in the pocket P, consists of a unique point [7]. Thus two segments of the form $[v, \prod p_B(v)]$ and $[w, \prod p_B(w)]$, with v, w on p_A, cannot intersect. This proves the claim of the proposition. ◢

For an acute pocket the search of $q^1,\ldots, q^n$ is straight forward.

Proposition 3.2: For an acute pocket P with boundaries p_A and p_B, $\prod_B(p) \subset p_B$, for all $p \in p_A$.

Proof: P is an acute pocket. So the angles γ_1 and γ_2 (see Fig. 3) are less than $\pi/2$. So both $\prod_B(p_A^1)$ and $\prod_B(p_A^n)$ are in p_B. A repeated application of proposition 3.1 yields the result. ◢

The computation of the exterior boundary average of a pocket can be further reduced to several simpler operations.

Calculating $\Pi_s(p)$ – the closest point to a point $p = (p_x, p_y)$ in a segment s in R^2.

This algorithm checks whether the projection of point p to the line containing the segment s is in the segment. If it is then the projection is returned. If not then the nearest end point of the segment is returned. ◢

For a point x and a polyline p, we denote by $\prod_p(x, h, d)$ the operation that starts a search from the seed $h \in p$ in the search direction d to compute $\prod_p(x)$. Using this operation the search for the closest points from the boundary of B to the points of p_A is in one direction in the case of acute pockets. This property of the algorithm guarantees its linear complexity (see Section 4).

Calculating $\Pi_p(x, h, d)$ – the closest point to a point x in a polyline *p* using a seed h and a search direction $d \in \{ -1, 1 \}$.

We denote by $p^1, \ldots, p^m$ the ordered vertices of p.

1) Let p^j be the first end point of the segment that h belongs to (relative to d).
   set $p^j := h$; set k: = j
2) If d = 1 then limit := m – 1
   Else limit := -2

3) While dk ≤ limit do:
   3.1 $p_{curr} = \prod_{[p^k, p^{k+d}]}(x)$
   3.2 If $p_{curr} \neq p^{k+d}$ then break loop 3.
4) Return p_{curr}. ◢

Remark: to calculate the closest point to a point x in a polyline p we use the previous algorithm with $h = p^1$ and d=1.

The following algorithm is the core algorithm for the computation of $Extb_t(P)$.

Calculating $M_t(s, p)$ for a segment *s* (with end points s_1 and s_2) and a polyline *p*.

We assume that *s* and p do not intersect (See Fig. 5.) or have an intersection point which is either s_1 or s_2. In calculating $Extb_t(P)$, $s \in p_A$ either does not intersect p_B or intersects p_B only at one of its boundary points. (See Fig. 4.)

1) Find the points q_1 and q_2 in the polyline p such that $q_1 = \prod_p(s_1)$ and $q_2 = \prod_p(s_2)$. (Assumption: q_1 is a predecessor to q_2 in some order.)
2) Define by $v_0,\ldots,v_n$ the breakpoints of p between q_1 and q_2 and define $v_{-1} = q_1$, $v_{n+1} = q_2$.
3) Calculate $q'_\alpha = ts_\alpha + (1-t)q_\alpha$, $\alpha = 1,2$; define $y_{-1} = q'_1$, $y_{2n+2} = q'_2$.
4) For each breakpoint v_i of the polyline p between q_1 and q_2 do:
   4.1 Construct a perpendicular *w* to the segment $v_{i-1}v_i$ from v_i.
   4.2 Find the intersection x_{2i} between *w* and *s*.
   4.3 Construct a perpendicular *w'* to the segment v_iv_{i+1} from v_i.
   4.4 Find the intersection x_{2i+1} between *w'* and *s*.
   4.5 Calculate $y_\alpha = tx_\alpha + (1-t)v_i$, $\alpha = 2i, 2i+1$.
   4.6 Connect $y_{2(i-1)+1}$ from the previous step with y_{2i}.
   4.7 Connect y_{2i} with y_{2i+1}.
5) Connect y_{2n+1} with y_{2n+2}. ◢

To see that the result of the algorithm is $M_t(s, p)$ observe that for any $x_{2i-1} < x < x_{2i}$, $\prod_p(x) \in [v_{i-1}, v_i]$, since the line through x, which is perpendicular to the line through v_{i-1}, v_i intersects the latter inside $[v_{i-1}, v_i]$. Also by the continuity of the projection mapping, all points in $[x_{2i}, x_{2i+1}]$ are projected to the point v_i, because x_{2i} and x_{2i+1} are projected to v_i. Thus $[y_{2i}, y_{2i+1}] = M_t([x_{2i}, x_{2i+1}], p)$. Since for $x \in [x_{2i-1}, x_{2i}]$, $\prod_p(x) \in [v_{i-1}, v_i]$ then $M_t([x_{2i-1}, x_{2i}], p) = [y_{2i-1}, y_{2i}]$. The number of breakpoints in the resulting polyline is O(n).

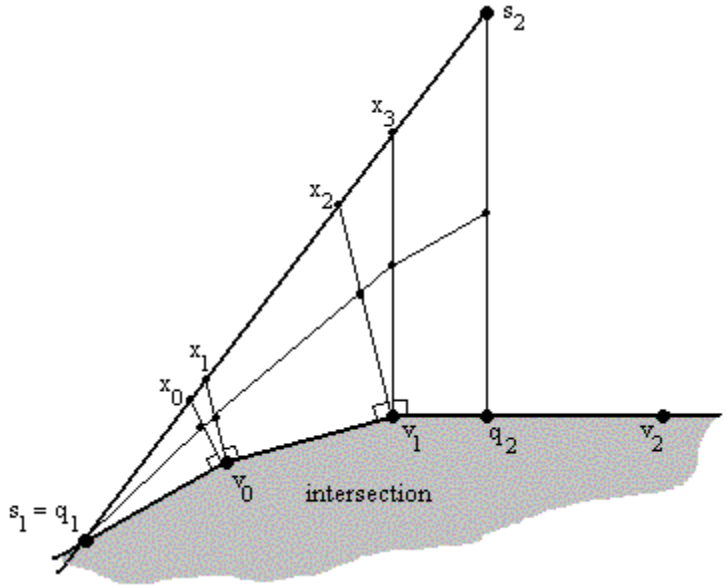

Fig. 4. Calculating $M_{1/2}(s, p)$ in the intersecting case.

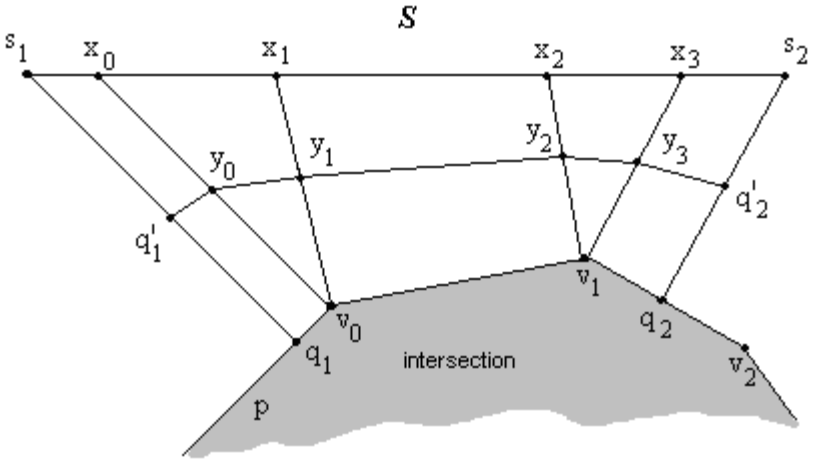

Fig. 5. Calculating $M_{1/2}(s, p)$ in the non-intersecting case.

Remark: the first step of the above algorithm can be omitted when calculating $Extb_t(P)$. Since q_1 is known from the previous step, only q_2 has to be calculated. Thus we introduce the operation $M_t(s, p, q_1, q_2)$ to replace $M_t(s, p)$ and compute q_2 prior to it.

We introduce two additional operations. For a polyline p and a point $q \in p$ we denote by $source_p(q)$ the index of the first end point of the segment in the polyline p that point q belongs to. In the same manner $target_p(q)$ denotes the second end point. For example, in Fig. 5 $v_1 = source_p(q_2)$ and $v_2 = target_p(q_2)$. We introduce the notation ∂B for the boundary of B. At this stage we have the tools for calculating $Ext_b(P)$.

Calculating $Extb_t$(P) for a pocket P.

1) If $\gamma_1 > \pi/2$ then dir := -1
   Else dir := 1
   Set $q_1 := x_b$
   Set $q_2 = \prod_{\partial B}(p_A^1, q_1, \text{dir})$.
   Set The_Result := $M_t(p_A^1 x_b, \partial B, q_1, q_2)$.
   $q_1 := q_2$
2) For each segment $p_A^i p_A^{i+1}$ in p_A (i = 1,...,n-1) do:
   2.1 If $| p_A^{i+1} source_{\partial B}(q_2) | < | p_A^{i+1} target_{\partial B}(q_2) |$ then dir := -1
   Else dir := 1
   2.2 Set $q_2 = \prod_{\partial B}(p_A^{i+1}, q_1^i, \text{dir})$.
   2.3 Set Current_result := $M_t(p_A^i p_A^{i+1}, \partial B, q_1, q_2)$.
   2.4 Concatenate Current_result with The_result
   2.5 Set $q_1 := q_2$
3) Concatenate $M_t(p_A^n x_e, \partial B, q_2, x_e)$ with the previously calculated polyline.

In the acute case the number of breakpoints in the produced polyline is O(n + m), where n is the number of breakpoints in p_A and m is the number of breakpoints in p_B.

Based on all previous algorithms, the algorithm for computing the boundary of the metric average of two intersecting convex polygons can be easily formulated.

Computing boundary(A $\oplus_t$ B)

1) Find the intersection A∩B and the pockets $T_1 \ldots T_k$.
2) For each pocket T_i, i = 1,..,k
   If T_i is an A-pocket Then calculate $Extb_t(T_i)$
   Else calculate $Extb_{1-t}(T_i)$
   Concatenate current result with the previous.

4. Complexity Analysis.

To estimate the complexity of our algorithm, we first derive a bound on the number of pockets of two intersecting convex polygons. For that we introduce the notation | A | for the number of vertices of a polygon A, and #{ A, B } for the number of pockets generated by the intersection of the polygons A, B .

Proposition 4.1: Let A and B be two intersecting convex polygons. Then
$\#\{ A, B \} \leq 2\min\{ | A |, | B | \}$.

Proof: We define an intersection of two polygons to be generic if the boundaries of the two polygons intersect at isolated points that are not vertices. Since degenerate cases do not increase the number of pockets relative to generic intersections, we can limit this proof to generic intersections. (See Fig. 2.)

As we mentioned above the boundary of a pocket P consists of two polylines: the inner polyline and the outer polyline. The minimal possible number of segments of an inner polyline is 1. (See Fig. 2.) Thus there are at most | B | A-pockets and at most | A | B-pockets. Each intersection point of the boundaries of A and B defines the end point of the

boundary polylines of a pocket and the starting point of the boundary polylines of the next pocket.

We term an A-pocket and a B-pocket as "corresponding" if the last intersection point of the polylines of the A-pocket is equal to the first intersection point of the polylines of the B-pocket. Here last and first refers to the clockwise order. Since each intersection point produces two corresponding pockets we may say that in the generic case the number of A-pockets is equal to the number of B-pockets. Due to this statement we have:
$\#\{ \text{A-pockets} \} = \#\{ \text{B-pockets} \} \leq \min\{ |A|, |B| \}$.
Thus the total number of pockets satisfies: $\#\{ A, B \} \leq 2\min\{ |A|, |B| \}$. ◢

Proposition 4.2: Let A be a convex polygon. Then the number of angles less than $\pi/2$ is at most 3.

Proof: Our proof is based on the following two properties of a convex polygon:

1. Each angle of a convex polygon is less than π.
2. If there are n vertices in a convex polygon then the sum of its angles is $\pi(n-2)$.

Let m be the number of acute angles in a convex polygon. Then $(n-m)\pi + m\pi/2 > (n-2)\pi$, showing that $m < 4$. ◢

Proposition 4.3: Let A and B be two convex intersecting polygons. Then the number of non-acute pockets is bounded by 6.

Proof: To each angle smaller than $\pi/2$ in $A \cap B$, there correspond at most two non-acute pockets (See Fig. 2). Since $A \cap B$ is a convex polygon, the number of angles smaller than $\pi/2$ in it is at most 3, and the number of non-acute pockets is at most 6. ◢

Remark: The number 6 is achieved in the intersection of an equilateral triangle with itself after a small rotation around its center.

The entire algorithm of calculating boundary($A \oplus_t B$) consists of two parts. In the first, $A \cap B$ is calculated and the pockets are determined. This is done in $O(|A| + |B|)$ operations [6].

The algorithm for calculating $Extb_t(P)$, for P – acute, process sequentially the vertices of the boundaries of the pocket, operating on each vertex at most twice.

The boundaries of any two pockets do not have common vertices, except for the vertices of $A \cap B$. The number of vertices of $A \cap B$ is the same as the number of pockets in the generic case, and hence is bounded by $2\min\{ |A|, |B| \}$, in view of Proposition 4.1. So the number of vertices in the boundaries of all pockets is bounded by $2(|A| + |B|)$, and the complexity of the calculation of $Extb_t(P)$ for all acute pockets is $O(|A| + |B|)$. In case of a non-acute pocket the number of processed vertices is bounded by $|A|$ or by $|B|$. By proposition 4.2, the complexity of computing $Extb_t(P)$ for at most 6 non-acute pockets is $O(|A| + |B|)$.

So the entire algorithm has $O(|A| + |B|)$ time complexity.

5. Example.

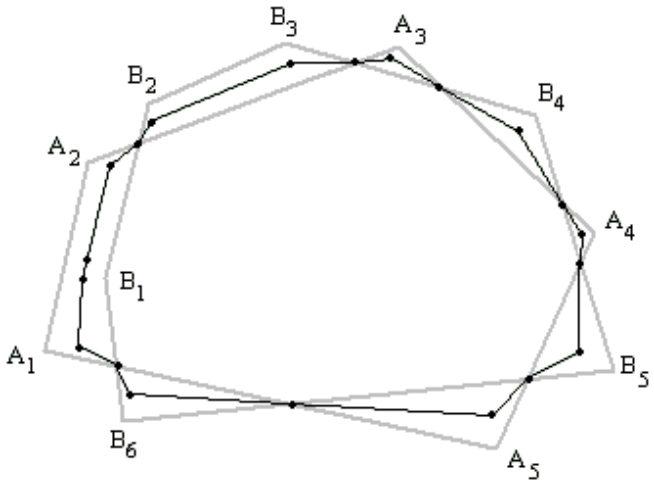

Fig. 6. A generic case: boundary($A \oplus_{1/2} B$) for a pentagon A and a hexagon B.
The boundary of the metric average is painted black. All pockets here are acute.

6. Application to Morphing. In this section we describe the application of the metric average to morphing. We allow morphing between two convex polygons, which are not necessarily intersecting.

Consider two convex (not necessarily intersecting) polygons A, $B \in R^2$ with the vertices ordered clock-wise. Our goal is to obtain a set of polygons $Q_0, \ldots, Q_n$ such that $Q_0 = A$, $Q_n = B$ and when the polygons $Q_0, \ldots, Q_n$ are displayed sequentially one after another, one at a time, an illusion of continuous transformation of A to B is created.
Our method is based on a continuous set-valued function S(t) satisfying S(1)=B, S(0)=A, which we descretize to get $Q_i = S(i/n)$, for i = 0, …, n.

First we define some operations on polygons. For a polygon $A \in R^2$ (with vertices $A_1, \ldots, A_n$) we define:

1) The point "center(A)" = (p_x, p_y) with

$$p_x = \Big(\sum_{i=1}^{n} (A_i)_x \Big) / n, \quad p_y = \Big(\sum_{i=1}^{n} (A_i)_y \Big) / n.$$

2) The polygon "translate(A, p)" with vertices:

$$(A_i)_x + (p_x - (\text{center}(A))_x),$$
$$i = 1, \ldots, n,$$
$$(A_i)_y + (p_y - (\text{center}(A))_y),$$

for a given point $p=(p_x,p_y)$.
Note that

$$\text{translate}(A, p) = A + \vec{p} - \vec{\text{center}}(A)$$

and that center(translate(A, p)) = p.

3) The polygon "$R_\alpha(A)$" = "rotate(A, α)" is obtained by a rotation of A in an angle α around its center. It has the vertices:

$$((A_i)_x - (\text{center}(A))_x)\cos(\alpha) + ((A_i)_y - (\text{center}(A))_y)\sin(\alpha) + (\text{center}(A))_x,$$
$$((A_i)_y - (\text{center}(A))_y)\cos(\alpha) - ((A_i)_x - (\text{center}(A))_x)\sin(\alpha) + (\text{center}(A))_y,$$
$$i = 1,\ldots,n.$$

The use of the metric average for morphing is effective only if the intersection part of the two polygons is significant. Therefore we add a rigid motion component to the morphing.

Determining the rigid motion component. We increase the intersection between A and B by a rigid motion to improve the visual effect. We obtain B'(rigidly moved B) in the following way:

1) Translate the center of polygon B to the center of polygon A.
2) Find μ, ν such that $A_\mu A_\nu$ is the longest diagonal in A, namely
$$|A_\mu A_\nu| = \text{Max}\{|A_i, A_j|, \ |i-j|>1, i,j = 1,\ldots,|A|\}$$
3) Find θ, η such that $B_\theta B_\eta$ is the longest diagonal in B, namely
$$|B_\theta B_\eta| = \text{Max}\{|B_k B_l|, |k-l|>1, k,l = 1,\ldots,|B|\}$$
4) Calculate the angle β between the lines determined by A_μ, A_ν and by B_θ, B_η.
5) Calculate the area $A \cap$ rotate(translate(B, center(A)), α) with $\alpha = \beta$ and $\alpha = \pi - \beta$. Set α to that angle which yields the maximal intersection area.
6) Set B' = rotate(translate(B, center(A)), α).
7) If there are several candidates for μ, ν or θ, η then check all the possible rotations and take the one corresponding to the maximal intersection area.
8) If area(A∩B) is greater than area(A∩B') then cancel the rigid motion and set B' = B and $\alpha = 0$.

The above choice of the rigid motion component has the important property:

Proposition 6.1: Let A and B be such that there exists a rigid motion transformation of B to A. Then the algorithm performs this transformation.

Proof: A rigid transformation consists of a translation and a rotation. First we consider the case of a convex polygon with a unique largest diagonal. The position of a polygon in R^2 is determined by the position of its center and by the angle between the polygon's largest diagonal and the positive direction of the x-axes, termed hereafter as the "positioning angle". After translating polygon B such that its center coincides with the center of polygon A, the algorithm chooses between two possible rotations around the center in an angle α and in an angle (π - α), where α is the difference between the positioning angles of A and B. After one of these two rotations B coincides with A. The algorithm performs this rotation, since after this rotation the area of the intersection is maximal.

In case of several maximal diagonals the algorithm compares each pair of diagonals for maximum intersection. At least one of the pairs achieves the coincidence of the two polygons. ◢

Performing the morphing of two convex polygons A and B. After applying the previous algorithm to the polygons A and B, and obtaining the rotation angle α, we define the continuous set-valued function S(t), $0 \le t \le 1$, such that S(1) = B, S(0) = A,

$$S(t) := R_{-\alpha t}(R_{\alpha}(B + \vec{b}) \oplus_t A) - \vec{b}\, t$$

with $\vec{b}$ = center(A) – center(B).
Note that in S(t) the metric average is performed between the two intersecting convex polygons A and B' generated by the algorithm in the previous paragraph. Also note that, up to a rigid transformation, the Hausdorff distance between S(t) and A is t times the Hausdorff distance between A and B', $0 \le t \le 1$.
We descretize S(t) and obtain polygons $Q_0,\ldots,Q_n$

$$Q_i = S(t_i) = R_{-\alpha t_i}(R_{\alpha}(B + \vec{b}) \oplus_{t_i} A) - \vec{b}\, t_i,$$

where $t_i = i/n$.

7. Example.
The following example illustrates the morphing of a hexagon A to a septagon B. Note that the morphing is not intuitive, since the intermediate polygons are not convex and the number of vertices is greater than that of A and of B.

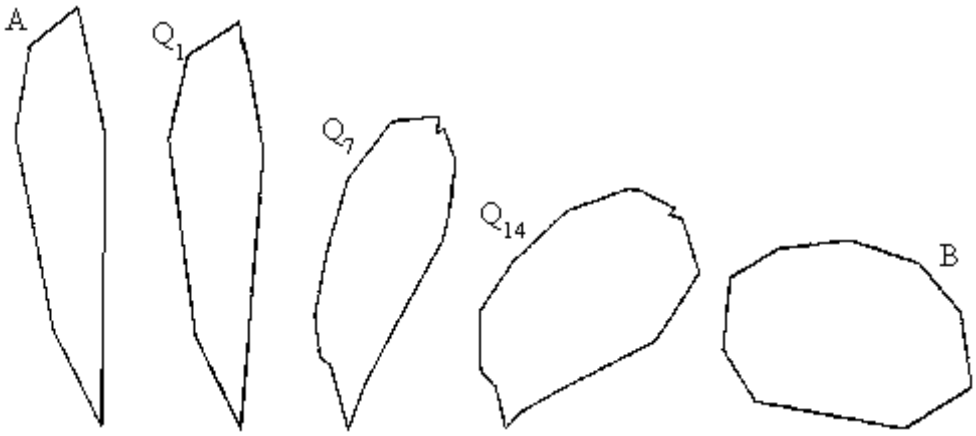

Fig. 7. Some snapshots of the morphing process. A generic case.

References

[1] M.Alexa, Mesh Morphing STAR. Eurographics 2001 State-of-the-Art Reports, (2001).

[2] Z.Artstein, Piecewise linear approximations of set-valued maps, Journal of Approx. Theory 56 (1989), 41-47

[3] R.Baier, N.Dyn and E.Farkhi, Metric averages of one dimensional compact sets, in Approximation theory X, C.Chui, L.L.Schumaker and J.Stoeckler (eds.), Vanderbilt Univ. Press, Nashville, TN, (2002), 9-22.

[4] N.Dyn and E.Farkhi, Spline subdivision schemes for compact sets with metric averages, in Trends in Approximation Theory, K.Kopotun, T.Lyche and M.Neamtu (eds.), Vanderbilt Univ. Press, Nashville, TN, (2001), 95-104.
[5] N.Dyn and E.Farkhi, Spline subdivision schemes for compact sets-a survey, Serdica Math. J. Vol.28 (4), (2002), 349-360.
[6] F.Preparata and M.Shamos. Computational Geometry: An Introduction. Texts and Monographs in Computer Science. Springer-Verlag, Berlin, Germany, 1985.
[7] R.T.Rockafellar, Convex Analysis, Princeton University Press, Princeton, NJ, 1970.