5-1-2021

Effect of Boundary Approximation on Visibility

Samridhi Jha

EFFECT OF BOUNDARY APPROXIMATION ON VISIBILITY

By

Samridhi Jha

Bachelor's Degree in Computer Science and Information Technology
Tribhuvan University
2013

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2021

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 23, 2021

This thesis prepared by

Samridhi Jha

entitled

Effect of Boundary Approximation on Visibility

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Laxmi Gewali, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Dean

Kazem Taghva, Ph.D.
Examination Committee Member

Wolfgang Bein, Ph.D.
Examination Committee Member

Henry Selvaraj, Ph.D.
Graduate College Faculty Representative

Abstract

The problem of simplifying a complex shape with simpler ones is an important research area in computer science and engineering. In this thesis we investigate the effect on the visibility properties of polygons when their boundaries are approximated to make them simpler. We present two algorithms for approximating a restricted class of polygons called 1.5 D terrain. We also present experimental investigations on the performance of reviewed and proposed approximation algorithms.

Acknowledgements

I gratefully acknowledge my advisor Dr. Laxmi Gewali for his continuous guidance and support throughout this thesis. His extensive knowledge and continuous guidance helped me a lot during research work and report writing. I would also like to acknowledge my committee members for insightful advice and motivation to enhance my work.

Finally, I would like to show my deepest gratitude to my family for continuous help, love and support.

Samridhi Jha

University of Nevada, Las Vegas

May 2021

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Simplifying a complex polygonal chain with simpler ones has been extensively investigated in many application areas such as cartography, geographical information system (GIS) [HG19], computer graphics, medical imaging, transportation research, data transmission, and pattern recognition [XYW07] [DP73]. One of the main objectives of polygonal curve simplification is to optimize the use of computing resources (memory and processing power) when we need to store and process curves with huge number of vertices. Properly selected subset of vertices could retain some of the properties of the original curve such as curvatures, inflection regions, diameter, and geometric morphological measures. The input polygonal chain Ch_1 is represented by listing the coordinates of its vertices $p_1, p_2, p_3, ..., p_n$ in the order they occur along the boundary. When the first and the last vertices are the same we have the boundary of a simple polygon. If the first and the last vertices are not the same we have an open polygonal chain. Polygonal chains are approximated in term of initially determined error tolerance ϵ. The goal is to approximate the given input chain with a new one Ch_2 which has fewer number of vertices and at the same time it retains its prominent features. Furthermore, each vertex of the approximated chain Ch_2 is required to be within ϵ distance from some vertex of Ch_1. The vertices of the approximated chain are usually a subset of the input chain.

In the process of approximation, various distance measures [vKLW18] [vdKKL$^+$19] can be considered that include (i) Euclidean metric, (ii) Manhattan metric, (iii) Hausdorff metric, and (iv) Frechet metric. A simple explanations of these measures are described in Wikipedia as well. While in some algorithms [DP73] [II86] the distance measure is applied locally in the sense that the distance optimization is done on the selected pieces of the boundaries separately, in other algorithms [vdKKL$^+$19] optimization is applied globally between the original polygon and the simplified version.

When the input polygonal curves are available digitally a whole new set of interesting problems arise for selecting a subset of pixels to construct the polygon edge by edge [PL12], [RR92]. One interesting technique of constructing edges from pixels is to use the notion of **blurred segments** as considered in [PL12]. Blurred segments are constructed by examining the convex hull of the pixels that correspond to the input digitized edge.

Algorithmic techniques based on swarm intelligence have been tried for approximating a polygon with fewer vertices without loosing the dominant features of the input polygon. Swarm intelligence methods used by insects have been tried. In particular, the decentralized collective working behaviour of bees for collecting nectar and accumulating honey is an example of swarm intelligence. An application of bee colony swarm intelligence for constructing an approximate polygonal shape has been reported in [Hua15].

The organization of the thesis is as follow. In Section 2, we present an overview of important algorithms reported in computational geometry literature, dealing with polygonal chain approximation. The reviewed algorithms include Douglas and Peucker [DP73] algorithm and In Section 3, we examine the effect of polygon boundary approximation on the visibility properties of the input polygon. In particular, we show that the widely used polygon simplification algorithms do not retain visibility properties. We then present an algorithm for simplifying boundary so that the approximated polygon tends to retain visibility properties. In our investigation we focus mostly on restricted class of polygons called **1.5 D terrain**. The exact definitions of 1.5 D terrain is given in Chapter 3.

In Chapter 3, we also present a promising simple technique called convex approximation. We use this notion to develop an efficient algorithm for approximating a 1.5 D terrain polygon by fewer number of vertices so that it tends to retain the visibility propertied of the input polygon. We also present time complexity of the proposed convex approximation method.

In Chapter 4, we describe the development of a prototype algorithm for implementing a few algorithms for polygonal boundary approximation. The prototype program is developed in Java programming language. The prototype allows the user to enter input polygon manually by mouse click on the graphic canvas. The entered polygons can be saved for future use in a file. The program lets the user to examine the output generated by the selected algorithm by interactively altering the vertices on the fly. We present the outputs in the form of the approximated polygonal drawings on the canvas supported by the Java graphics. The generated results are also presented in the tabular form, that shows (i) the number of original vertices, (ii) the number of vertices in the approximated

polygons for various values of error tolerance level ϵ.

In Section 5, we discuss scopes for further generalizations and investigations of the proposed problem. In particular we discus how the proposed algorithms for restricted class of polygons (1.5 D terrain sky line) can be adopted for the class of simple polygons. We also highlight the applicability of the proposed techniques for near optimal placement of cellular towers 1.5 D terrain.

Chapter 2

Review of Polygonal Approximation

One of the first algorithmic attempts to approximate a complex polygonal chain $Ch_1 = p_1, p_2, p_3, ..., p_n$ with simpler one with targeted application in cartography was investigated by Douglas and Peucker [DP73], which we refer to in short as *D-P Algorithm*. For a given error tolerance value ϵ, the objective is to approximate Ch_1 by a chain Ch_2 with fewer number of vertices. Additionally, any point on the approximate chain is within ϵ distance from the original chain.

D-P Algorithm takes a polygonal chain Ch_1 and tolerance error value ϵ as input and proceeds to construct approximated chain Ch_2 recursively in top down manner. We can illustrate the main idea behind *D-P Algorithm* by a running an example depicted in Figure 1. The first part (from top) in Figure 1 shows the input polygonal boundary (drawn in solid edges), where the vertices are numbered 1 to 24. The length of edge segment (18,19) is taken as the value of the tolerance error ϵ. D-P algorithm starts with the straight line segment $L1, 24$ connecting vertices 1 to 24 as the starting approximation for input chain Ch_1. If all vertices of Ch_1 are within distance ϵ from the current approximation ($L_{1,24}$), then the required approximation is given by the line segment $L_{1,24}$ itself. In the running example, not all vertices are within distance ϵ from $L_{1,24}$. Vertex number 11 is the one *furthest* from $L_{1,24}$. The DP algorithm proceed now recursively by partitioning the input chain at the furthest vertex into two chains *left chain* $L_{11,24}$ and *right chain* $L_{1,11}$ shown by dashed edges in the first part of Figure 1. The approximations of left part and right part are shown in the second part of Figure 1. After two more round of recursions, all vertices are within ϵ of corresponding approximating line segments as shown in the fourth part of Figure 1.

The bottom most part of the figure (drawn in dashed edges) is the approximation obtained by D-P Algorithm. In this example, the D-P algorithm approximates a 24 vertices chain with a smaller number of chain with 11 vertices. The size of approximated chain depends on the value

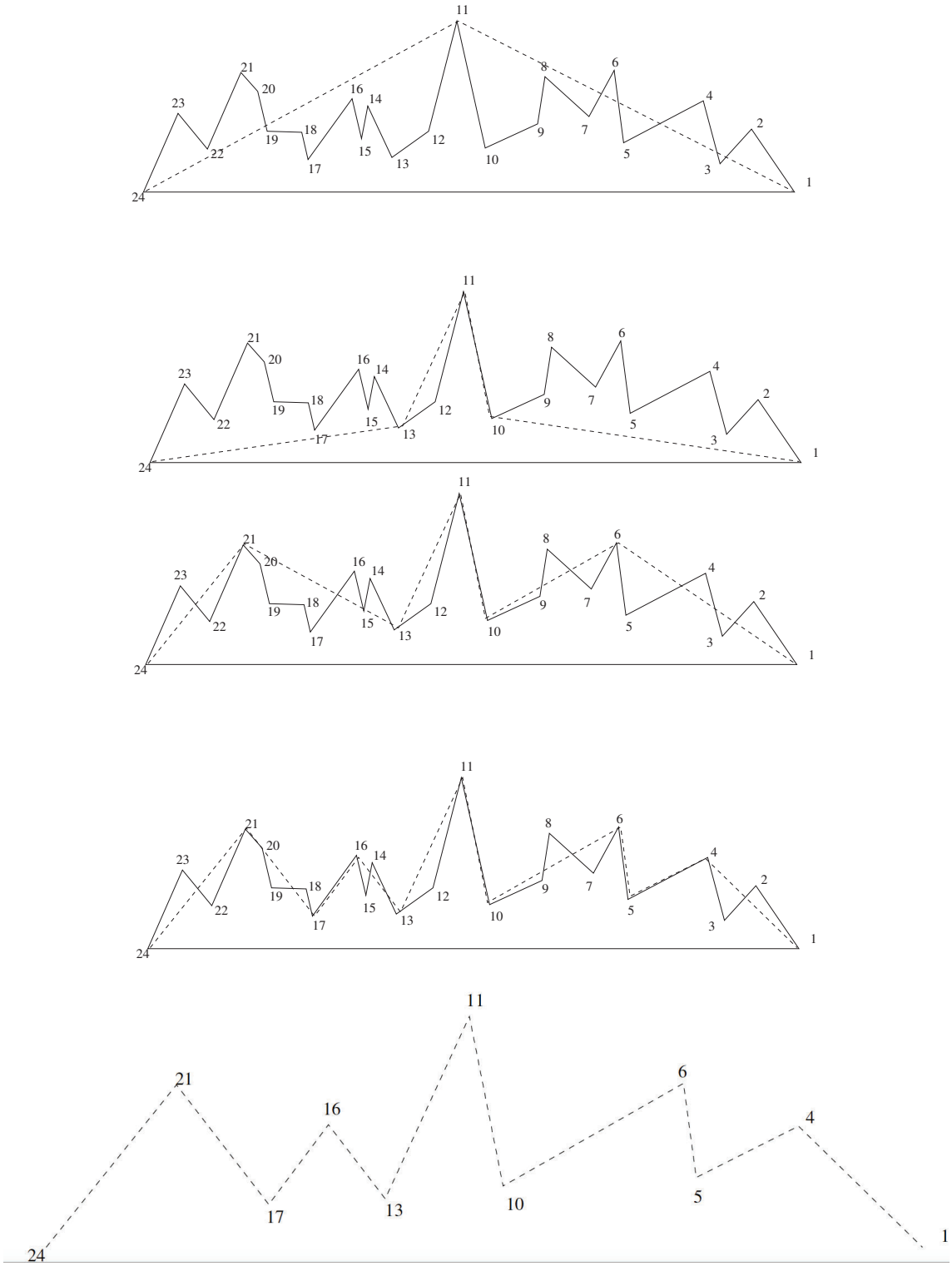

Figure 2.1: Illustrating D-P Algorithm

of ϵ. For large values of ϵ, the size of the approximated solution is small but the quality of the solution degrades. How to choose ϵ, so that the size of the approximation solution is small without compromising the quality of the solution is an important issue of D-P Algorithm.

Another approximation algorithm for polygonal boundary simplification was proposed by *Imai and Iri*, referred as *I-I algorithm* [II86] in short, works iteratively to approximate a complex polygonal chain $Ch_1 = p_1, p_2, p_3, ..., p_n$. This algorithm first specifies a predetermined error value ϵ. The approximated chain Q is such that any original vertex is within distance ϵ from some vertex of Q. The algorithm proceeds by scanning the whole boundary of polygonal chain. I-I algorithm makes use of a rectangle of width ϵ defines as follows.

Definition 2.1 Given an error tolerance error level ϵ, and a polygonal chain $Ch_1 = p_i, p_2, p_3, ..., p_j$, ϵ *-Rectangle* denotes the smallest rectangle of width ϵ that covers the maximum number of nodes in Ch_1.

After determining the sequence of $\epsilon - Rectangles$ that cover all the vertices in the input polygonal chain, *I-I algorithm* proceeds to select two vertices from each rectangle to obtain the approximating line segments. The end points of approximating line segments are also the end points the corresponding sub-chains. It is remarked that the ending vertex and starting vertex corresponding to consecutive $\epsilon - Rectangles$ are the same. An example of the approximation process of *I-I algorithm* is depicted in Figure 2. The top part of the figure shows (i) the input polygonal chain (68 vertices), and (ii) the predetermined error level. The middle part of the figure shows the covering of input chain with 12 $\epsilon - Rectangles$. The bottom part of the figure shows the approximated chain (drawn in dotted line segments) obtained by replacing each $\epsilon - Rectangle$ with corresponding approximating line segment. It is seen that a polygonal chain with 68 vertices is approximated by a simple polygonal chain with 13 vertices.

An improved version of I-I Algorithm is reported in [CC96]. This paper improves the algorithm given in [II86] from $O(n^2 \log n)$ to $O(n^2)$, where n is the number of vertices in the input polygon.

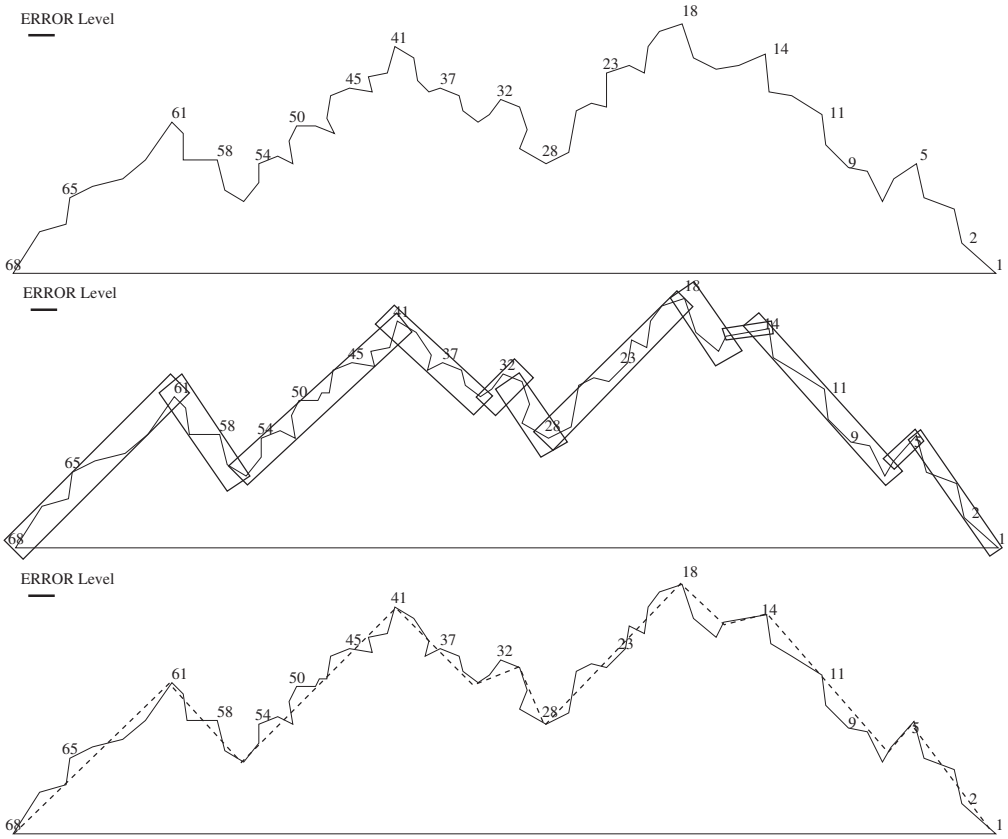

Figure 2.2: Illustrating E-I Algorithm

Chapter 3

Visibility Aware Approximation

In this chapter we present two algorithms for approximating the boundary of a simple polygon by a fewer number of vertices so that the approximated polygon tends to retain the visibility structures of the original polygon. In the first algorithm, we describe how to modify the I-I algorithm [II86] so that approximated algorithm has fewer number of vertices and at the same time it retains most of the visibility properties of the original polygon. In the second algorithm we present, we present a simple linear time algorithm for obtaining a simpler shape with fewer number of vertices in which visibility properties are completely retained. Both algorithms are developed to apply for a restricted class of polygons called 1.5 D terrain.

3.1 Modification of I-I Algorithm

We initiate with a some technical definitions dealing with the visibility of simple polygons [Jos98]. While studying the visibility properties of polygons, the boundary of the input polygon P is treated as an opaque wall. Two points p and q inside P are said to be *visible* if the line segment $\overline{p,q}$ connecting p and q does not intersect with the exterior of the polygon. In Figure 3.1, point C is visible to point D and vice versa, while point C is not visible to point E and vice versa. In term of this notion of visibility, the set of points inside the polygon visible from a given interior or boundary point R is called its *Visibility Polygon* and is denoted by $VP(R)$. In Figure 3, the visibility polygon from vertex R is shown shaded.

The vertices of a visibility polygon can be distinguished into two kinds: (i) *original vertices*, and (ii) *foot vertices* . While the original vertices are the vertices of the input polygon, foot vertices are the vertices formed at the end of the chords bounding the visibility polygon. In the visibility

polygon shown in Figure 3, there are nine vertices, 7 of which are original vertices and the remaining 2 are foot vertices.

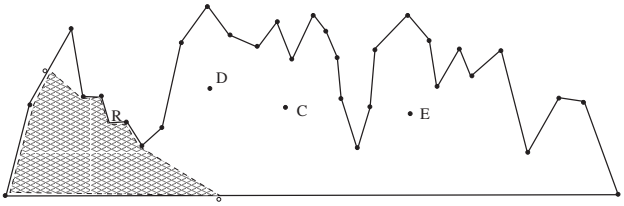

Figure 3.1: Illustrating Visibility inside a polygon

Let us consider the effect on visibility inside a polygon P when its boundary is approximated by a simpler polygon Q. Note that the approximation of the boundary of a polygon is essentially replacing some sub-chains with line segments.

Definition 3.1 Consider an instance where a small portion of the boundary $Ch_{i,j} = p_i, p_{i+1}, p_{i+2}, ..., p_j$ of a polygon P is approximated by line segment $s_{i,j} = \overline{p_i, p_j}$ to obtain an approximated polygon Q. An approximating line segment $s_{i,j} = \overline{p_i, p_j}$ is said to be *visibility preserving* segment if the area of Q visible from the chain $Ch_{i,j}$ is also visible from segment $s_{i,j}$.

The above definition is elaborated in Figure 3.2, where approximations of two chains by corresponding line segments are shown. The approximating segments are drawn in thick lines and the enclosing rectangles used for approximation are drawn in dashed lines. Let us refer the chains that are shown approximated as *left-chain* and *right-chain* with obvious meaning. Similarly, the corresponding approximating line segments are refereed to as *left-E* and *right-E*, respectively. Now observe that the set of points visible from *right-chain* are also visible from the end points of *right-E*. However, not all points visible from *left-chain* are visible from *left-E*. Also observe that the shaded area is visible from the *left-chain* but not from the corresponding approximating segment *left-E*.

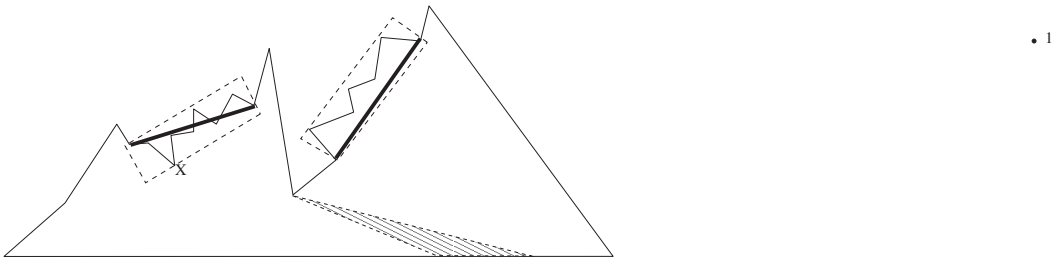

Figure 3.2: Visibility from chain and line segment

This motivates us to define visibility aware polygonal chain approximation problem as follows.

V-Aware Approximation Problem (VAP)

Given: (a) A simple polygon $P = p_0, p_1, p_2, ..., p_{n_1}$

(b) Error level ϵ.

Question: Construct a simpler polygon Q with fewer a number of vertices that approximates P such that (i) all vertices of P are within ϵ distance from some vertex in Q and (ii) the visibility region of Q from any edge of Q is also visible from the corresponding chain in P.

Approximating a polygonal chain with a line segment may lead to arbitrarily large change in visibility. This is illustrated in Figure 3.3. In Figure 3.3, the approximating line segment $\overline{p_4, p_{12}}$ is drawn dashed which replaces the polygonal chain $Ch_1 = p_4, p_5, ..., p_{12}$. Before the approximation, the entire convex component $Cx_1 =< p_1, p_2, p_{14}, p_{15}, ..., p_{16}, p_{17}, p_{18} >$ is visible from vertex p_9. When the chain Ch_1 is replaced by the lone segment $\overline{p_4, p_{12}}$, none of the region inside Cx_1 is visible from the endpoints of $\overline{p_4, p_{12}}$. This is stated in the following observation.

Observation 3.1: Approximating a polygonal chain by a single line segment may lead to arbitrary change in visibility.

Steiner Vertices: For mitigating the situations stated in Observation 3.1, we need to introduce new vertices on the edges of the input polygon which are called *Steiner Vertices*. Steiner vertices are formed by chords of the polygon constructed by connecting reflex vertices and by extending edges incident on reflex vertices. Figure 6, shows the distinction between original vertices (drawn black dots) and Steiner vertices (drawn white dots).

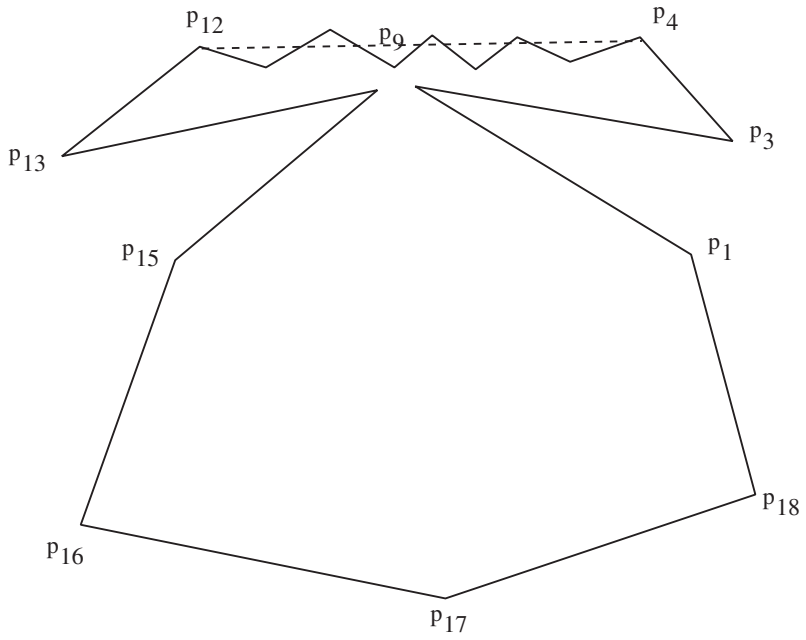

Figure 3.3: Illustrating Observation 3.1

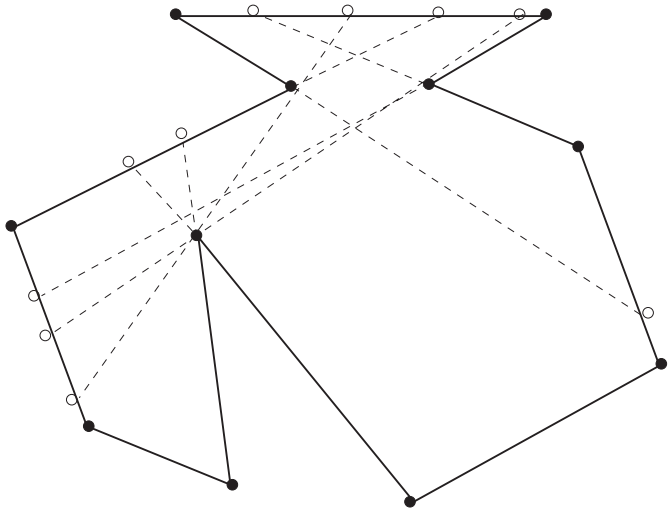

Figure 3.4: Depicting Steiner Vertices

Monotone polygon and terrain modeling

A *monotone polygons* defined as a restricted class of simple polygons have been extensively considered in computational geometry literature [Jos98]. A simple polygon is called *monotone* with respect to a given direction $d^{\rightarrow}$ if its boundary can be partitioned into two chains each of which are monotone with respect to $d^{\rightarrow}$. Furthermore, if one of the chains of a monotone polygon is just one horizontal line segment then it is called a *monotone mountain* [Jos98]. The polygon example shown in Figure 3.5 is a monotone mountain. In recent years, many researchers [Eid02] have used the term *1.5 D terrain* to indicate a monotone mountain. This term is becoming popular due to the fact that the sky line of the terrain has the form that is structurally between a two dimensional terrain and one dimensional terrain.

Visibility graph of a simple polygon A data structure extensively used for investigating visibility properties of a simple polygon is its *visibility graph* [Jos98]. Specifically, the visibility graph $G(V, E)$ in the interior of a polygon is such that V is the set of vertices of the polygon and edges in E consists of all pairs of vertices that are visible to each other. In most applications of visibility graph, *grazing visibility* is allowed which means that the boundary edges of the polygons are also part of the visibility graph. When grazing visibility is not allowed the boundary edges are not included in E. The size of the visibility graph depends on the structural shape of the polygon. For convex polygons the size of the visibility graph is quadratic in the number of vertices. For polygons where large proportion of vertices are visible only with a constant number of other vertices, the size of the visibility graph is much smaller and tends to be linear in the number of of vertices. Visibility graph of polygons can be computed in $|E|$ time by using the algorithm reported in [GM87]. An example of the visibility graph for a 1.5D terrain in shown in Figure 3.5.

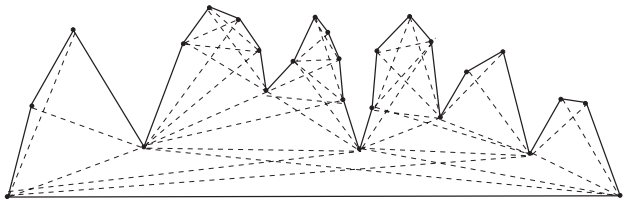

Figure 3.5: Illustrating Visibility Graph

We now have technical ingredients to describe the proposed algorithm. The main part of the algorithm is the process of checking whether a given polygonal sub chain Ch_1 of a simple polygon P is preserving visibility when Ch_1 is approximated by a line segment L. It is noted that the

approximating segment R_i is determined by finding the ϵ **-Rectangle** for Ch_1. To mitigate the problem highlighted in Observation 3.1, additional vertices (Steiner vertices) are identified as shown in Figure 3.4. This way a candidate approximating segment R_i will have needed Steiner vertices. To determine the visibility coverage of candidate segment R_i, we compute visibility polygons from each vertices in R_i and aggregate them to determine the visibility polygon $VisiTot$. The aggregated visibility polygon $VisiTot$ is compared with the combined visibility polygons of the end vertices of R_i. If $VisiTot$ is contained in the combined visibility polygons of end vertices of R_i then R_i is marked as 'visibility retaining'. It is noted that each visibility polygon is computed by navigating the visibility graph $VG(P)$ of polygon P. A formal sketch of the algorithm to determine whether R_i is "visibility retaining" or not is listed as Algorithm 1. Algorithm 1 makes use of function *bool VisEq(Vis1, Vis2)*, which returns *true* if $Vis1$ and $Vis2$ have the same set of vertices. This check is necessary to incorporate the relaxed version of equivalency visibility. In relaxed version of visibility two visibility polygons are equivalent (rather vertex equivalent) if they have the same set of non-Steiner vertices, even though they may have some uncommon areas. Algorithm 2 marks all visibility retaining segments of Q by repeatedly invoking Algorithm 1.

Algorithm 1 bool IsCovered(P, R_i)

1: Let the ordered list of vertices of R_i be $\{q_k, p_{k+1}, ...p_r\}$
2: Compute visibility polygons $Visi(q_k)$, $Visi(q_{k+1})$, ... , $Visi(q_r)$ from vertices q_k, q_{k+1}, ... , q_r, respectively.
3: Set $VisiL = Visi(q_k) \bigcup Visi(q_r)$
4: Set $VisiTot = Visi(q_k)$; Set $e = k + 1$
5: **while** $e \leq r$ **do**
6: $\quad VisiTot = VisiTot$ U $Visi(q_e)$
7: $\quad e = e + 1$
8: **if** (VertexEQ($VisiTot$, $VisiL$)) **then**
9: $\quad$ Return TRUE
10: **else**
11: $\quad$ Return FALSE

Algorithm 2 Marking Visibility Retaining Segments

```
Read (i) Polygon vertices P ={p_0,p_1,.....p_{n-1}}
Read Chain vertices Ch_1 = {p_i,p_{i+1},.....p_j}
Read Error level ε.
Compute approximating polygon Q for input polygon P using I-I algorithm
Let the list of approximating line segments in Q be R_1, R_2, R_3, ..., R_k
for i = 1 to k do
    if IsCoverted(P, R_i) then
        Mark R_i Accepted
```

3.2 Convex strip approximation

In this section we present the development of an algorithm for approximating a 1.5D terrain T by a terrain T' with a fewer number of vertices that preserves visibility relations.

Lemma 3.3 Let $ER_i = \langle v_{i-1}, v_i, v_{i+1} \rangle$ be an ear (i.e. v_i is convex) of a 1.5D terrain T. Then the visibility polygon VP_1 of edge $e_i = \overline{(v_{i-1}, v_{i+1})}$ is identical to the visibility polygon VP_2 of ear ER_i.

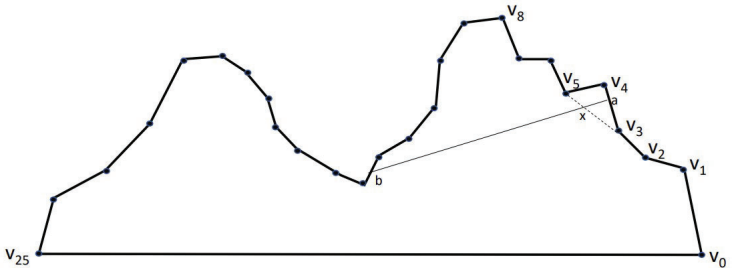

Figure 3.6: Illustrating the proof of Lemma 3.1

Proof: We prove the lemma by arguing that for any visibility ray r_1 originating from ER_i, there is a visibility ray r_2 originating from e_i and coinciding with r_1 and vice versa. Consider a 1.5D terrain T as shown in Figure 3.6. Let us examine the visibility polygon VP_2 produced by an ear $ER_i = \langle v_{i-1}, v_i, v_{i+1} \rangle$. Any visibility ray r_i originating from a point a in ER_i and hitting the boundary of the polygon at a point b must intersect the edge e_i at a point x. Such an intersection point x exists follows from the convexity of ear ER_i. This means for every visibility ray $r_1 = \overrightarrow{(a, b)}$, emanating from a point in ER_i, there is a corresponding visibility ray $r_2 = \overrightarrow{(x, b)}$, emanating from a point x in e_i and coinciding with r_1. This implies that VP_1 is a subset of VP_2. Similar arguments leads to the fact that visibility polygon VP_2 is a subset of VP_1, implying that VP_1 and VP_2 are identical.

■

Lemma 3.3 can be generalized so that the ear is replaced by a convex chain. This is stated in the following lemma.

Lemma 3.4 Let $CH_{i,j} = \langle v_i, v_{i+1}, v_{i+2}...v_j \rangle$ be a convex chain (i.e. all vertices in the chain are convex) of a 1.5D terrain T. Then the visibility polygon VP_1 of edge $g_{i,j} = \overline{(v_i, v_j)}$ is identical to the visibility polygon VP_2 of chain $CH_{i,j}$.

The algorithm we are presenting uses special kind of convex shapes called **podal convex** shapes which is introduced next. A very useful concept in computational geometry is the notion of **anti podal** points [Tou83]. Anti podal points of a convex polygon are the pair of vertices v_i and v_j such that the infinite parallel lines through through v_i and v_j can enclose the polygons.

Definition 3.2 The diameter of a convex polygon is the distance between the antipodal points [Jos98]. A convex polygon that has a boundary edge with length equal to the diameter of the polygon is called **podal convex**. Figure 3.6 shows examples of podal convex polygons. The **podal width** of podal convex polygon is the distance between the diameter edge d_e and the furthest vertex from d_e.

Figure 3.7: Illustrating instances of podal convex polygons

Definition 3.3 For a given error threshold value ϵ, ϵ **-podal components** are the podal convex polygons whose podal width is no more than ϵ.

We can now describe the convex striping algorithm. Given an error threshold level ϵ, and a 1.5D terrain T, the algorithm scans the boundary of T to identify ϵ **-podal polygons** that are disjoint with each other. For identifying ϵ **-podal polygons**, only those polygonal chains are considered that form a convex chain inward. From each such convex chain, the sub-chains that form maximal ϵ **-podal polygons** are identified. The edge that corresponds to the diameter of each ϵ **-podal polygon** is taken as the approximating segment. Figure 3.8 shows convex striping of a 1.5D polygon, where red segments approximate the corresponding convex chains.

For the purpose of clarity of presentation, the alternate sequence of maximal convex chains and

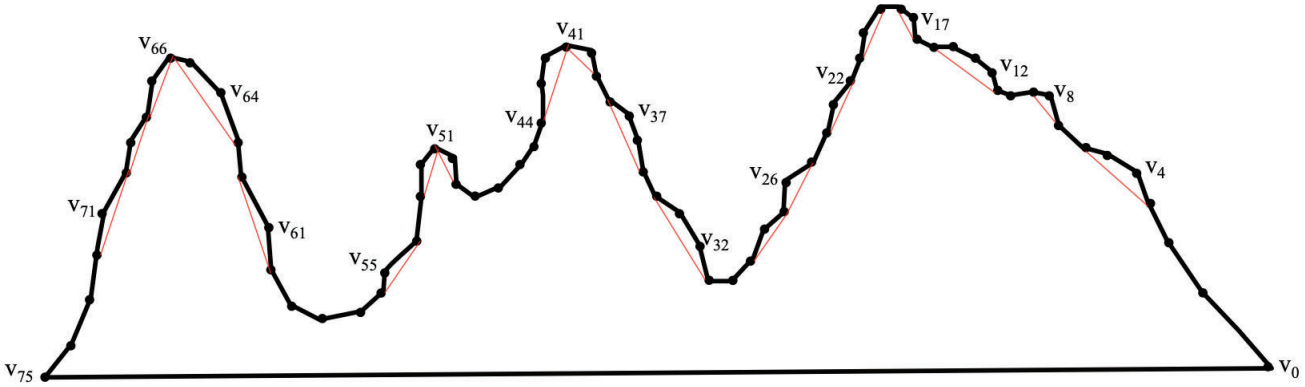

Figure 3.8: Illustrating convex approximation

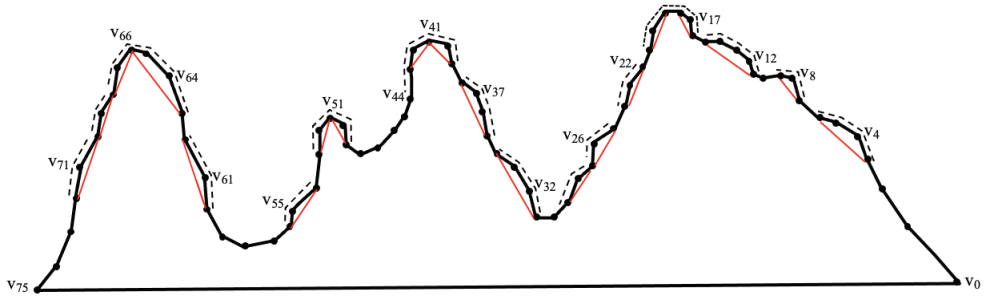

Figure 3.9: Illustrating Alternate Maximal Convex Chains

maximal reflex chains are shown in Figure 3.9, where the maximal convex chains are marked with dashed edges. The parts that are not marked with dashed edges are the maximal reflex chains. We first describe an efficient algorithm for extracting ϵ **-podal** components from a given convex chain $CH = \langle v_i, v_{i+1}, v_{i+2}...v_j \rangle$. The algorithm examines the podal widths of sub-chains $CH_{i,j} = \langle v_i, v_{i+1}, v_{i+2}...v_j \rangle$ of length 3, 4, 5 in turn to find consecutive sub-chains $CH_{i,j}$ and $CH_{i,j+1}$ such that the following condition is satisfied, where $WD(CH_{i,j})$ denotes the podal width of the chain $\langle v_i, v_{i+1}, v_{i+2}...v_j \rangle$

$$WD(CH_{i,j}) < \epsilon \tag{3.1}$$

$$WD(CH_{i,j+1}) > \epsilon \tag{3.2}$$

We refer to the first vertex of the sub-chain v_i as **first leg** and the last vertex v_i as **last leg**. The algorithm initially starts with sub-chain $CH_{i,i+2}$ (the triangle corresponding to the first three vertices) as the candidate ϵ **-podal** component. If the podal width of $CH_{i,j} = CH_{i,i+2}$ is greater than ϵ, then the input chain has no ϵ **-podal** component and the chain can not be approximated. On the other hand, if the width of $CH_{i,i+2}$ is smaller than ϵ then the second leg (j) is incremented and the podal width of $CH_{i,j}$ is checked. This process of moving the second leg continues until the end of the input chain is reached or conditions (3.1) and (3.2) are satisfied. If the end of the input chain is reached then the last ϵ podal component if found. If conditions (3.1) and (3.2) are satisfied without reaching the end of the input chain then a ϵ podal component is found and the algorithm proceeds to identify the next ϵ podal component by resetting the first leg to the running last leg and last leg equal to new first leg plus 2.

A formal description of the algorithm for extracting ϵ **-podal** components for a given input maximal convex chain is listed as Algorithm 3.

The time complexity of Algorithm 3 can be analysed in a straight forward manner. It is noted that the podal width of a convex chain is obviously a uni-modal function, due to its convexity. Hence the podal widths of all convex chains satisfying equations (3.1) and (3.2) can be computed on the fly as the vertices of the chains are scanned. Since each vertex of the input chain is examined no more than 2 times, the total time used by the while loop can be charged to the number of vertices examined. Hence the total time of Algorithm 3 is $O(n)$.

Algorithm 3 Extract ϵ-Podal Components

```
Input: (i) Convex chain ch_1 ={p_1,p_2,.....p_n}
Input: (ii) Tolerance Threshold ε
Output: Disjoint ε-podal components
while (j ≤ n) do
    if (pw(ch_1, i, j) ≥ ε) then
        i++
        j = j+2
    else
        if (j ≥ n) then
            output segment (p_i, p_j)
        else if (pw(ch_1, i, j+1) ≥ ε) then
            output segment (p_i, p_j)
            i = j
            j = i+2
        else
            j++
```

Chapter 4

Experimental investigation

This chapter presents the experiment investigations of (i) one algorithm (II-Algorithm) reviewed in Chapter 2, and (ii) visibility aware approximation algorithm presented in Chapter 3. The purpose of this investigation is to determine the experimental results when polygons are restricted to 1.5D terrain. Quality of approximation is determined by applying the algorithms on several 1.5D terrain polygons of several sizes (number of vertices) ranging from 50 to 250. The program prototype is developed so that the users can generate the data both by mouse-click input or from input file.

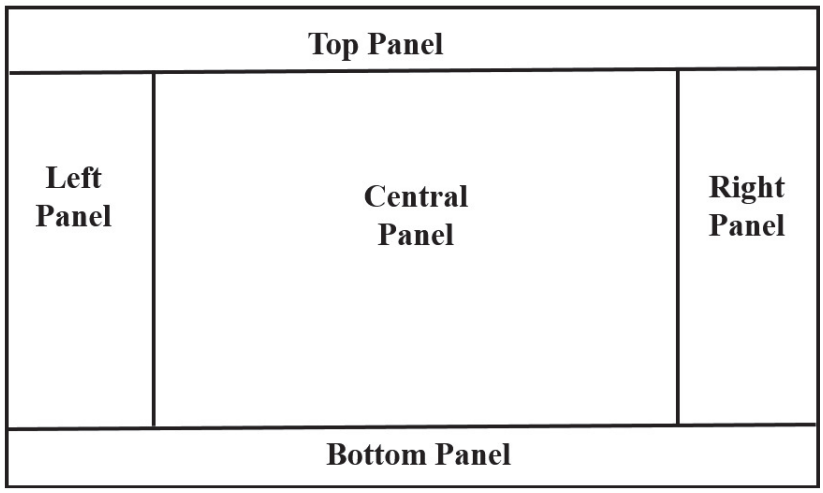

Figure 4.1: The layout structure of the main interface

4.1 Program Prototype

The coding part of this experiment is implemented using the latest version of Java programming language (jdk 1.80). The input/output of the computation on polygons is shown by drawing graphical images and by listing coordinates of vertices. The interface consist of familiar GUI components which are (i) Drawing panels, (ii) Check boxes, (iii) Radio Bottoms, (iv) Drop down bottoms, (v) Text areas, and (vi) Text line.

The front-end graphical interface of the prototype has the structural layout as shown in figure 4.1.

The main-frame GUI contains five components: (i) file read/write drop down list (ii) Computational Panel (iii) Left Panel (iv) Right Panel, and (v) Central Panel.

4.1.1 Panels

Seven panels are used as containers for the various GUI components. The purpose and contents of each of these panels are:

File Read/Write drop down menu : The file menu bar in the main frame has two options in a drop-down menu. The two options are: Read File and Save File. These options helps the user to open/read the previously saved data file, save the current data in a new file and so on. The functionalities of read/save sections are lisred in the following table.

S.N.	**File Read/Write Menu**	**Function**
1	Read/Write File	Read/Write and Open a previously saved terrain vertices set
2	Save	Save the current terrain vertices to currently open file or the new file

Table 4.1: File Read/Write Menu

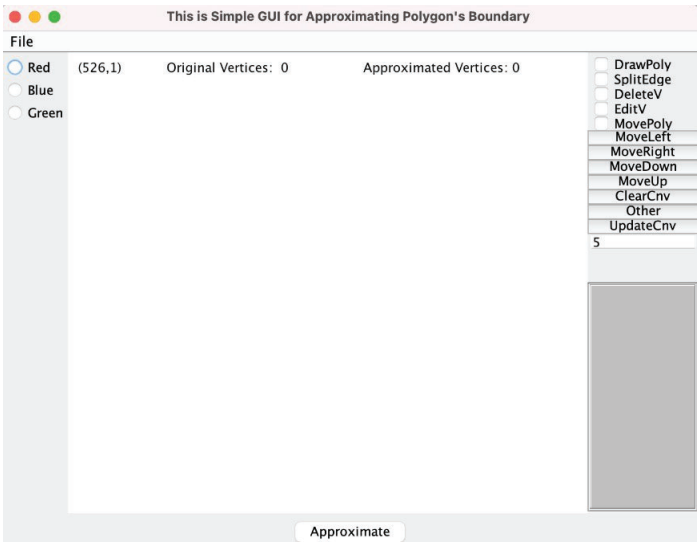

Figure 4.2: The Front-End Frame that greets the user

Computational Panel : This panel is used to display input polygon as a graphic drawing. It is also used to display both the input polygon and the approximated one.

Right Panel : The right panel has different sub Panels. The right top sub-panel contains the general control buttons for polygon drawing which include (i) DrawPoly (ii) Split Edge (iii) DeleteV (iv) EditV, and (v) MovePoly. The right middle Panel contains general control buttons for drawing polygons. They are (i) MoveLeft (ii)MoveRight (iii) MoveDown (iv) MoveUp (v) ClearCnv (vi) Other (vii) UpdateCnv. It also contains text-input to display the error value. Finally, right bottom sub-panel consist of text area which shows the coordinates of terrain vertices in editable form.

Left Panel : The left panel displays "Red", "Blue" and "Green" options for coordinates of terrain vertices on the scrollable canvas.

Central Panel : The central panel is used to draw polygon and perform the computation on the polygon. There is a button labelled "Approximate" which is used to invoke the method that approximates the input polygon.

4.1.2 Control Buttons

The GUI interface has different buttons whose functionalities are briefly specified in the following table:

S.N.	Control Buttons	Description
1	MoveLeft	Move the polygon to left side of the canvas (Central panel)
2	MoveRight	Move the polygon to right side of the canvas (Central panel)
3	MoveDown	Move the polygon to down side of the canvas (Central panel)
4	MoveUp	Move the polygon to up side of the canvas (Central Panel)
5	ClearCnv	clears the canvas (Erases polygon drawn on central panel)
6	Other	
7	UpdateCnv	save polygon with updated vertices.
8	Approximate	approximate the visibility of the vertices on the polygon.

Table 4.2: Functionality of Control Buttons

4.1.3 Check Boxes

S.N.	Check-Boxes	Function
1	DrawPoly	Enable vertex drawing with mouse click.
2	SplitEdge	Enable inserting vertices between two other vertices on the polygon.
3	DeleteV	Enable deleting of node coordinates.
4	EditV	Enable editing of node coordinates.
5	MovePoly	Enable block movement of all drawn object in Central Panel.

Table 4.3: Functionality of Check-Boxes

4.1.4 Text Area

S.N.	Text-Area	Function
1	Vertices Coordinates	displays the coordinates of a polygon vertices in the text area.

Table 4.4: Functionality of Text-Area

4.1.5 Text Input

S.N.	Text-Input	Function
1	Error Value(ϵ)	Used for setting error level to approximate minimum number of vertices on the polygon.

Table 4.5: Functionality of Text-input

4.2 Generated results

We generated many 1.5D polygons manually by editing the vertices assisted by the graphical interface. Th convex stripping algorithm was executed on these polygons for several values of error tolerance. The error tolerance was measured in term of pixel sizes. Some of the results are shown in the table below. In the table the size of both the original polygons and the approximated polygons are shown for error tolerance values 1, 2, 3, 4, 5, 6 pixel sizes.

Approximated Vertices						
Number of Vertices	**ϵ=6**	**ϵ=5**	**ϵ=4**	**ϵ=3**	**ϵ=2**	**ϵ=1**
50	19	21	26	26	31	32
75	22	25	30	32	43	49
100	28	34	38	44	54	62
125	34	41	48	54	62	77
150	42	47	53	60	69	88
250	56	61	68	78	90	114

Table 4.6: Generated Results

Snap shot of graphical outputs are shown in the following figures.

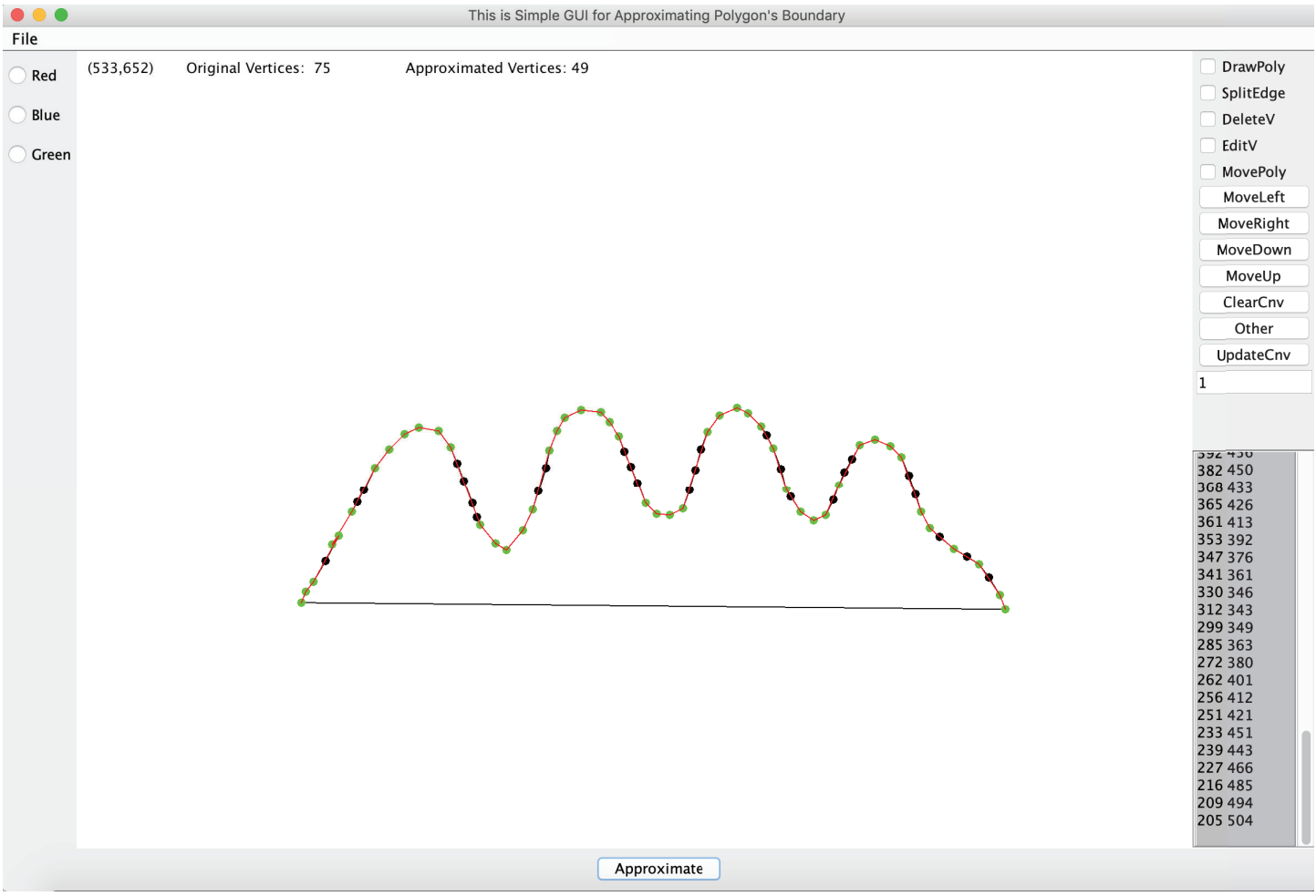

Figure 4.3: Graphical output example 1

Chapter 5

Discussion

We presented approaches for developing algorithms for approximating the boundary of a simple input polygon P with another simpler one Q with a fewer number of vertices so that Q satisfies two conditions: (i) each vertex of Q is within ϵ distance from some vertex in P, and (ii) most of the structural visibility of P are retained in Q. One of the proposed algorithms is based on modifying $I-I$ algorithm [II86] by comparing visibility of approximating line segment and the corresponding chain. The second approximation algorithm we proposed guarantees the retention of the visibility properties of the original polygon. This is done by stripping podal convex parts. Preliminary investigation of the proposed algorithms shows that visibility preservation in the approximated polygon is significantly effective when the input polygon is restricted to $1.5D$ terrain. Detail experimental investigation of the proposed algorithm are ongoing for other restricted class of polygons.

For modeling visibility we have adopted the traditional convention that two points are mutually visible if the line segment connecting them does not intersect with the boundary of the polygon, no matter how long is the length of the line segment. This is a kind of oversimplification for real application. For example, if application is in cellular communication, direct visibility between two points is limited by distance. It would be interesting to apply our proposed algorithm under limited visibility model.

For experimental investigation we generated 1.5D polygons manually. It would be appropriate to generate 1.5D input polygons randomly. Generating polygon randomly is an interesting problem in itself. A few papers have been reported [ZSSM96] that deal with random generation of polygons. For applications of our interest, we need to find effective ways of random generation of 1.5D polygons. One approach is to start with a guiding strip as adopted in [**?**] and test our proposed algorithms by randomly generating several 1.5D polygons of different sizes.

We performed experimental investigation of proposed algorithm on polygon of sizes no more than 400 vertices. If the number of vertices is larger than 1000 the polygon does not fit in the canvas used in the prototype program. We did not have time to put scrolling mechanism in our Java implementation. It would be good to incorporate scrolling mechanism in the Java Frame containing the canvas so that large size polygons can be generated and viewed.

The proposed algorithms are based on visibility in the interior of the 1.5P polygons. Visibility can be similarly defined in the external region of the polygon. A very interesting problem would be to look for the effect of boundary approximation on the placement of vertical tower on the surface of 1.5D terrain.

Bibliography

[CC96] W. S. Chan and F. Chin. Approximation of polygonnal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry and Applications*, 6:59–77, 1996.

[DP73] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:112–122, 1973.

[Eid02] Stephan Eidenbenz. Approximation algorithms for terrain guarding. *Information Processing Letters*, 82(2):99–105, 2002.

[GM87] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 11–19, Oct 1987.

[HG19] Müslüm Hacar and Türkay Gökgöz. A new, score-based multi-stage matching approach for road network conflation in different road patterns. *ISPRS Int. J. Geo Inf.*, 8(2):81, 2019.

[Hua15] Shu-Chien Huang. Polygonal approximation using an artificial bee colony algorithm. *Mathematical*, 2015.

[II86] Hiroshi Imai and Masao Iri. Computational-geometric methods for polygonal approximations of a curve. *Comput. Vis. Graph. Image Process.*, 36(1):31–41, 1986.

[Jos98] O'Rourke Joseph. *Computational Geometry in C*. Cambridge university press, 2 edition, 1998.

[PL12] D. Prasad and M. Leung. Polygonal representation of digital curves. 2012.

[RR92] B.K. Ray and K.S. Ray. An algorithm for detection of dominant points and polygonal approximation of digitzed curves. 13:849–856, 1992.

[Tou83] Godfried Toussaint. Solving geometric problems with the rotating calipers, 1983.

[vdKKL+19] Mees van de Kerkhof, Irina Kostitsyna, Maarten Löffler, Majid Mirzanezhad, and Carola Wenk. Global Curve Simplification. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages

67:1–67:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[vKLW18] Marc J. van Kreveld, Maarten Löffler, and Lionov Wiratma. On optimal polyline simplification using the hausdorff and fréchet distance. *CoRR*, abs/1803.03550, 2018.

[XYW07] Z. Xie, Z. Ye, and L. Wu. Research on building polygon map generalization algorithm. In *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, volume 3, pages 786–791, 2007.

[ZSSM96] Chong Zhu, Gopalakrishnan Sundaram, Jack Snoeyink, and Joseph S.B. Mitchell. Generating random polygons with given vertices. *Computational Geometry*, 6(5):277 – 290, 1996. Sixth Canadian Conference on Computational Geometry.

Curriculum Vitae

Graduate College
University of Nevada, Las Vegas

Samridhi Jha
samridhijha@gmail.com

Degrees:

Master of Science in Computer Science 2021
University of Nevada Las Vegas

Thesis Title: Effect of Boundary Approximation on Visibility

Thesis Examination Committee:

Chairperson, Dr. Laxmi Gewali, Ph.D.
Committee Member, Dr. Kazem Taghva, Ph.D.
Committee Member, Dr.Wolfgang Bein, Ph.D.
Graduate Faculty Representative, Dr. Henry Selvaraj, Ph.D.