M. Tech. (Computer Science) Dissertation Series

Geometric Shape Recognition for Visually Challenged People

a dissertation submitted in partial fulfillment of the requirement for the M. Tech. (Computer Science) degree of the Indian Statistical Institute

By

Ritankar Mandal

under the supervision of

Prof. Bhargab B. Bhattacharya

INDIAN STATISTICAL INSTITUTE
203, Barackpore Trunk Road
Calcutta - 700 035

CERTIFICATE

This is to certify that the thesis titled **Geometric Shape Recognition for Visually Challenged People** submitted by **Ritankar Mandal**, in partial fulfillment for the award of the degree of **Master of Technology** is a bonafide record of work carried out by him under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Prof. Bhargab B. Bhattacharya
Indian Statistical Institute,
Dated : July, 2013.

ACKNOWLEDGEMENT

I would like to express my gratitude to my supervisor Prof. Bhargab B. Bhattacharya, who gave me the golden opportunity to do this wonderful project on the topic and guided me throughout this work. This project helped me a lot to research in the field of computational geometry and I came to know about so many new things for which I am really thankful to him. Secondly I would also like to thank my parents, family members, professors and friends who helped me a lot in finalizing this project within the limited time frame.

Contents

Chapter 1

Introduction

Our perceptual system has the ability to reconstruct the shape of objects from sparse partial data. The way how the brain does this led the pattern recognition researchers to data condensation problem. However such a task is very difficult to accomplice by a visually challenged person. They don't have the ability to match pairs of points on the same vertical or horizontal line. They use a is a tactile writing system called *Braille system* created by Frenchman Louis Braille. This system is used for books, menus, signs, elevator buttons, and currency. Braille-users can read computer screens and other electronic supports thanks to refreshable braille displays. They can write braille with the original slate and stylus or type it on a braille writer, such as a portable braille note-taker, or on a computer that prints with a braille embosser. This is the way to encode alphabetic symbols. However expressing non-textual information such as maps, paintings, graphs and diagrams to a blind person is a challenging problem. This is done by *Tactile Maps*. These are images that use elevated surfaces so that a visually impaired person can feel them. But this technique is costly. An easier way of doing this is to put some elevated dots along the edges of the diagrams so that they can touch and connect the elevated dots to reconstruct the underlying geometric shape. If the elevated dots are placed close enough along an edge, reconstruction by perception can be made unambiguous. Our problem determines the necessary number of elevated dots and their positions. This leads to a solution for manufacturing cheap Tactile/Braille maps, which can be used for depicting necessary floor-plans in public places or for teaching in classrooms for visually challenged people. We will refer to this problem as an *unlabeled version of connect-the-dots*.

1.1 Problem Definition

Let P be a collection of orthogonal polygons $P_1, P_2, \ldots, P_n$. Our algorithm produces V, an irredundent set of points lying on the boundaries of the polygons that captures the underlying geometric continuity and the neighbourhood relations defining the boundaries of the polygons. In this article by orthogonal polygon, we mean a polygon whose edges are parallel to either *x-axis* or *y-axis*.

1.2 Related Previous works

In 1964 Hugo Steinhaus [1] posed the following problem. Consider a set S of n (≥ 3) points in the plane such that no three of them lie on the same straight line. Is it always possible to find a closed polygon with n non-intersecting sides whose vertices are these n points? Then he proceeded to give a clever proof by induction that this is true. His proof removes an extreme point p of S and, by induction, assumes that the remaining $n-1$ points admit such a polygon. Next, an edge e of this polygon that is completely visible from p is found, p is connected to the endpoints of e and e is removed. A direct implementation of this proof yields an $O(n \log n)$ time algorithm for constructing the required polygon.

Independently, in 1966 Michael Gemignani [2] posed the following similar problem: given n (≥ 3) points in the plane, not all lying on the same line, are they the vertices of a simple closed polygonal chain and, if so, produce a witness, i.e., construct one. Note that the problem posed by Gemignani is more general than the version posed by Steinhaus, since Gemignani assumes only that not all the points are collinear, whereas Steinhaus assumes no three points are collinear. Gemignani's proof yields immediately an algorithm that runs in $O(n \log n)$ time, which is optimal since Shamos [5] proved an $\Omega(n \log n)$ lower bound on this problem.

Since then several mathematicians have provided alternate proofs (Quintas and Supnick [3], and Grnbaum [4]). This is now a well known result in computational geometry. In fact this problem has often been tackled in computational geometry as a stepping stone to solving other problems. For example, in 1972 Ron Graham [6] proposed a simple optimal $O(n \log n)$ time algorithm for computing the convex hull of S by first computing a star-shaped polygonization of S.

1.3 Motivation

1.3.1 Reconstructing Polygons from Scanner Data

Scanners are used in many applications, e.g., to capture objects on a platform, looking down to capture terrain. In addition to point coordinates, different scanners may be able to provide surface labels, normals, or unobstructed segments of scanned rays. The problem of reconstructing a surface from a set of data points has been studied for both theoretical and practical interests. Theoretical solutions can provably reconstruct the original surface when the samples are sufficiently dense relative to local feature size. Various solutions to this problem handle noisy data and often incorporate additional information along with the point coordinates, such as estimated normals. Reconstructing the $2D$ floor-plan of a room from different types of scanned data, specifically whether knowledge about the geometry (monotonicity, orthogonality) or topology (connectedness, genus) of the room allows efficient reconstruction from less dense data.

1.3.2 Robotics

Today there are several tasks which are performed by autonomous robots. How powerful do the sensors and movement capabilities of a robot need to be for a given problem? From a theoretical point of view, trying to answer this question leads to a better insight of both the essential difficulty of the problem at hand and of the inherent power of different sensors. From a practical point of view, investigating the question might enable us to replace sophisticated robot designs with cheap and robust counterparts that can more easily be produced in masses. In military applications, an important problem is to draw a map by a robot in an unknown environment. So, here the problem is how many markers do we need so that a robot which can only sense the nearest marker can walk around the edges of a polygonal map.

Chapter 2

Problem Formulation

A simple polygon is a polygon whose two edges never intersect other than the vertices. Our problem deals with simple orthogonal polygons whose edges are parallel to either horizontal or vertical axises. Suppose C is the given collection of simple orthogonal polygons. *If we erase all the edges of the polygons can we reconstruct C ?*

O'Rourke [7] proved that a collection of non-intersecting orthogonal polygons is uniquely determined by its set of vertices V, and its edges can be reconstructed from V in $O(n \log n)$ time where $|V| = n$. The implication of this theorem is that the edges of such collection is redundant. If there is a way, then there is only one way to connect a set of points to make such a collection of orthogonal polygons. And the proof of this theorem is constructive , leading to an $O(n \log n)$ time algorithm for the reconstruction. An easy to implement algorithm for orthogonal polygon reconstruction in $\mathbb{R}^2$ is as follows :

- In each horizontal level, sort the points in that level, and then pair-up consecutive points starting from the left-most point, and connect each pair by a horizontal edge.
- Do the same for each vertical level.
- Finally report the polygon starting from any point.

This procedure is done in $O(n \log n)$ time by sorting the points horizontally and vertically. The same approach works in $\mathbb{R}^3$ with the same time complexity. Here in each plain, we need to execute the earlier algorithm, and the points in two consecutive horizontal planes are also connected following the same rule.

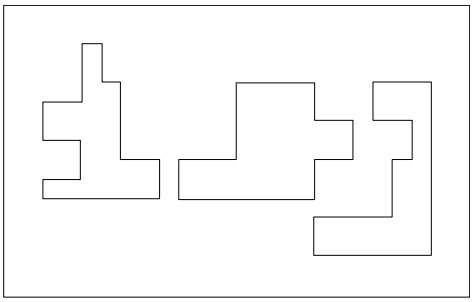


Figure 2.1: Collection of non-intersecting orthogonal polygons

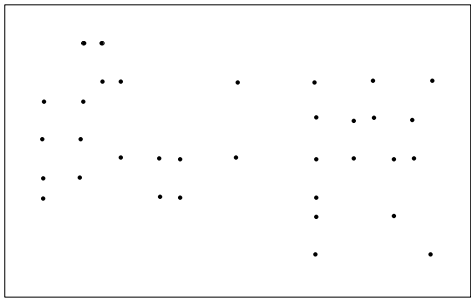


Figure 2.2: Vertices of the polygons required to reconstruct

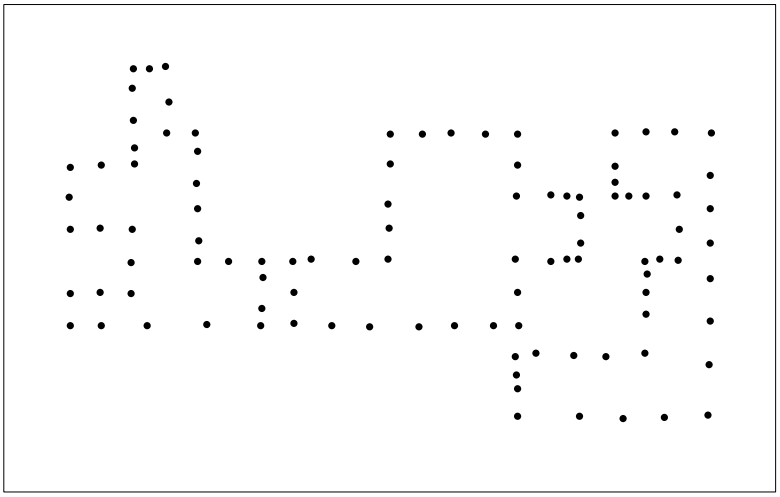


Figure 2.3: Extra points should be added

O'Rourke and Streinu [8] defined a richer combinatorial structure called the vertex-edge visibility graph which includes edge-to-edge visibility. Jackson and Wismath [9] introduced an extended visibility structure, called *stab-graph*, and they proposed an axis-parallel polygon reconstruction scheme using stab-graph.

The visibility graph of a simple polygon traditionally consists of a vertex for each corner of the polygon with an edge joining a pair of vertices if the corresponding corners are internally visible. Polygon reconstruction results attempt to build a polygon consistent with a given visibility graph. The general polygon reconstruction problem was shown to be in PSPACE by Everett [10], who also characterized the visibility graph of spiral polygons. However, a general characterization of visibility graphs of simple polygons has proven

elusive; see for example, the paper by Everett and Corneil [11].

The closest line of research of related problem is of estimating the medial axis from a set of boundary points. Brandt and Algazi [12] proved that Delauny Triangulation of a sufficiently dense set of points on the boundary can reconstruct the boundaries as a subset of edges. Robinson et.al [13] proposed boundary reconstruction by comparing the length of Delauny edges. Various *geometric graphs* has been proposed for this reconstruction scheme. One such example is *nearest neighbour graph* which can be obtained by each point with its nearest neighbour(s) [14]. Another similar graph is *relative neighbourhood graph* which is obtained by joining each pair of relative neighbor(s) [15].

In O'Rourke's algorithms the user has the knowledge that the polygons are orthogonal. But the polygons intersect only at the vertices that means in the corner points. We have taken care of cases where two edges of two different polygon can intersect at any general point and where two edges of two different polygons can share a vertex or an edge.

The reconstruction rule for our scheme is as following :

1 At the time of reconstruction the user only have a set of unlabeled points on a $2D$ plane.

2 At each step the user connect two closest points.

In the following example we can show that only the set of vertices is not enough information to reconstruct the polygons using our *connect-nearest-point* rule.

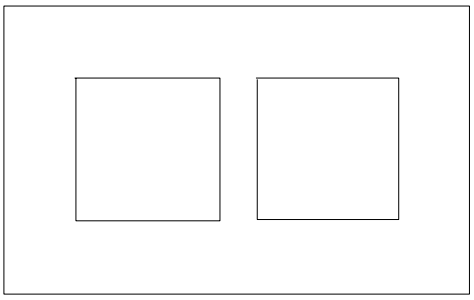


Figure 2.4: Initial collection of the polygon(s)

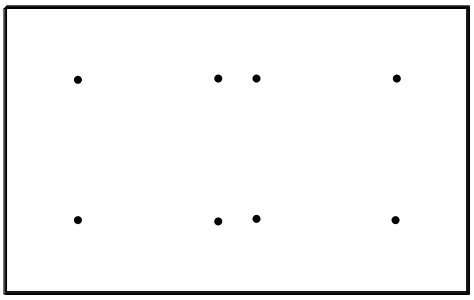


Figure 2.5: Only the vertices of the Initial polygon(s)

We can see given only the information about the vertices a different collection of polygon is generated if we add edges between two closest points. So for correct reconstruction we have to add some extra points on the edges

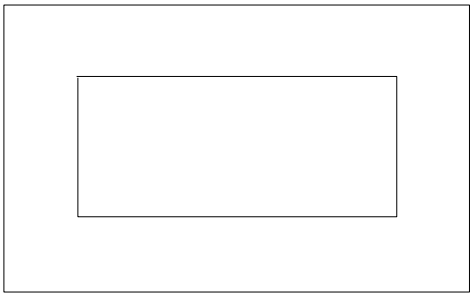


Figure 2.6: Incorrectly Reconstructed Polygon(s)

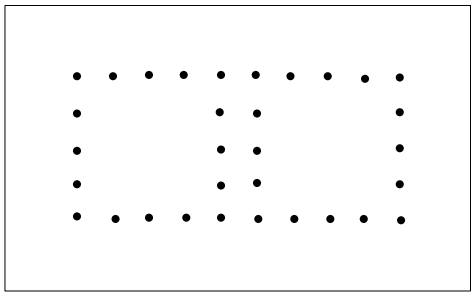


Figure 2.7: Extra points should be added

so that any pair of closest pair and these pairs only define the original edges. So our algorithm takes input P as the collection of orthogonal polygons (may be intersecting) and as output gives V' a set of unordered points. Our aim is to minimize the number of points in V'.

Chapter 3

Two phases of the scheme

Our scheme has two phases to solve this problem. The first phase is the important phase where the extra points are added along the boundaries of the polygons. In the second phase the algorithm mimics the touch-and-sense technique of a visually challenged people and reconstructs the polygons from the given point-set.

3.1 Assumptions

We have some restrictions about the arrangement of the polygons.

1 Among the input points supplied for the reconstruction, the left-most top corner vertex of every polygon is marked differently. The user can identify these special points and start reconstruct each polygon individually at each stage.

2 Each polygon is reconstructed in clockwise manner. In other words, the vertices and edges are reported in *clockwise order* as the output. A direction is necessary to minimize the number of points in V'.

3 At most two edges of two different polygons can intersect at one point.

4 At most two polygons can share a line segment as part of their edge.

5 At most two polygons can share a vertex.

6 ϵ is the minimum distance user can discriminate during reconstruction.

3.2 Different types of arrangement : Allowed and not Allowed

As mentioned in the assumptions all types of arrangements are not allowed in our algorithm. Direction plays a major role in our *connecting nearest point* rule. So any vertex or edge can be shared by at most two different polygons. Here we present some cases of arrangements which are not allowed in our scheme.

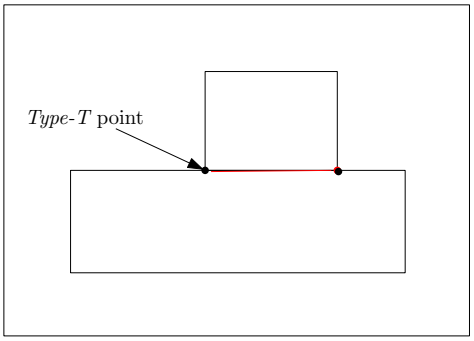


Figure 3.1: Two polygons sharing an edge is allowed

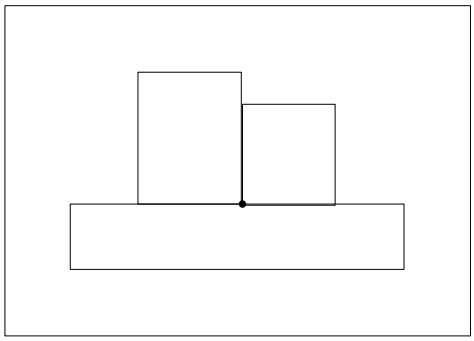


Figure 3.2: Three polygons sharing an edge is **not** allowed

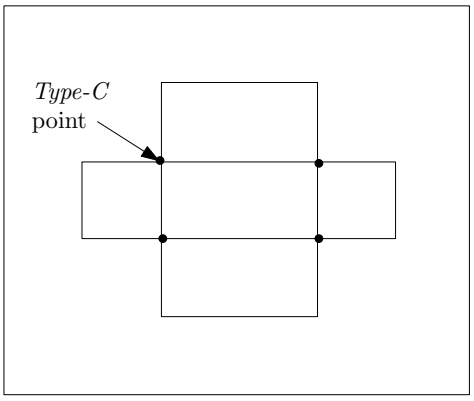


Figure 3.3: Two intersecting polygons are allowed

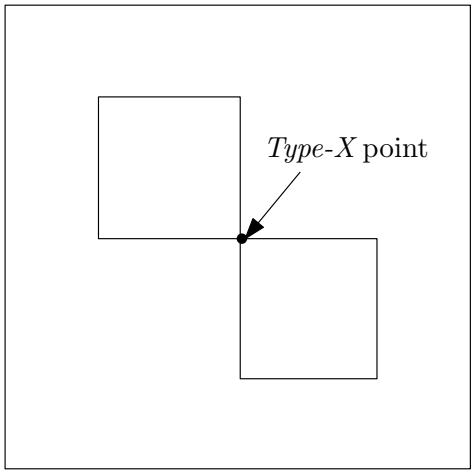


Figure 3.4: Two polygons sharing a vertex is allowed

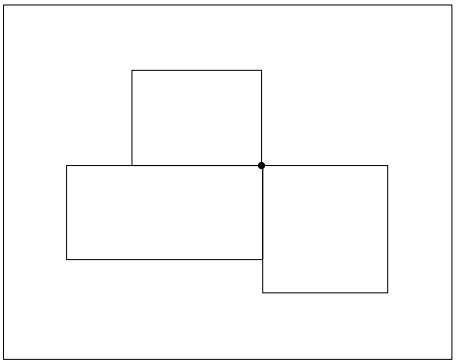


Figure 3.5: Three polygons sharing a vertex is **not** allowed

3.3 Reconstruction phase

In this phase we present the simple algorithm to reconstruct the collection P from the point set V.

3.3.1 Algorithm

1. Start from a left most top corner point which is marked specially.

2. Let we are at point p now. Find out its nearest point r from all the points in the horizontal and vertical direction. Add an edge $\overline{pr}$ and walk to r and do this step again.

3.4 Point Generation phase

As input we take the collection of orthogonal polygons. We walk along the boundaries of the polygons one by one and add extra points where necessary.

3.4.1 Definition

As we are dealing with polygons that may intersect or overlap each other, we need to label the vertices and extra points in different types. The definitions of different types of points are given below.

1. *Type-C* point : The intersecting points of two edges of two different polygons are of these type.

2. *Type-X* point : If two polygons overlap on a corner point but none of its incident edges are overlapped then this corner point is of *Type-X*.

3. *Type-T* point : If two polygons overlap on a corner point and at least one of its incident edges are overlapped then this corner point is of *Type-T*.

4. *Type-N* point : All other points are of this type.

Data Structure

In our algorithm we have the query to find out all the points which have same *x-coordinate value* or same *y-coordinate value* with a given point p from a set of points V. kD-tree is the appropriate Data structure for answering this kind of queries efficiently. The main problem in our algorithm is that V is dynamic in nature, that means new points are constantly being added to the set V.

Algorithm

Input: P, A collection of orthogonal polygons $P_1, P_2, \ldots, P_m$
Output: V, A set of points
for *each vertex* v_i **do**
  if v_i *is corner point of two polygons* P_i *and* P_j **then**
    Add v_i to V with label *type-X*.
  end
  if v_i *is corner point of* P_i *and is on an edge of* P_j **then**
    Add v_i to V with label *type-T*.
  end
  else
    Add v_i to V with label *type-N*.
  end
end
for *each* P_i *and* P_j **do**
  if *any two edges of* P_i *and* P_j *intersect* **then**
    Add that intersecting point to V with label *type-C*.
  end
end

Algorithm 1: Point set generation from a collection of orthogonal polygons

while *no new point is needed to be placed* **do**
  for *each polygon* P_i **do**
    h_i is left-most top corner of P_i. *previous* $\leftarrow h_i$. *current* $\leftarrow h_i$;
    while h_i *is not reached again in this stage* **do**
      Let the next vertex in the polygon is *next*;
      if *current is type-T or type-N* **then**
        S is the set of all the points in V having same x *or* y-*coordinate* value with *current* except the points on the line $\overline{current, previous}$ on the direction of *previous* of *current*;
      end
      if *current is type-X or type-C* **then**
        S is the set of all the points in V having same x *or* y-*coordinate* value with *current*;
      end
      From S find out the nearest point(s) of *current*;
      if *there is only one nearest point and it is next* **then**
        previous $\leftarrow$ *current*. *current* $\leftarrow$ *new*;
      end
      else
        d is distance of *current* and its nearest point(s);
        if p *is of type-N* **then**
          place a new point *new* on $\overline{current, next}$ at a distance $d - \epsilon$ from *current* and add *new* to V;
          previous $\leftarrow$ *current*. *current* $\leftarrow$ *new*;
        end
        if p *is of type-X or type-C* **then**
          place a new point new_1 on $\overline{current, next}$ at a distance $d - \epsilon$ from *current* and another new point new_2 on the other incident edge of *current* at a distance $d - \epsilon$ from *current* and add new_1 and new_2 to V;
          previous $\leftarrow$ *current*. *current* $\leftarrow$ new_1;
        end
        if p *is of type-T* **then**
          place a new point new_1 on $\overline{current, next}$ at a distance $d - 2\epsilon$ from *current* and another new point new_2 on the other incident edge of *current* at a distance $d - \epsilon$ from *current* and add new_1 and new_2 to V;
          previous $\leftarrow$ *current*. *current* $\leftarrow$ new_1;
        end
      end
    end
  end
end

Algorithm 1 Point set generation from a collection of orthogonal polygons (contd.)

Algorithm 2: *

Chapter 4

Upper Bound Construction and Complexity Analysis

The time and space complexity of this procedure clearly depends on the size of V, the output. Here we try to find an upper bound, that means in worst case what is the maximum number of extra points need to be added. Let the maximum size of V is v_{max}, m is the number of polygons and n is the total number of corner points (as well as edges). It is clear that v_{max} does not only depend on n. It also depends on the arrangement or the relative position of the polygons and distance between their vertices and edges.
Let L is the length of the longest edge of the all the polygons. Let us draw a vertical line along each vertical edge and horizontal line along each horizontal edge of the polygons. That creates a grid structure on the collection. In this grid let δ_{min} is the shortest line segment.

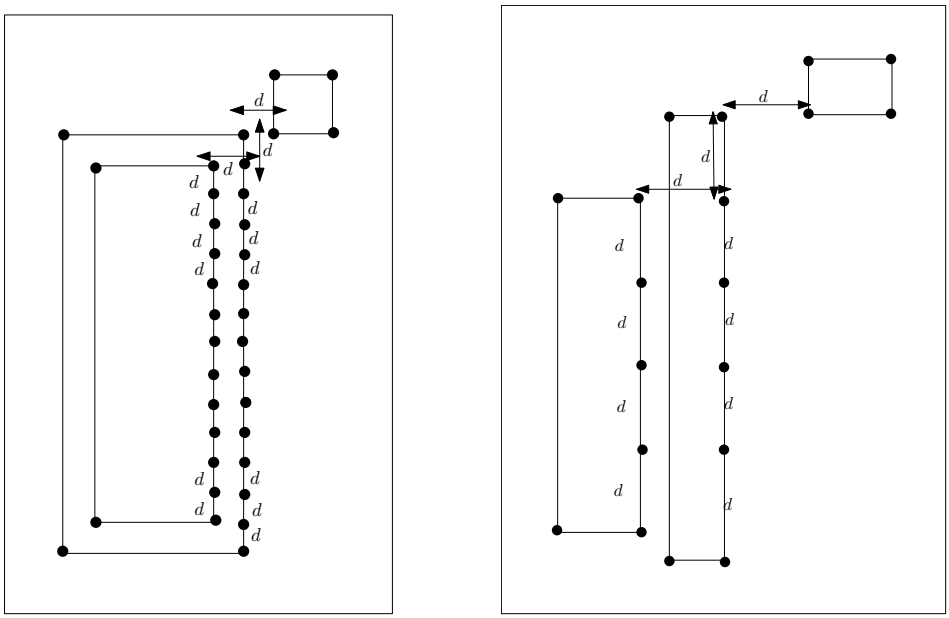
Lemma 4.0.1. *On any edge of the distance between two newly added point is at least $\delta_{min} - \epsilon$.*

Proof. According to our point generation rule if there are two newly added points just one after another on any edge, then the one, q, which comes later while walking in clockwise direction must be generated while preprocessing the earlier one, p. Now as δ_{min} is the shortest line segment in the whole grid structure, so there is no other edge parallel or perpendicular to $\overline{pq}$ within a distance less than δ_{min}. So while p was being preprocessed, there were no point within a distance less than δ_{min}. So the distance between p and q can be at least $\delta_{min} - \epsilon$. □

Theorem 4.0.1. *For any collection of orthogonal polygons P, v_{max} is at most $O(nL/\delta_{min})$.*

Proof. As L is the length of the longest edge of the all the polygons and $\delta_{min} - \epsilon$ is the shortest distance between two newly added point, so the maximum number of points on any edge is $O(L/\delta_{min})$. Total number of edges are n. So maximum number of points is $O(nL/\delta_{min})$. □

Here we present two examples of worst case scenario. In these cases we have shown such arrangements that one edge is bound to have points every δ_{min} distance apart.



Theorem 4.0.2. *The Point Generation Algorithm takes $O(c \log c)$ time to compute the guiding point set of a given collection of orthogonal polygons where c is the size of the output point set.*

Proof. Each point may be preprocessed many times but they are added only once into the output set V. And most importantly whenever a point is proceeded, a new point is added. So number of points added encompasses the whole time complexity of this algorithm. Now each time a new point is added it need to search through the whole point set V stored in a kD-tree, thus taking $O(\log c)$ time in the process. So we can conclude that altogether this algorithm will take $O(c \log c)$ time to compute the guiding point set of a given collection of orthogonal polygons where c is the size of the output point set. □

Chapter 5

Some Experimental Results

If we want to teach the proof of the theorem $(a+b)^2 = a^2 + b^2 + 2ab$ geometrically, this picture-sequence will help us to do that.

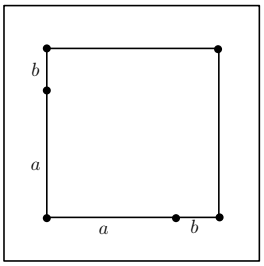


Figure 5.1: $(a+b)^2$

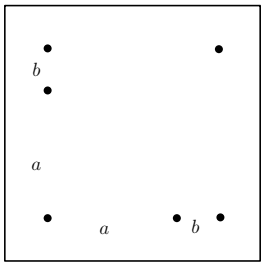


Figure 5.2: $(a+b)^2$

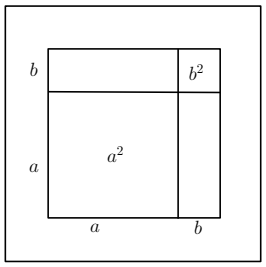


Figure 5.3: $a^2 + b^2 + 2ab$

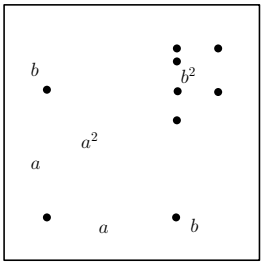


Figure 5.4: $a^2 + b^2$

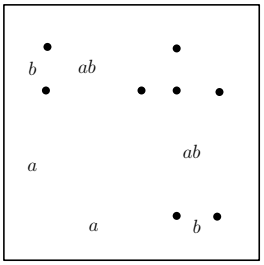


Figure 5.5: $ab + ab$

Chapter 6

Future Works

We have two directions of thoughts for further works related to this problem. One is about handling more complex types of figures. Another is about handling it in more simpler way with respect to lower bound and time complexity.

6.1 Handling general type of polygonal figures

In our problem we have dealt with only orthogonal polygons. Later we would like to work on polygons of general type without any restriction on their arrangement. That would be a more appealing and accurate work for our ultimate target.

6.2 Finding an optimal order of polygons during preprocessing

During the simulation of our algorithm, we noticed that the order in which the polygons are being preprocessed plays a great role in determining the number of points in the output. So in future, we would like to find out an optimal ordering of the polygons that will reduce the number of points generated. In this way we may produce some tighter upper bound on the size of the output.

Bibliography

[1] Hugo Steinhaus, *One Hundred Problems in Elementary Mathematics* Dover Publications Ltd, 1964.

[2] M. Gemignani, *On finite subsets of the plane and simple closed polygonal paths*, Mathematics Magazine, pages 38-41, Jan-Feb. 1966.

[3] L. V. Quintas and F. Supnick *On some properties of shortest Hamiltonian circuits*, American Mathematical Monthly, 72, 977-980. 1965.

[4] B. Grunbaum, *Hamiltonian polygons and polyhedra*, Geombinatorics, 3:83-89. January 1994.

[5] Micheal Shamos *Geometric Complexity*, In Proceedings of the seventh ACM symposium on the Theory of Computing, pages 224-253, 1975.

[6] R. L. Graham *An efficient algorithm for determining the convex hull of a finite planar set*, Information Processing Letters, 1:132-133, 1972.

[7] J. O'Rourke, *Uniqueness of orthogonal connect-the-dots*,Computational Morphology (G. T. Toussaint Ed.), North Holland, 1988.

[8] J. O'Rourke and I. Streinu, *The vertex-edge visibility graph of a polygon*, Computational Geometry, 10 (1998) 105–124.

[9] L. Jackson and S. K. Wismath, *Orthogonal polygon reconstruction from stabbing information*, Comput. Geom., vol. 23, pp. 69-83, 2002.

[10] H. Everett,*Visibility graph recognition*, PhD thesis, University of Toronto, Department of Computer Science, January 1990.

[11] H. Everett and D.G. Corneil, *Negative results on characterizing visibility graphs*, Computational Geometry 5 (1995) 51–63.

[12] J. Brandt and V. R. Algazi, *Continious skeleton computation by voronoi diagram*, Comput. Vision, Graphics Image Process vol. 55 pp. 329-338, 1992

[13] G.P. Robinson, A.C.F. Colchester, L.D. Griffin, and D.J Hawkes, *Integrated skeleton and boundary shape representation for medical image interpretation*, Proc. European Conf. Comput. Vision, pp. 725-729, 1992

[14] Franco P. Preparta and Micheal Ian Shamos, Computational Geometry - An Introduction, Springer-Verlag, 1988.

[15] G.T. Toussaint, *The relative neighbourhood graph of a finite planar set*, Pattern Recognition, vol.12, pp. 261-268, 1980.