Separation Algorithm for Second Degree Convex Polygons

REIF MORDECHAI N.
Department of Computer Science
Jerusalem College of Technology-Machon Lev,
21 HaVaad HaLeumi St., Jerusalem,
Israel

Abstract. Many algorithms have been developed for testing the intersection of two convex polygons. We have developed for this problem a new optimal \( O(n) \) sequential and \( O(\log n) \) parallel algorithms. Our algorithm has four advantages: (1) the same algorithm is used for the sequential and parallel algorithm. (2) If the polygons do not intersect each other, the algorithm builds sequentially in \( O(\log n) \) time, and parallelly in \( O(1) \) time a data structure for all the feasible separating lines between them. (3) The algorithm could be used for second-degree convex polygons. (4) The algorithm could easily extended to 3-D.

Key-Words: - Convex polygons, Intersection, Separation, Computational geometry, Parallel algorithm.

1 Introduction.

The problem of reporting whether two convex polygons intersect each other arises in a number of applications. For examples CAD/CAM, wire and component layout in VLSI, computational geometry, motion planing, and collision detection. A number of algorithms was developed for solving this problem [CHA], [DOB], [ORK], [SHA], but basically all of them are sequential algorithms. Another disadvantage of those algorithms is that they can be used only for convex polygons with straight edges. The one that can be extended to convex splin polygons [DOB] assume the existents of an oracle for solving some supporting and intersection problems. In this paper we use the dual transformation from [MEL] to develop a sequential-parallel algorithm for this problem, which we can use also for second degree convex splin polygons, and for 3-D convex polyhedrons.

2 Second degree polygon.

Definition 2.1 Second degree segment.

A second degree segment \( \Gamma \subset \mathbb{R}^2 \) is: \( \Gamma = \{(x,y) | a_1x + a_2y + b = 0 \} \) a vertical segment. Or, \( x \in [x_1, x_2] \), \( y = f(x) \), \( f \) is constant, or a function of the first or second order.

Definition 2.2 Second degree convex polygon. (Fig-1).

A second degree convex polygon \( P \subset \mathbb{R}^2 \), is a convex set such that \( b(P) \) (P boundary) is the union of second degree segments. The edges of \( P \) are those second degree segments. A point \( v \in b(P) \) is a vertex of \( P \) if: or \( v = \{ \text{the intersection of two edges} \} \), or there is a vertical line \( h \) through \( v \) supporting \( P \) such that \( v = \{ h \cap b(P) \} \).

Fig-1. A Second Degree Convex Polygon.

Let \( \Omega \) be a three-dimensional projective space.

Definition 2.3 Boundary polygon \( L(P) \).

Let \( P \subset \mathbb{R}^2 \subset \Omega \) be a second degree convex polygon. Let \( V(P), E(P) \) be respectively the vertices and edges of \( P \). \( L(P) = \{ P \text{ boundary polygon} \} \) is the set of all lines supporting \( P \).

We can construct \( L(P) \) by:

1. Every edge \( e \in E(P) \) such that \( e \) is a straight segment, defined a supporting line \( l \in L(P) \).
2. If \( e \in E(P) \) is a second degree segments, then in \( \mathbb{R}^3 \subset \Omega \), \( e : aX^2 + bY^2 + cZ^2 + 2hXY + 2gXZ + 2fYZ = 0 \).

Therefore in \( \Omega \), \( e \) defined a supporting, line-conic set \( L_e = \{ l | l \in L(P) \} \) with equation:

\[
\begin{pmatrix}
a & h & g & X \\
h & b & f & Y \\
g & f & c & Z \\
X & Y & Z & 0 \\
\end{pmatrix} = 0.
\]
3. The vertex $v(a,b,1) \in V(P)$, defined a set $L_2=\{l|l \in L(P)\}$ of supporting lines whose equations are or: $l: Y=mX+n$, and $b=ma+n$. Or $l:X=a$.

3. **Dual Transformation.**

Let $\Omega$ be a three-dimensional projective space. Let $V(\Omega)$ be the set of all points in $\Omega$. Let $L(\Omega)$ be the set of all lines in $\Omega$. Let $M$ be the following metrics:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Let $v \in V(\Omega)$ and $h \in L(\Omega)$, we define two dual transformations $T_1, T_2: \Omega \rightarrow \Omega$.

**Definition 3.1 Dual Transformation.**

$h = T_1(v) = M^v h^v$, $v = T_2(h) = M^{-1} h^t$

$T_1$ and $T_2$ are isomorphism between $V(\Omega)$ and $L(\Omega)$. A point $v \in \mathbb{R}^2 \subseteq \Omega$ is the triple $(a,b,1)$, and a line $h \in \mathbb{R}^2 \subseteq \Omega$, $h:Ax+By+C=0$ is the triple $(A,B, C)$. $T_1$ transfer the line $y=mx+n$ to the point $(m,n,1)$, and $T_2$ transfer the point $(a,b,1)$ to the line $y=ax+b$, and $T$ transfer the line $y=mx+n$ to the point $(m,n,1)$.

In [MEL] there is a wide discussion on the dual transformation and its properties; we will bring the results, which are needed, for our algorithms.

Let $S \subseteq \mathbb{R}^2$ be a convex set with boundary $b(S)$. Let $H=\{v \text{the set of all lines in } \mathbb{R}^2\}$. We divide $H$ to seven sub sets, and divide $b(S)$ to three sub sets:

**H sub sets:**
1. $H_0(S)=\{h \in H| h \text{ intersect } S \text{. } h \text{ is of the form } y=mx+n\}$
2. $H_{x}(S)=\{h \in H| h \text{ a vertical line intersecting } S \text{. } h \text{ is of the form } x=c\}$
3. $H_{y}(S)=\{h \in H| h \text{ a vertical line supporting } S \text{. } h \text{ is of the form } x=c\}$
4. $H_{1}(S)=\{h \in H| h \text{ above } S\}$
5. $H_{2}(S)=\{h \in H| h \text{ below } S\}$
6. $H_{3}(S)=\{h \in H| h \text{ supports } S\}$
7. $H_{4}(S)=\{h \in H| h \text{ supports } S\}$

**b(S) sub sets:**
1. $b_1(S)=\{v \in b(S)|v \in b(S) \cap H_{0}(S)\}$
2. $b_2(S)=\{v \in b(S)|v \in b(S) \cap H_{x}(S)\}$
3. $b_3(S)=\{v \in b(S)|v \in b(S) \cap H_{y}(S)\}$

**Theorem 3.1 ([MEL] pp. 244-250).** (Fig. 2).

1. In $\mathbb{R}^2 \subseteq \Omega$, $U(S)=T_2(H_0(S))$ is a convex up set. $D(S)=T_2(H_0(S))$ is a convex down set. $U(S) \cap D(S)=\emptyset$. $T_2(H_0(S))$ is the set of points between $U(S)$ and $D(S)$. In $\Omega$, $T_2(H_0(S))$ is the ideal point set of $T_2(H_0(S))$. $T_2(H_0(S))$ is the ideal point set of $T_2(H_0(S))$.
2. For every point $v \in (S-b(S))$, $T_1(v)$ is a line strictly separating $U(S)$ and $D(S)$. And vice versa, for every line $h$ strictly separating $U(S)$ and $D(S)$, there is a point $v \in (S-b(S))$, such that $h=T_1(v)$.
3. Every line $h \in T_1(b_0(S))$, supports $U(S)$. Every line $h \in T_1(b_2(S))$, supports $D(S)$. And vice versa, for every line $h$ which supports $U(S)$, there is a point $v \in b_0(S)$ such that $h=T_1(v)$. For every line $h$ which supports $D(S)$, there is a point $v \in b_2(S)$ such that $h=T_1(v)$. Therefore, if $v \in b_0(S) \cap b_2(S)$ then $h=T_1(v)$ is a line supporting both $U(S)$ and $D(S)$.
4. For every line $h \in H_{1}(S)$, $T_2(h) \in b(U(S))$. For every line $h \in H_{2}(S)$, $T_2(h) \in b(D(S))$. And vice versa, for every point on $b(U(S))$ there is a line $h \in H_{1}(S)$ such that $T_2(h)$ and for every point on $b(D(S))$ there is a line $h \in H_{2}(S)$ such that $v=T_2(h)$.
5. Every straight edge (vertex) of $b_1(S), b_2(S)$ determine respectively a vertex (edge) of $U(S), D(S)$. If $b_1(S), b_2(S)$ vertices and edges order is clockwise order, then the determined vertices and edges of $b(U(S)), b(D(S))$ are in anti clockwise order.
6. Every second degree edge of $b_1(S), b_2(S)$ determine respectively a second degree edge of $U(S), D(S)$. This result is not from [MEL] but it is a straight result from the above 1-4.

We say that:
1. $\Delta(S) = U(S) \cup D(S)$ is the dual transformation of $S$.
2. $b(U(S))$ is the dual transformation of $b_1(S)$, and $b(D(S))$ is the dual transformation of $b_2(S)$.

![Fig-2 (a). A second degree convex polygon, and its dual transformation.](image-url)
Let $A,B \in \mathbb{R}^2$ be two convex sets. Let $CH(A,B)$ be the convex hull of $A \cup B$.

**Conclusion 3.1**

1. If $A \cap B \neq \emptyset$, then $\Delta(A \cap B) = CH(U(A) \cap U(B)) = CH(D(A), D(B))$.
2. If $A \cap B = \emptyset$, then, or $U(A) \cap D(B) \neq \emptyset$, or $D(A) \cap U(B) \neq \emptyset$.
3. $\Delta(CH(A,B)) = [U\{v_1, \ldots, v_n\}, E\{e_1, \ldots, e_{n-1}\}]$.

**3.1 Dual transformation of a second degree convex polygon.**

Let $P$ be a second degree convex polygon. Let $V(P) = \{v_i | i = 1, \ldots, n\}$, $E(P) = \{e_i | i = 1, \ldots, r\}$ be respectively the vertices and edges of $P$. Let $V_l(P) = \{v_i | i = 1, \ldots, k\} \subset V(P)$, and $E_l(P) = \{e_i | i = 1, \ldots, k-1\} \subset E(P)$ be respectively the vertices and edges of $b_l(P)$, and similarly $V_r(P) = \{v_i | k, \ldots, n\} \subset V(P)$, and $E_r(P) = \{e_i | k, \ldots, n\} \subset E(P)$ be respectively the vertices and edges of $b_r(P)$.

The dual transformation $\Delta(P)$ of $P$ is the following:

1. Let $v \in V(P)$, $v = e \cap e_{i+1}$. Then $\Delta(v_i)$ is an edge of $\Delta(P)$ with vertices $T_1(v_i) \cap T_2((L_o))$ and $T_2(v_i) \cap T_3((L_o))$.
2. Let $e \in E(P)$ be a straight segment, $e = [v_i, v_{i+1}]$.

Then $\Delta(e_i)$ is a vertex of $\Delta(P)$ such that $\Delta(e_i) = T_1(v_i) \cap T_2((L_o))$.
3. Let $e \in E(P)$ be a second degree segment with $f$ of second degree order, and with vertices $\{v_i, v_{i+1}\}$. Then $\Delta(e_i)$ is a second degree segment whose vertices are $T_1(v_i) \cap T_2((L_o))$ and $T_2(v_{i+1}) \cap T_3((L_o))$.

**Conclusion 3.2**

By theorem 3.1 we have that $\Delta(P) = U\{v_1, \ldots, v_n\}$ and $\Delta(P) = D\{e_1, \ldots, e_{n-1}\}$ of $P$ is the following:

1. Let $v \in V(P)$, $v = e \cap e_{i+1}$. Then $\Delta(v_i)$ is an edge of $\Delta(P)$ with vertices $T_1(v_i) \cap T_2((L_o))$ and $T_2(v_i) \cap T_3((L_o))$.
2. Let $e \in E(P)$ be a straight segment, $e = [v_i, v_{i+1}]$.

Then $\Delta(e_i)$ is a vertex of $\Delta(P)$ such that $\Delta(e_i) = T_1(v_i) \cap T_2((L_o))$.
3. Let $e \in E(P)$ be a second degree segment with $f$ of second degree order, and with vertices $\{v_i, v_{i+1}\}$. Then $\Delta(e_i)$ is a second degree segment whose vertices are $T_1(v_i) \cap T_2((L_o))$ and $T_2(v_{i+1}) \cap T_3((L_o))$.

**4 Algorithms.**

Let $P_1, P_2$ be two second degree convex polygons. We assume that the data structure, which presents the vertices and edges of $P_1$ and $P_2$, are well defined. Let:

1. $V_i(P_i) = \{v_{i,1}, \ldots, v_{i,k}\} \subset V(P_i)$, and $E_i(P_i) = \{e_{i,1}, \ldots, e_{i,k-1}\} \subset E(P_i)$ be respectively the vertices and edges of $b_i(P_i)$, and similarly $V_i(P_i) = \{v_{i,k}, \ldots, v_{i,n}\} \subset V(P_i)$, and $E_i(P_i) = \{e_{i,k}, \ldots, e_{i,n}\} \subset E(P_i)$ be respectively the vertices and edges of $b_r(P_i)$.
2. $V_i(P_i) = \{v_{i,1}, \ldots, r\} \subset V(P_i)$, and $E_i(P_i) = \{e_{i,1}, \ldots, e_{i,r}\} \subset E(P_i)$ be respectively the vertices and edges of $b_l(P_i)$, and similarly $V_i(P_i) = \{v_{i,k}, \ldots, m\} \subset V(P_i)$, and $E_i(P_i) = \{e_{i,k}, \ldots, m\} \subset E(P_i)$ be respectively the vertices and edges of $b_r(P_i)$.

**4.1 Intersection Algorithm.**

**Algorithm 4.1 (sequential).** Build $CH[P_1, P_2]$

1. Construct $\Delta(P_1) = U\{v_1, \ldots, v_n\}$ and $\Delta(P_2) = U\{v_1, \ldots, v_n\}$.
2. Construct $\Delta(P_1) = U\{v_1, \ldots, v_n\}$ and $\Delta(P_2) = U\{v_1, \ldots, v_n\}$.
3. Transform back $U[P_1, P_2]$ which gives the upper hull of $CH[P_1, P_2]$, and similarly transform back $D(P_2)$ which gives the lower hull of $CH[P_1, P_2]$.

**Algorithm 4.2 (sequential).** Build $P_1 \cap P_2$:

1. Construct $\Delta(P_1) = U\{v_1, \ldots, v_n\}$ and $\Delta(P_2) = U\{v_1, \ldots, v_n\}$.
2. Construct $\Delta(P_1) = CH[U(P_1), U(P_2)]$, and $\Delta(P_2) = CH[U(P_1), U(P_2)]$.
3. Transform back $U[P_1, P_2]$ which gives the upper hull of $P_1 \cap P_2$, and similarly transform back $D(P_2)$ which gives the lower hull of $P_1 \cap P_2$.

**Complexity of algorithm 4.1**

Step 1. By theorems 3.1 section 3, and conclusion 3.2, we can build the lists of vertices and edges of $\Delta(P_1)$ and $\Delta(P_2)$ from the ordered list of vertices and edges of $P_1$ and $P_2$. Because we assumed that the data structure, which presents the vertices and edges of $P_1$ and $P_2$, are well defined, therefore constructing the ordered lists of
vertices and edges of $\Delta(P_1)$ and $\Delta(P_2)$ from the ordered list of vertices and edges of $P_1$ and $P_2$ requires $O(n+m)$ time.

Step-2. To construct $U(P_{12})=U(P_1)\cap U(P_2)$ and $D(P_{12})=D(P_1)\cap D(P_2)$, (1) To merge the vertices and edges lists of $U(P_1)$ and $U(P_2)$, and of $D(P_1)$ and $D(P_2)$, requires $O(n+m)$ time. (2) Because the edges are represented by functions up to the second degree, therefore the intersection involved function up to the forth degree. Hence constructing the intersection between two edges requires $O(1)$ time. Therefore stepping along the merged lists and constructing the appropriate intersections requires $O(n+m)$ time.

Step-3=Step-1 and requires $O(n+m)$ time.

**Theorem 4.1**

The algorithm for constructing the common convex hull of two second degree convex polygons requires $O(n+m)$ time.

**Complexity of algorithm 4.2**

Step-1=step-1 of algorithm 4.1 and requires $O(n+m)$ time.

Step-2. To construct $U(P_{12})=CH[U(P_1), U(P_2)]$, and $D(P_{12})=CH[D(P_1), D(P_2)]$, we use algorithm 4.1 that requires $O(n+m)$ time.

Step-3=Step-1 and requires $O(n+m)$ time.

**Theorem 4.2**

The algorithm for constructing the intersection of two second degree convex polygons requires $O(n+m)$ time.

Using $n+m$ processors, algorithms 4.1 and 4.2 can be easily modified to parallel algorithms.

**Algorithm 4.3 (parallel).**

1. Construct $\Delta(P_1)=U(P_1)\cup D(P_1)$ and $\Delta(P_2)=U(P_2)\cup D(P_2)$. Every processor will contain an edge and its two vertices, and will transform them. Therefore step-1 of the algorithms requires $O(1)$ time and $O(n+m)$ work.

2. Construct $U(P_{12})=U(P_1)\cap U(P_2)$, and $D(P_{12})=D(P_1)\cap D(P_2)$. This step requires two sub steps:

   (1) a merging step. This step requires $O(lg(lg(n+m)))$ time and $O(n+m)$ work [Akl], [GUA].

   (2) Every processor will contain an edge and its two vertices of $\Delta(P_{12})$. Let $e_i\in U(P_1)$, if in the merged list of the vertices and edges lists of $U(P_1)$ and $U(P_2)$, $e_i$ has the same index $i$, then we have to check if there is an intersection between $e_i\in U(P_1)$ and $e_i\in U(P_2)$. If in the merged list $e_i\in U(P_1)$ has an index $j\neq i$, then we need to check if there is an intersection between $e_j\in U(P_1)$ and $e_j\in U(P_2)$. Therefore constructing the intersection requires $O(1)$ time, and $O(n+m)$ work.

3. Every processor will contain an edge and its two vertices of $\Delta(P_{12})$, and will transform them back. Requires $O(1)$ time, and $O(n+m)$ work.

**Theorem 4.3**

Using $n+m$ processors, the algorithms for constructing the intersection, and the common convex hull of two second degree convex polygons requires $O(lg(lg(n+m)))$ time, and $O(n+m)$ work which is optimal.

**4.2 Separation Algorithm.**

Using conclusion 3.1 and the dual data structure it is possible to reporting whether two second degree convex polygons intersect each other.

**Algorithm 4.4 Separation.**

1. Find the intersection $U(P_1)\cap D(P_2)$, and the intersection $D(P_1)\cap U(P_2)$.

2. If the intersection is not empty, then every common point represents a separating line between $P_1$ and $P_2$.

**Complexity of algorithm 4.4**

Sep-1. Using [MEL pp. 90] algorithm we can find the intersection points in $O(lg(lg(n+m)))$ time.

**5 Summary.**

In this article we presented an optimal sequential $O(n+m)$ time, and parallel $O(lg(nm))$ time and $O(n+m)$ work, intersection algorithm for second degree convex polygons. Based on the dual transformation we developed two basic algorithms 4.1 and 4.2 that we can use them for sequential and parallel computation. Using the dual data structure we developed an algorithm that checks sequentially in $O(lg(n))$ time if two second degree convex polygons are separated, and in if they are, the algorithm build a data structure for all the line which separate the polygons. In a separate paper [REI] we show how to extend the algorithms to 3-D.

**References.**


Reif M.N. “A New 3-D Sequential-Parallel Intersection-Separation Algorithm” In preparation.