PROJECT REPORT ON

GENERATION AND COST OPTIMIZATION OF UNIQUE CLOSED ISOTHETIC POLYGON

SUBMITTED BY

RAHUL CHOWDHURY,

(Exam Roll:111205007)

SHREYASHEE DE

(Exam Roll:111205027)

SIDDHARTH KUMAR

(Exam Roll:111205048)

IN FULFILMENT OF THE MINI PROJECT OF 3RD SEMESTER B.E. IN COMPUTER SCIENCE AND TECHNOLOGY OF

BENGAL ENGINEERING AND SCIENCE UNIVERSITY, SHIBPUR

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
| **1.** | **Problem Statement** | **2** |
| **2.** | **Introduction** | **2** |
| **3.** | **Proposed Method** | **3** |
| **4.** | **Plane Sweep Algorithm** | **3** |
| **5.** | **Challenges faced,and their proposed solutions** | **5** |
| **6.** | **Observation and Result** | **6** |
| **7.** | **Conclusion** | **7** |
| **8.** | **References** | **7** |
|  |  |  |

**1. Problem Statement**

Generate a unique closed Isothetic polygon in a 2-dimensional plane with minimum cost. Cost here refers to the addition and/or deletion of new vertices. Design the algorithm to solve the problem and implement the algorithm in the C programming language.

**2. Introduction**

An Isothetic polygon is a polygon whose alternate sides belong to two parametric families of straight lines which are pencils of lines with centers at two points. In this case the point is at infinity because we are considering only orthogonal polygons.

It is proven that a collection of non-intersecting simple orthogonal polygons is uniquely determined by its vertex set, and reconstructible from this set in *O (n log n)* time. The theorem is also extended to three dimensions: a collection of orthogonal polyhedra is uniquely determined by its set of edges. (Rourke, 1988)

Here we investigate the problem of connecting dots to form orthogonal polygons and with efficient algorithm at the core of the entire process. The conclusion is that there is at most one way to connect a set of dots in a collection of orthogonal polygons.

Consider a collection of non-intersecting simple orthogonal polygons in a plane. Each polygon is composed of alternating sequence of horizontal and vertical edges any two of which meet at most one point, which point is the vertex of the polygon. Each edge contains exactly two vertices and vertices are only permitted at the corners. The polygons may be inside one another, but they should not intersect or share vertices. We call the P the collection of the orthogonal polygons which is made of two sets: the vertices V and the edged E.

Now if we remove V from P we can note that E is uniquely determined by V, that is there is only(at most) one way to connect the set of dots into orthogonal polygons.

The given problem required us to find the intersection points of lines in a 2-D plane. It is solvable by brute force with *O (n^2)* complexity using two nested loops. But since cost optimization was one of the objectives of the problem, we used a special algorithm known as *THE PLANE SWEEP ALGORITHM* to generate the intersection points. The main components of the algorithm are as follows:

1. A vertical sweep-line moving left-to-right.
2. Active segments: horizontal segments that intersect the sweep-line.
3. Sweep event schedule: x-sorted segment endpoints (implemented by a sorted array).
4. Sweep Status: y-ordering of the active horizontal segments maintained

    in a BINARY SEARCH TREE.

**3. Proposed Method:**

* There are a set of points in the plane.Each row and column must have even number of points.If they do not, we must make sure that they have even number of points.This operation should be done in minimum cost.
* The polygon is then generated by joining alternate vertices,both in the rows and columns.
* Some cases of exception may arise,namely intersecting polygons and disjoint polygons.
* The intersecting polygon problem may be removed by finding out the intersection points and then taking steps as necessary.For finding out the intersection points,a special algorithm has been used known as the PLANE SWEEP ALGORITHM ,which makes the complexity of the search O(nlog n) from a brute force approach ,which has a complexity O(n^2).
* The ploygon is then plotted with a graph-plotter.

1. **PLANE SWEEP ALGORITHM:**

We were given a set *V* of *n* points in the plane, and we used the plane sweep technique. We sweep a vertical line across the set from left to right keeping track of the closest pair seen so far. Here we shall describe an O(*n*log*n*) algorithm, and then examine how it works.

We take a vertical line and sweep it across the set of points, keeping track of certain data, and performing certain actions every time a point is encountered during the plane sweep.  
As we sweep the line, we maintain the following data:

* The closest pair among the points encountered
* The distance *d* between the points in the above pair
* All points within a strip of distance *d* to the left of the sweep line These points will be stored in an ordered set *D* (sorted by their *y*-coordinates), which we will implement as a (2,4)-tree .

Now, every time the sweep line encounters a point *p*, we perform the following actions:

1. Remove the points further than *d* to the left of *p* from the ordered set *D* that is storing the points in the strip.
2. Determine the point on the left of *p* that is closest to it
3. If the distance between this point and *p* is less than *d* (the current minimum distance), then update the closest current pair and *d*.

**Therefore we summarize the Plane Sweep Algorithm as follows:**

* Sorting the set according to *x*-coordinates (this is necessary for a plane sweep approach to determine the coordinates of the next point), takes O(*n* log *n*) time.
* Inserting and removing each point once from the ordered set *D* (each insertion takes O(log *n*) time as shown in the section on (2,4)-trees), takes a total of O (*n* log *n*) time.
* Comparing each point to a constant number of its neighbors (which takes O(*n*) time in total).

**7. Challenges faced,and their proposed solutions**

The challenges that we faced during the project were:

1.**Intersecting line segments-**In the generated polygon,various cases of intersecting line segments could be seen.This led to intersecting polygons,which was against our goal.

2.**Disjoint polygons-**Since alternate points were joined,sometimes disjoint polygons could be observed.

The proposed solutions to the aforementioned problems are:

1.We first found out the intersection points.For a large data set,this was very time consuming via a brute-force algorithm-O(n^2) complexity.Thus we had to use an efficient algorithm,namely the Plane sweep algorithm,which had a complexity of O(n log n).By this process,we found out the inrsection points.

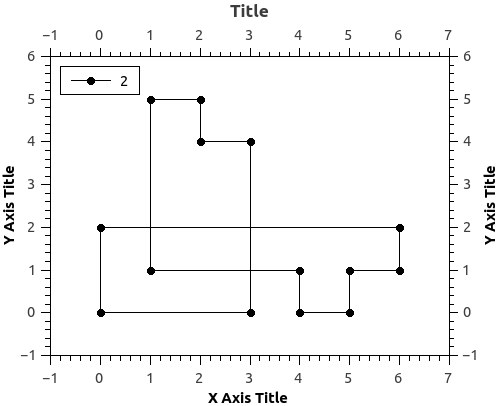
STATUS-Intersection points are being written to a file but removal of these intersection points have not been done yet.

2.Delete a pair of adjacent points in the middle so that we are left with a whole polygon instead of a disjoint one.

STATUS-It has not been implemented yet.

**8. Observation and Results**

The Plane Sweep Algorithm gave us a significant improvement in the time complexity of the problem as compared to the brute force approach. The Plane Sweep Algorithm has a time complexity of *O(n log n)* much effective as compared to the *O(n2)* time of the brute approach. This made the program significantly efficient for large data.



**Plot for a particular set of randomly generated coordinates.(intersection points have not yet been removed)**

**9. Conclusion**

**10. References**

O'Rourke, J., Booth, H., & Washington, R. (1987). Connect-the-Dots: A New Heuristic. *Computer Vision, Graphics and Image Processing* .

Rourke, J. O. (1988). UNIQUENESS OF ORTHOGONAL CONNECT THE DOTS . *Computational Morhphology* , 97-104.

© 2008, 2000, 1997 Springer-Verlag Berlin Heidelberg COMPUTATIONAL GEOMETRY-ALGORITHMS AND APPLICATIONS 3RD EDITION

Professor Andy Mirzaian,University of York INTERSECTIONS(COSC 6114)