Università degli Studi di Udine
Dipartimento di Scienze
Matematiche, Informatiche e Fisiche
Corso di Laurea in
Informatica Magistrale

Minimum-Area Rectangle Containing a Set of Points

Computational Geometry

Michele Collevati
collevati.michele@spes.uniud.it
7 March 2020

Introduction

- **Problem**: given a finite set of 2D points, compute the minimum-area rectangle that contains the points
- The rectangle is **not** required to be axis aligned
- The minimum-area rectangle is supported by the **convex hull** of the points
- Convex hull $\equiv$ convex polygon
- Any points interior to the polygon have **no** influence on the bounding rectangle
- **Assumption**: the polygon have no triple of collinear vertices $\longrightarrow$ Simpler implementation
- **Rewriting the problem**: compute the minimum-area rectangle containing a convex polygon of n vertices

Theorem 1

Theorem

Given a rectangle with four points arbitrarily chosen such that no edge contains more than one point, there exists another rectangle such that each edge contains one and only one of these points and the **area** *of the rectangle is* **less** *than that of the given rectangle.*

Proof.

Select a rectangle R with one point on each of its sides such that the sides u_i are each divided into two segments, $u_i = u_{i1} + u_{i2}$, $i = 1, \ldots, 4$. We now form another rectangle R' with sides passing through the same points and such that each side of R' makes an angle α in the counterclockwise sense with the corresponding side of R, as illustrated in Figure 4, where the four points are marked A, B, C, and D.

Proof (contd.)

Let ΔA be the amount by which the area of R exceeds that of R'. Using the notation of Figure 4, we can write:

$$\Delta A = \sum_{i=1}^{4} S_i - \sum_{i=1}^{4} T_i$$

Using some algebra and trigonometry:

$$\Delta A = K_1 + K_2 \left[\sum_{i=1}^{4} u_{i1}^2 - \sum_{i=1}^{4} u_{i2}^2 \right]$$

where K_1 and K_2 are both positive.

Proof (contd.)

We now examine the case where R' is obtained by rotating its sides by an amount α in the clockwise sense, as shown in Figure 5. Now:

$$\Delta A = K_1 - K_2 \left[\sum_{i=1}^{4} u_{i1}^2 - \sum_{i=1}^{4} u_{i2}^2 \right]$$

ΔA is the amount by which the area of R' is less than that of R. It is apparent from the previous two equations that ΔA is positive in at least one of the two cases if not in both.
By letting $u_{11} = u_{42} = 0$, or $u_{11} = u_{42} = u_{22} = u_{31} = 0$, the proof also covers the two special cases in which points overlap (see Figure 6). □

Fig. 4. Formation of rectangle R' by counterclockwise edge rotation.

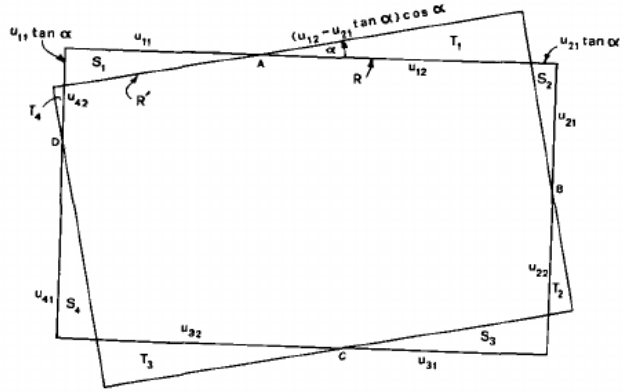


Fig. 5. Formation of rectangle R' by clockwise edge rotation.

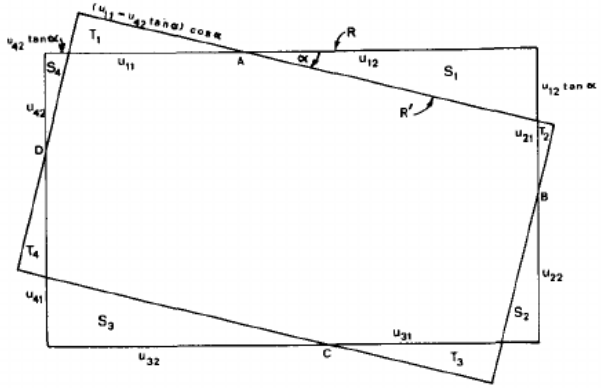
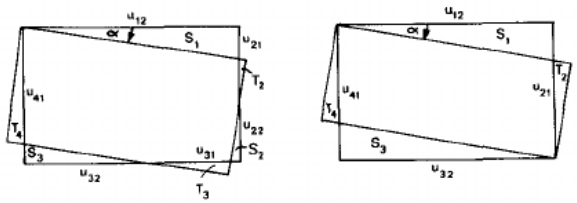


Fig. 6. Two degenerate cases involving overlapping points.



Theorem 2

Theorem

An edge of the minimum-area bounding rectangle **must be coincident** *with some edge of the polygon.*

Proof.

Let us assume that the theorem is false. Since the polygon is convex, only one of its vertices can be on any one side of the rectangle. Since no polygon edge lies on a rectangle side, we can construct another rectangle which will also enclose the polygon, and, by the Theorem 1, will be of lesser area. Hence the given rectangle could not have been of minimum area. □

In some cases, **multiple** edges of the rectangle can coincide with polygon edges

Two algorithms for the problem

- **Exhaustive Search Algorithm**: $\mathcal{O}(n^2)$ time complexity
- **Rotating Calipers Algorithm**: $\mathcal{O}(n)$ time complexity

Exhaustive Search Algorithm

- **Preprocessing**:
  - **Input**: set of m points
  - Computes the convex hull $\longrightarrow$ **New input**: set of n points

- **Algorithm steps**:
  1. Iterates over the edges of the convex polygon
  2. For each edge, computes the smallest bounding rectangle with an edge coincident with the polygon edge
  3. Of all n rectangles, chooses the one with the minimum area

- Details on (2):
  1. Computes the **rectangle axis** directions
  2. **Width** of the rectangle:
     - Project the polygon vertices onto the line of the edge
     - Width = the maximum distance between the projected vertices
  3. **Height** of the rectangle:
     - Project the polygon vertices onto the line perpendicular to the polygon edge
     - Height = the maximum distance between the projected vertices
  4. Computes the **rectangle vertices**

- Code details...

Correctness

Trivial...the algorithm always ends by returning the minimum-area rectangle among all possible smallest bounding rectangle

Complexity

- The asymptotic behavior of the convex hull algorithm depends on $m \longrightarrow$ Potentially m is much larger than n
- The two loops make it clear why the algorithm is $\mathcal{O}(n^2)$ (given the convex hull)
- Small $n \implies$ Viable algorithm in practice

Why improve it?

- For the sake of science
- The minimum-area rectangle is a **subproblem** for computing the minimum-volume box containing a convex polyhedron

It is worthwhile to have a faster algorithm available for the 3D setting

Rotating Calipers Algorithm

The method of **rotating calipers** is an algorithm design technique first used by Michael Shamos in 1978 to generate all antipodal pairs of points on a convex polygon and to compute the diameter of a convex polygon in $\mathcal{O}(n)$ time

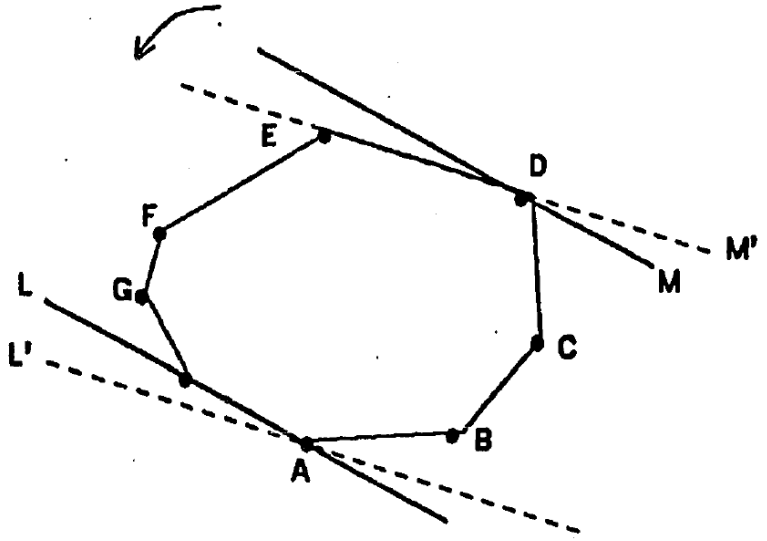


Figure 3.24: Generating antipodal pairs of vertices.

"...A pair of points that does admit parallel supporting lines will be called antipodal. ...The problem is to find them without examining all pairs of points. ..." - from Shamos's PhD thesis

- Godfried Toussaint coined the phrase *rotating calipers* and also demonstrated that the method was applicable in solving **many other** computational geometry **problems**:
  - The Minimum-Area Enclosing Rectangle
  - The Maximum Distance Between Two Convex Polygons
  - The Vector Sum of Two Convex Polygons
  - Merging Convex Hulls
  - Finding Critical Support Lines
- *"The rotating calipers constitutes a powerful, simple and elegant tool that can solve many computational geometric problems efficiently in practice"* - Godfried T. Toussaint
- The animated gif illustrates the idea

Rotating calipers applied to our problem

Algorithm in macro-steps:

1. Starts with a bounding rectangle having an edge coincident with a polygon edge and a supporting set of polygon vertices for the other rectangle edges
2. The rectangle axes are rotated counterclockwise by the **smallest angle** that leads to the rectangle being coincident with another polygon edge
3. The new supporting set of vertices is built from the previous set and from the new polygon edge vertices

Basic Definitions & Notation

- $\boldsymbol{V}_i$ for $0 \leq i < n$: the polygon vertices
- $\langle \boldsymbol{V}_i, \boldsymbol{V}_{i+1} \rangle$: a polygon edge $\longrightarrow$ Abuse of notation: $\langle i, i+1 \rangle$
- Arithmetic on the indices into the polygon vertices is performed modulo n
- $S = \{\boldsymbol{V}_{i_0}, \boldsymbol{V}_{i_1}, \boldsymbol{V}_{i_2}, \boldsymbol{V}_{i_3}\}$: a multiset of the **supporting vertices** for a rectangle. Each vertex supporting an edge of the rectangle $\longrightarrow$ Order: (b)ottom, (r)ight, (t)op, (l)eft
- Why S can be a **multiset**? $\longrightarrow$ One vertex can support two edges
- $I = \{i_0, i_1, i_2, i_3\}$: the **supporting indices**

Typical configuration for a bounding rectangle

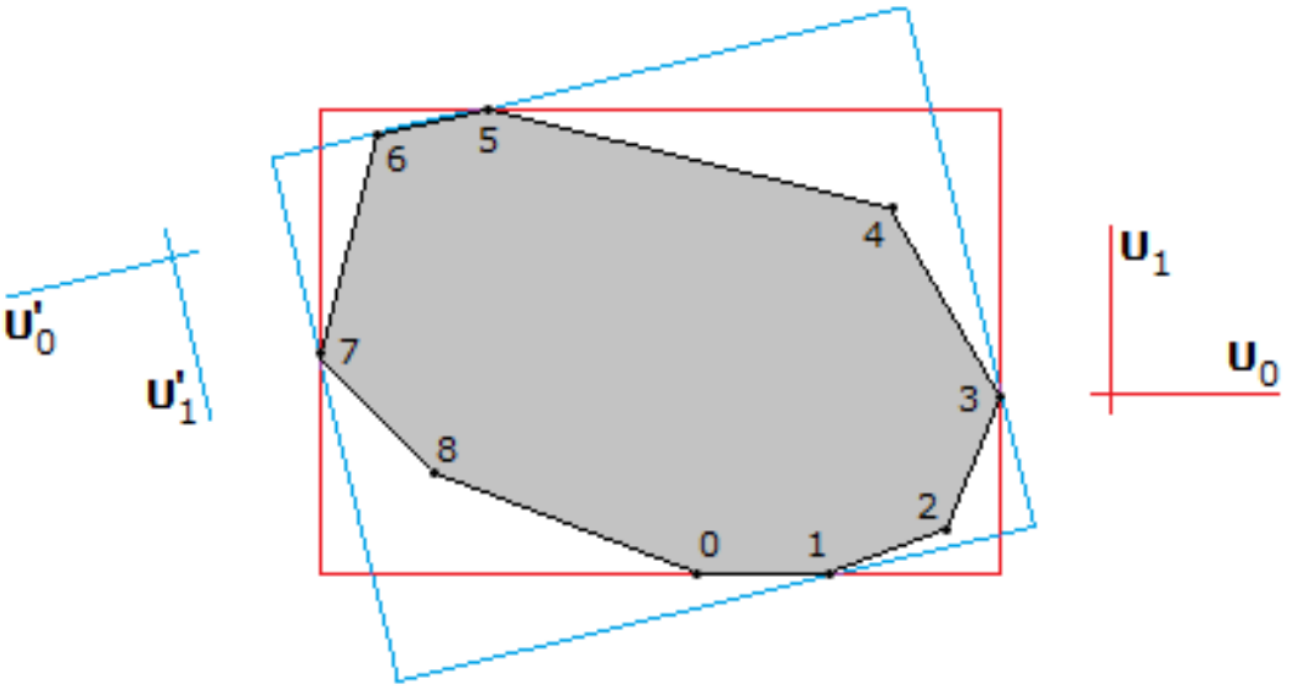
Current rectangle:

- axis direction $\boldsymbol{U}_0$ and $\boldsymbol{U}_1$
- supported by $\langle 0, 1 \rangle$

Next rectangle:

- axis direction $\boldsymbol{U}'_0$ and $\boldsymbol{U}'_1$
- supported by $\langle 5, 6 \rangle$

The **minimum rotation** is determined by $\langle 5, 6 \rangle$



- Supporting vertices:
  - b is supported by $\boldsymbol{V}_1$ (counterclockwise-most vertex)
  - r is supported by $\boldsymbol{V}_3$
  - t is supported by $\boldsymbol{V}_5$
  - l is supported by $\boldsymbol{V}_7$
- Supporting indices: $I = \{1, 3, 5, 7\}$

- Candidate **rotation angles**:
  - $\langle 1, 2 \rangle$ with b
  - $\langle 3, 4 \rangle$ with r
  - $\langle 5, 6 \rangle$ with $t \longrightarrow$ **Minimum**
  - $\langle 7, 8 \rangle$ with l
- Rotates the current rectangle to the next rectangle by the minimum angle
- The next rectangle has b coincident with $\langle 5, 6 \rangle$, but the other polygon edges were not reached by this rotation because their angles are larger than the minimum angle

- New axis directions:
  - $\boldsymbol{U}_0' = (\boldsymbol{V}_6 - \boldsymbol{V}_5)/|\boldsymbol{V}_6 - \boldsymbol{V}_5|$
  - $\boldsymbol{U}_1' = \text{PerpCCW}(\boldsymbol{U}_0')$, where PerpCCW = counterclockwise normal vector operator
- New supporting vertices:
  - b is supported by $\boldsymbol{V}_6$ (counterclockwise-most vertex)
  - r is supported by $\boldsymbol{V}_7$
  - t is supported by $\boldsymbol{V}_1$
  - l is supported by $\boldsymbol{V}_3$
- New supporting indices: $I' = \{6, 7, 1, 3\}$

- A **single** loop is used to visit the minimum-area candidate rectangles
- **Loop invariant**: axis directions $\{\boldsymbol{U}_0, \boldsymbol{U}_1\}$ and ordered supporting indices $I = \{b, r, t, l\}$
- $\langle b-1, b\rangle$ is coincident with the rectangle's bottom edge
- $\boldsymbol{V}_b$ is required to be the **counterclockwise-most** vertex on the bottom edge

In the previous example:

- $5 \in I$ is selected because of the minimum-angle constraint
- The remaining elements of I support the rotated rectangle
  $\Longrightarrow$ They become elements of I'
- $\boldsymbol{V}_6$ successor of $\boldsymbol{V}_5 \implies 6 \in I'$

Computing the initial rectangle

- The initial rectangle has b coincident with $\langle n-1, 0\rangle \longrightarrow$ Avoiding the modulus argument in the loop indexing
- Initial axis directions:
  - $\boldsymbol{U}_0 = (\boldsymbol{V}_0 - \boldsymbol{V}_{n-1})/|\boldsymbol{V}_0 - \boldsymbol{V}_{n-1}|$
  - $\boldsymbol{U}_1 = \text{PerpCCW}(\boldsymbol{U}_0)$

- The polygon vertices are **converted** to the coordinate system with origin $\boldsymbol{V}_0$ and axis directions $\boldsymbol{U}_0$ and $\boldsymbol{U}_1$
- Need to search for the **extreme values** of the converted points
- A polygon edge parallel to one of the axis directions

$$\Downarrow$$

An extreme value generated by two vertices

- Assumption: no triple of collinear vertices

$$\Downarrow$$

Not possible to have three or more vertices attain the extreme value in an axis direction

- An extreme value occurs twice

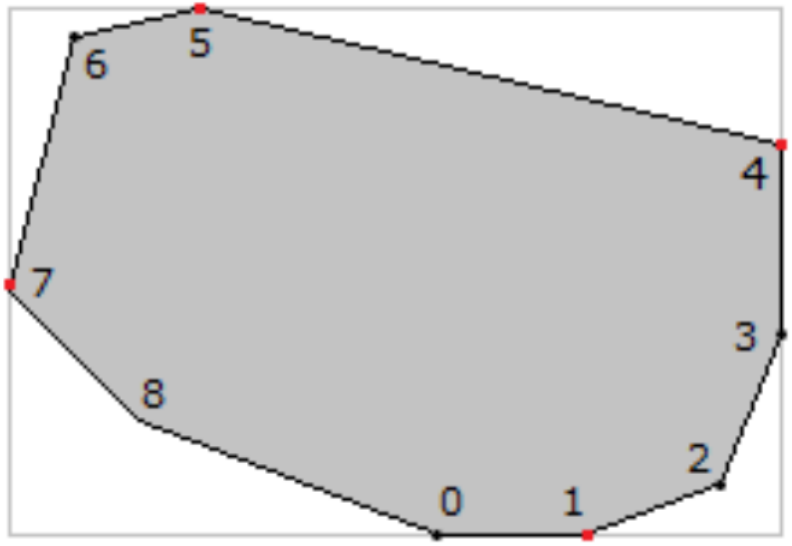
  $\Downarrow$

  The supporting point for the corresponding edge of the rectangle is the **counterclockwise-most** vertex of the edge

  $\downarrow$

  Required because the update step computes rotation angles for edges emanating from the supporting vertex, and those edges must have direction pointing to the interior of the rectangle
- In the previous example, $\langle 0, 1\rangle$ is parallel to $\boldsymbol{U}_0$:
  - $\boldsymbol{V}_1$: support vertex
  - $\langle 1, 2\rangle$: emanating edge
- Code details...

A configuration where two polygon edges are parallel to rectangle edges $\Longrightarrow$ Two extreme values occur twice



- Supporting vertices
- $\langle 0, 1 \rangle$ supports b and $\boldsymbol{V}_1$ is a supporting vertex
- $\langle 3, 4 \rangle$ supports r and $\boldsymbol{V}_4$ is a supporting vertex

Updating the rectangle

- **Problem**: determine the rotation angle to obtain the next polygon edge and its corresponding rectangle
- **Typical configuration**: one coincident edge and three other supporting vertices

...but other configurations can arise that require special handling

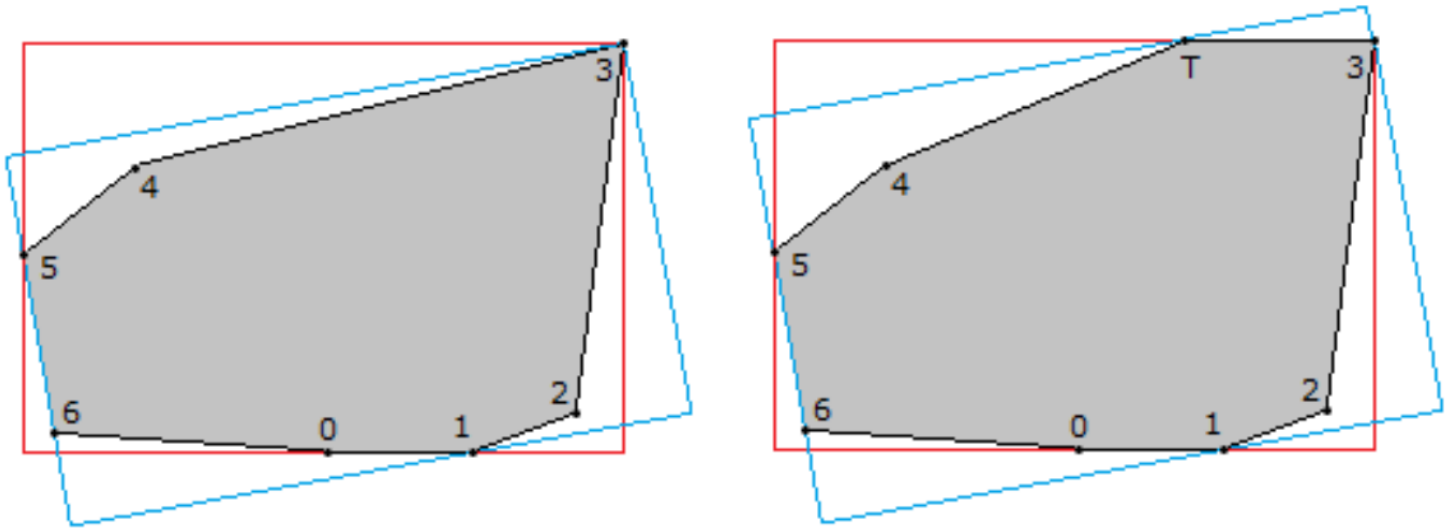
Distinct supporting vertices

- The typical configuration is the simplest to update:
  - **single** coincident edge
  - **unique** supporting vertices
- Let $\langle i_j, i_j + 1 \rangle$ be the **unique** minimum-angle edge, where $j \in \{0, 1, 2, 3\}$
- In the previous example, $\boldsymbol{V}_5$ $(j = 2)$ is the vertex from which the edge emanates

- **New coincident edge**: $\langle i_j, i_j + 1 \rangle$
- The **new supporting indices** I' are obtained from I by replacing i_j with $i_j + 1$ and by cycling so that $i_j + 1$ is the first (b-supporting) vertex: $I' = \{i_j + 1, i_{j+1}, i_{j+2}, i_{j+3}\}$
- The additions in the j-subscripts are computed modulo 4

Duplicate supporting vertices

- Current rectangles
- Next rectangles



(a) (b)

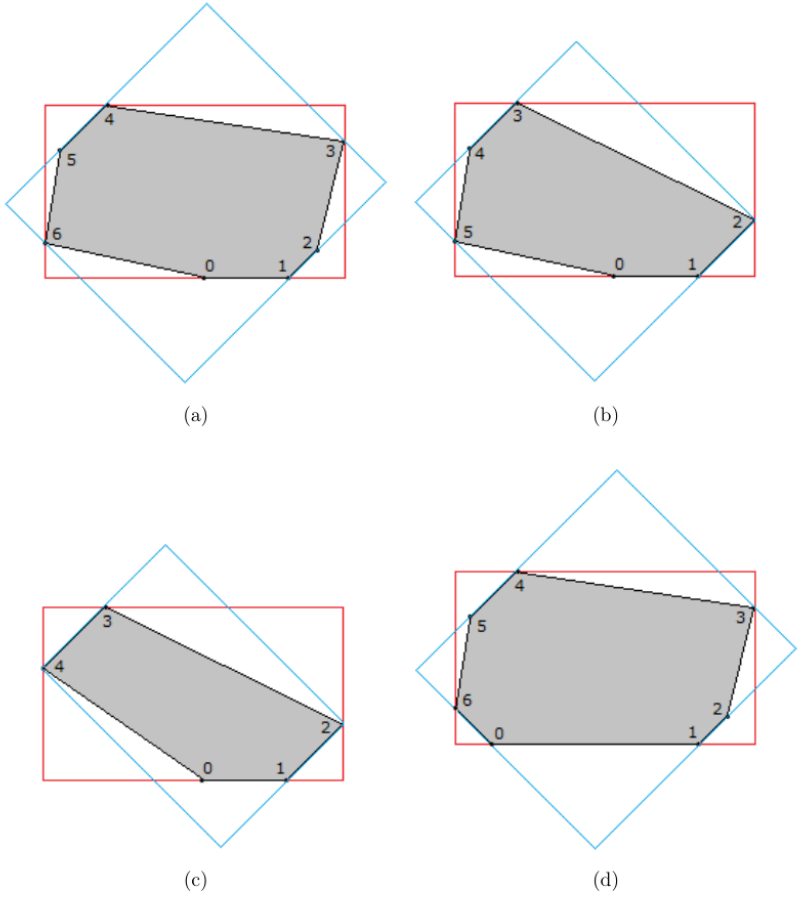
(a) A configuration where a polygon vertex **supports two rectangle edges** $\Longrightarrow$ It is a corner of the rectangle:
- $\boldsymbol{V}_3$ supports r and t
- $I = \{1, 3, 3, 5\}$
- $I' = \{6, 1, 3, 3\}$

(b) The duplicate-vertex configuration can be thought of as the **limiting case** of a distinct-vertex configuration, where $\boldsymbol{V}_T \to \boldsymbol{V}_3$:
- $I = \{1, 3, T, 5\}$
- $I' = \{6, 1, 3, T\}$

Move $\boldsymbol{V}_T$ so that it coincides with $\boldsymbol{V}_3$ $\Longrightarrow$ The current rectangles do not change, but the t of the next rectangle of the distinct-vertex case becomes the t of the next rectangle of the duplicate-vertex case

- The new coincident edge and the new supporting indices are computed as in the distinct case, **except** there is no minimum-angle candidate edge emanating from the first of the duplicate vertices
- In the example (a):
  - Candidate edges: $\langle 1, 2\rangle$, $\langle 3, 4\rangle$ (emanating from the second $\boldsymbol{V}_3$), $\langle 5, 6\rangle$ (minimum-angle)
  - No edge emanating from the first $\boldsymbol{V}_3$ (think of the edge as $\langle 3, 3\rangle$, which is degenerate) $\longrightarrow$ That supporting vertex is **skipped**
  - I' generated from I by replacing $\boldsymbol{V}_5$ by $\boldsymbol{V}_6$, and the vertices are cycled so that $\boldsymbol{V}_6$ occurs first

Multiple polygon edges attain minimum angle



(a/b/c) Two polygon edges attain the minimum angle

(d) Three polygon edges attain the minimum angle

(a)

- $I = \{1, 3, 4, 6\}$ (Current rectangle)
- Two minimum angle edges: $\langle 1, 2\rangle$, $\langle 4, 5\rangle$
- New b: $\langle 1, 2\rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_2$
- New t supported by $\langle 4, 5\rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_5$
- $I' = \{2, 3, 5, 6\}$ (Next rectangle)
- From I to I':
  - $\boldsymbol{V}_1$ replaced by $\boldsymbol{V}_2$
  - $\boldsymbol{V}_4$ replaced by $\boldsymbol{V}_5$
  - 2 already the first in $I' \longrightarrow$ No cycling necessary

(b)

- $l = \{1, 2, 3, 5\}$ (Current rectangle)
- Two minimum angle edges: $\langle 1, 2 \rangle$, $\langle 3, 4 \rangle$
- New b: $\langle 1, 2 \rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_2$
- New t supported by $\langle 3, 4 \rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_4$
- $l' = \{2, 2, 4, 5\}$ (Next rectangle)
- $\boldsymbol{V}_2$ duplicate that supports b and r
- From l to l':
  - $\boldsymbol{V}_1$ replaced by $\boldsymbol{V}_2$
  - $\boldsymbol{V}_3$ replaced by $\boldsymbol{V}_4$
  - 2 already the first in $l' \longrightarrow$ No cycling necessary

(c)

- $l = \{1, 2, 3, 4\}$ (Current rectangle)
- Two minimum angle edges: $\langle 1, 2\rangle$, $\langle 3, 4\rangle$
- New b: $\langle 1, 2\rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_2$
- New t supported by $\langle 3, 4\rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_4$
- $l' = \{2, 2, 4, 4\}$ (Next rectangle)
- $\boldsymbol{V}_2$ duplicate that supports b and r
- $\boldsymbol{V}_4$ duplicate that supports t and l
- From l to l':
  - $\boldsymbol{V}_1$ replaced by $\boldsymbol{V}_2$
  - $\boldsymbol{V}_3$ replaced by $\boldsymbol{V}_4$
  - 2 already the first in $l' \longrightarrow$ No cycling necessary

(d)

- $I = \{1, 3, 4, 6\}$ (Current rectangle)
- Three minimum angle edges: $\langle 1, 2\rangle$, $\langle 4, 5\rangle$, $\langle 6, 0\rangle$
- New b: $\langle 1, 2\rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_2$
- New t supported by $\langle 4, 5\rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_5$
- New l supported by $\langle 6, 0\rangle \longrightarrow$ Supporting vertex: $\boldsymbol{V}_0$
- $I' = \{2, 3, 5, 0\}$ (Next rectangle)
- From I to I':
  - $\boldsymbol{V}_1$ replaced by $\boldsymbol{V}_2$
  - $\boldsymbol{V}_4$ replaced by $\boldsymbol{V}_5$
  - $\boldsymbol{V}_6$ replaced by $\boldsymbol{V}_0$
  - 2 already the first in $I' \longrightarrow$ No cycling necessary

The general update step

Current rectangle:

- b coincident with $\langle i_0 - 1, i_0 \rangle$
- $\boldsymbol{U}_0 = (\boldsymbol{V}_{i_0} - \boldsymbol{V}_{i_0-1})/|\boldsymbol{V}_{i_0} - \boldsymbol{V}_{i_0-1}|$
- $\boldsymbol{U}_1 = \text{PerpCCW}(\boldsymbol{U}_0)$
- $I = \{i_0, i_1, i_2, i_3\}$

Next rectangle:

- b coincident with $\langle i'_0 - 1, i'_0 \rangle$
- $\boldsymbol{U}'_0 = (\boldsymbol{V}_{i'_0} - \boldsymbol{V}_{i'_0-1})/|\boldsymbol{V}_{i'_0} - \boldsymbol{V}_{i'_0-1}|$
- $\boldsymbol{U}'_1 = \text{PerpCCW}(\boldsymbol{U}'_0)$
- $I' = \{i'_0, i'_1, i'_2, i'_3\}$

Problem: determine I' from I

- $M \subseteq \{0, 1, 2, 3\}$: $m \in M$ iff $\langle i_m, i_m + 1 \rangle$ forms the minimum angle with the rectangle edge supported by $\boldsymbol{V}_{i_m}$
- M has multiple elements $\implies$ Minimum angle attained multiple times

- Example 1: $M = \{2\}$
- Example 2(a): $M = \{3\}$
- Example 3(a/b/c): $M = \{0, 2\}$
- Example 3(d): $M = \{0, 2, 3\}$

Steps to generate I' from I (code details...)

1. Processes each $i_k \in I$ for which $i_{k+1} \neq i_k$
2. Computes the angle θ_k formed by $\langle i_k, i_k + 1 \rangle$ and the supported rectangle edge. It is necessary that $\theta_k > 0$
3. Stores the pair (θ_k, k) in a set A sorted on the angle component
4. After processing all elements of I, the first element of A is $(\theta_{k_{min}}, k_{min})$
5. Iterates over A and inserts all the index components k into M for which $\theta_k = \theta_{k_{min}}$
6. For each $m \in M$ replaces i_m by $i_m + 1$. The resulting multiset is J
7. Let m_0 be the smallest element of M. The next rectangle is selected so that its b is supported by $\langle i_{m_0} - 1, i_{m_0} \rangle \implies i'_0 = i_{m_0}$
8. Cycles the elements of J so that i_{m_0} occurs first, which produces the multiset I'

A robust implementation

- The **correctness** of the algorithm depends on:
  - the input polygon being **convex**
  - the **rounding errors** that occur when normalizing the edges (further propagated in the various algebraic operations)
- **First solution**: use exact rational arithmetic

...but normalizing a vector generally cannot be done exactly using rational arithmetic

- **Second solution**: the normalization can be omitted by making some algebraic observations
- **Result**: the minimum-area rectangle can be computed exactly
- An oriented-rectangle representation with unit-length axis directions can be achieved at the very end of the process, incurring floating-point rounding errors only in the conversion of rational values to floating-point values

Avoiding normalization

- Consider the computation of the smallest rectangle for a specified $\langle j_0, j_1 \rangle$
- $\boldsymbol{W}_0 = \boldsymbol{V}_{j_1} - \boldsymbol{V}_{j_0}$ and $\boldsymbol{W}_1 = \text{PerpCCW}(\boldsymbol{W}_0)$: the unnormalized axis directions with $|\boldsymbol{W}_1| = |\boldsymbol{W}_0|$
- $\boldsymbol{U}_i = \boldsymbol{W}_i / |\boldsymbol{W}_0|$ for $i = 0, 1$: the unit-length axis directions
- $\boldsymbol{V}_b, \boldsymbol{V}_r, \boldsymbol{V}_t, \boldsymbol{V}_l$: the support vertices

- The **width** w and **height** h of the rectangle are:

$$w = \boldsymbol{U}_0 \cdot (\boldsymbol{V}_r - \boldsymbol{V}_l), \quad h = \boldsymbol{U}_1 \cdot (\boldsymbol{V}_t - \boldsymbol{V}_b)$$

- The **area** a is:

$$a = w \cdot h = \frac{(\boldsymbol{W}_0 \cdot (\boldsymbol{V}_r - \boldsymbol{V}_l))(\boldsymbol{W}_1 \cdot (\boldsymbol{V}_t - \boldsymbol{V}_b))}{|\boldsymbol{W}_0|^2}$$

Numerator and denominator can be computed **exactly** using rational arithmetic when the input vertices have rational components

- The construction of the support vertices uses comparisons of dot products. For the current r support:

$$\boldsymbol{U}_0 \cdot (\boldsymbol{V}_i - \boldsymbol{V}_{i_1}) > \boldsymbol{U}_0 \cdot (\boldsymbol{V}_{r'} - \boldsymbol{V}_{i_1})$$

  where
  - $\boldsymbol{V}_{i_1}$: the origin of the coordinate system
  - r': the index of the current support vertex
  - i: the index of the vertex under consideration
- The Boolean value of the comparison is **equivalent** to that using **unnormalized** vectors:

$$\boldsymbol{W}_0 \cdot (\boldsymbol{V}_i - \boldsymbol{V}_{i_1}) > \boldsymbol{W}_0 \cdot (\boldsymbol{V}_{r'} - \boldsymbol{V}_{i_1})$$

  The expressions in the comparison can be computed **exactly** using rational arithmetic

Indirect comparisons of angles

- Consider the localization of the minimum-angle edge
- Let $\boldsymbol{D}$ be the unit-length rectangle edge direction (counterclockwise ordering):
  - $\boldsymbol{U}_0$ for b
  - $\boldsymbol{U}_1$ for r
  - $-\boldsymbol{U}_0$ for t
  - $-\boldsymbol{U}_1$ for l
- $\boldsymbol{E} = \boldsymbol{V}_f - \boldsymbol{V}_s$: the vector for an emanating edge from $\boldsymbol{V}_s$ to $\boldsymbol{V}_f$

- The angle θ between $\boldsymbol{D}$ and $\boldsymbol{E}$ is determined by:

$$\cos\theta = \frac{\boldsymbol{D} \cdot \boldsymbol{E}}{|\boldsymbol{E}|}$$

The angle is extracted by applying the inverse cosine function (acos)

  - $\boldsymbol{D}$ obtained by normalizing one of $\boldsymbol{U}_0$ or $\boldsymbol{U}_1$
  - Computation of the length of $\boldsymbol{E}$ $\longrightarrow$ Normalization
  - The inverse cosine function cannot be computed exactly

- $\theta \in (0, \pi/2] \longrightarrow$ **Positive** because the edge is directed toward the rectangle interior
- $\theta = \pi/2 \implies$ The support vertex is a **corner** of the rectangle
- Unless the polygon is already a rectangle, there must be at least one other bounding rectangle edge that has an emanating edge with angle smaller than $\pi/2$

- To avoid the inexact computations to obtain θ, the quantities $|\boldsymbol{U}_0|^2 \sin^2\theta$ can be computed instead, which is a **positive** and **increasing** function of θ
- Sorting of θ and $|\boldsymbol{U}_0|^2 \sin^2\theta$ lead to the **same ordering**
- Using some algebra and trigonometry:

$$|\boldsymbol{U}_0|^2 \sin^2\theta = \frac{(\pm \boldsymbol{W}_i \cdot \mathrm{PerpCW}(\boldsymbol{E}))^2}{|\boldsymbol{E}|^2}$$

The fraction on the right-hand side of the equation can be computed **exactly** using rational arithmetic

Updating the support information

- **Problems**:
  - the polygon edges are not visited sequentially in-order
  - if the polygon has pairs of parallel edge, it is possible that the minimum-area rectangle is computed without visiting all polygon edges
- **Solution**: an array of n Boolean values, one for each counterclockwise-most vertex of a polygon edge, is maintained:
  - the Boolean values are all initially `false`
  - after the smallest rectangle is computed for a $\langle i_0, i_1 \rangle$, the Boolean value at index i_1 is set to `true`
  - when the support information is updated, the Boolean value is tested to see whether the edge has been visited a second time
  - if it has been visited before, the algorithm terminates

Correctness

The rectangle is rotated by the minimum angle that the rectangle edges make with their polygon edges

Every polygon edge generates its candidate rectangle as the rectangle makes a full revolution around the polygon

Complexity

Each candidate rectangle is generated in $\mathcal{O}(1)$ time...there are n

$\mathcal{O}(n)$ time complexity

References

David Eberly.
Minimum-Area Rectangle Containing a Set of Points.
https://www.geometrictools.com/Documentation/MinimumAreaRectangle.pdf.
[Online, last access 09/03/20].

Herbert Freeman and Ruth Shapira.
Determining the minimum-area encasing rectangle for an arbitrary closed curve.
Communications of the ACM, 18(7):409–413, 1975.

Joseph O'Rourke.
Finding minimal enclosing boxes.
International journal of computer & information sciences, 14(3):183–199, 1985.

References (2)


Franco P Preparata and Michael I Shamos.
Computational Geometry: An Introduction.
Springer Science & Business Media, 2012.


Michael Ian Shamos.
Computational Geometry.
PhD thesis, USA, 1978.
AAI7819047.


Godfried Toussaint.
Rotating calipers.
http://www-cgrl.cs.mcgill.ca/~godfried/research/calipers.html.

[Online, last access 07/03/20].

References (3)

Godfried T Toussaint.
Solving geometric problems with the rotating calipers.
In *Proc. IEEE Melecon*, volume 83, page A10, 1983.

Godfried T Toussaint.
The rotating calipers: An efficient, multipurpose, computational tool.
In *The International Conference on Computing Technology and Information Management (ICCTIM)*, page 215. Society of Digital Information and Wireless Communication, 2014.

Wikipedia.
Rotating calipers — Wikipedia, the free encyclopedia.
https://en.wikipedia.org/wiki/Rotating_calipers.
[Online, last access 07/03/20].

Akl–Toussaint heuristic (for convex hull)

- **Throw-away** pre-processing step to improve the performance of convex hull algorithms $\longrightarrow$ If the points are random variables then for a (narrow but common) class of probability density functions this step will make them run in **linear expected time**, even if the worst-case complexity is m^2
- **Idea**: quickly exclude many points that would not be part of the convex hull anyway
- **Algorithm steps**: $\mathcal{O}(\mathrm{m})$ time complexity
  - Find the two points with the lowest and highest x-coordinates
  - Find the two points with the lowest and highest y-coordinates
  - These four points form a convex quadrilateral, and all points that lie in this quadrilateral (except for the four initially chosen vertices) are not part of the convex hull

- The points with smallest and largest sums of x- and y-coordinates as well as those with smallest and largest differences of x- and y-coordinates can be added to the quadrilateral, thus forming an **irregular convex octagon**, whose insides can be safely discarded
- Code details...
- *"A fast convex hull algorithm"* (1978) – S. Akl, G. Toussaint
- *"Akl–Toussaint heuristic"* – Wikipedia