8-2011

Roaming region for Delaunay triangulation

Romas James Hada
University of Nevada, Las Vegas

ROAMING REGION FOR DELAUNAY TRIANGULATION

by

Romas James Hada

Bachelor of Computer Engineering, I.O.E., Pulchowk Campus

Tribhuvan University, Nepal

2007

A thesis submitted in partial fulfillment

of the requirements for the

Master of Science Degree in Computer Science

School of Computer Science

Howard R. Hughes College of Engineering

Graduate College

University of Nevada, Las Vegas

August 2011

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

THE GRADUATE COLLEGE

We recommend the thesis prepared under our supervision by

Romas James Hada

entitled

Roaming Region for Delaunay Triangulation

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
School of Computer Science

Laxmi P. Gewali, Committee Chair

Ajoy K. Datta, Committee Member

John T. Minor, Committee Member

Rama Venkat, Graduate College Representative

Ronald Smith, Ph. D., Vice President for Research and Graduate Studies and Dean of the Graduate College

August 2011

ABSTRACT

Roaming Region for Delaunay Triangulation

by

Romas James Hada

Dr. Laxmi Gewali, Examination Committee Chair

Professor of Computer Science

University of Nevada, Las Vegas

Delaunay graphs have been used in CAD/CAM, sensor network and geographic information systems. We investigate the reliability properties of nodes in Delaunay graphs. For measuring the reliability we formulate the concept of *roaming-region* for nodes. A node v with large roaming-region $r(v)$ such that v is positioned near the center of $r(v)$ is identified as a reliable node. We develop algorithms for constructing roaming-regions and present an implementation of the proposed algorithm in the Java programming language.

ACKNOWLEDGEMENTS

I would like to express my inmost gratitude to my supervisor, Dr. Laxmi Gewali for his invaluable support, encouragement and guidance from the initial to the final step of the thesis work. With his patience and generosity, he made me understand the thesis work and guided me towards the way to become a good researcher and taught me a unique way to learn innovative ideas throughout my academic period.

I would like to thank Dr. Ajoy K. Datta for his academic guidance for the last two years. I would also like to thank Dr. John T. Minor and Dr. Rama Venkat for serving as committee members.

Finally and most importantly, I would like to express my deepest gratitude to my beloved Kristina Shrestha for her encouragement, love and support without whom this thesis wouldn't have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

A network or graph consists of a set of nodes V and a set of edges E. An edge $e \in E$ connects two nodes in V. Such a network is usually denoted as G(V,E). The term vertex is also used to indicate node. Similarly, the term link is used to indicate edge. A class of simple networks used extensively in sensor networks and geographical information systems is the planar network. It is noted that a network is called planar if it can be drawn in the plane without intersecting edges. Delaunay triangulation, relative neighborhood graph and Gabrial graph are examples of widely used planar network. One of the main reasons for the popularity of planar graphs in application areas is the fact that the size of a planar graph (number of edges) is not large. In fact, in a planar graph the number of vertices and the number of edges are linearly related. Furthermore, the data structure for representing planar graphs is much simpler and can be updated quickly.

In this thesis we consider the reliability properties of planar network when nodes of the networks are allowed to change slightly in their neighborhood. Broadly speaking, a node in a network is called reliable if the connectivity of the network does not change if the node moves slightly from the initial position. In particular, we investigate the reliability properties of nodes in a Delaunay triangulation. In Chapter 2, we review properties and algorithms for Delaunay triangulation and related structures.

In chapter 3, we present the main contribution of the thesis. We first formulate the notion of *roaming-region* for a node of Delaunay triangulation. We show that as long as a node remains within its roaming-region, the underlying Delaunay network does not change. We present an $O(n^2)$ time algorithm for computing the roaming-region of a node in Delaunay triangulations.

In chapter 4, we consider the implementation of our proposed algorithm for computing roaming-regions for Delaunay nodes. The implementation is done in the Java programming language with user friendly graphical interfaces. Users can generate De-

launay network interactively by specifying node positions. The implementation also allows the construction of the Delaunay network for randomly generated nodes. The interface can be used to identify the roaming-regions for candidate nodes specified by the user.

Finally, in chapter 5, we discuss the scope and extension of the problem of computing roaming-regions for Delaunay and related networks.

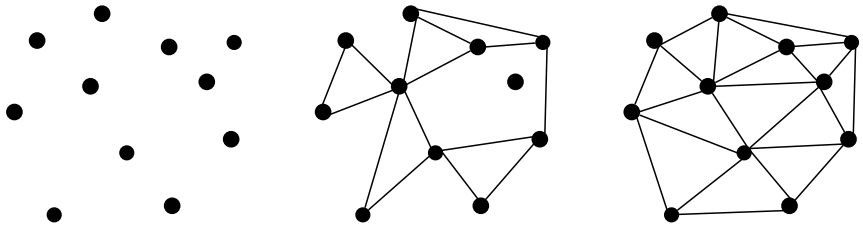
CHAPTER 2

REVIEW OF TRIANGULATED NETWORK AND NODE RELOCATION

In this chapter we present a brief review of algorithms for generating triangulated networks in two dimensions. We also review algorithms for relocating nodes in sensor networks.

2.1 Triangulated Network

Triangulation of a set of point sites S = $\{p_0, p_1, p_2, \ldots, p_{n-1}\}$ is the construction of the maximum number of non-intersecting triangles with vertices in S. Triangulations could be partial or maximal as illustrated in Figure 2.1. The term *triangulation* is

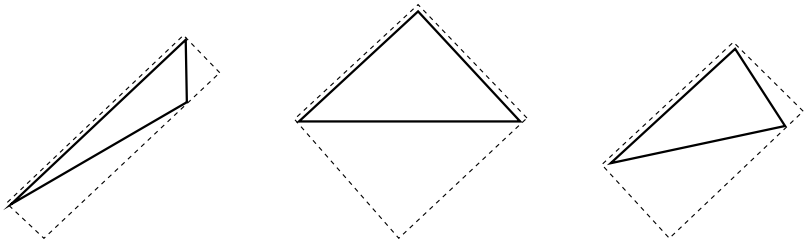


(a) point sites S (b) Partial triangulation (c) Maximal triangulation

Figure 2.1: Illustrating partial and maximal triangulation

generally understood to mean maximal triangulation. The triangulation problem has been investigated and used in many branches of science and engineering that include surveying, cartography, robotics, and geographic information systems [8].

After the advent of computational geometry in mid 1970's, there was a flurry of research activities dealing with the development of efficient algorithms for triangulation [8]. In this review, we mostly consider the algorithms and data structures for generating triangulation. It is noted that a given set of point sites can be triangulated in exponentially many ways [8]. The triangles in a triangulation can be distinguished as *fat, medium* and *thin or skinny*. These notions can be better clarified in terms of the *aspect ratio* of a triangle as follows. Consider the smallest rectangle enclosing a triangle. Let w and h ($w <= h$) denote the width and height of the smallest enclosing rectangle. The ratio w/h is called the aspect-ratio of the triangle. These notions are



(a) Thin/Skinny (b) Fat (c) Medium

Figure 2.2: Aspect ratio and quality triangles

illustrated in Figure 2.2.

Most researchers are interested in a triangulation where the number of *skinny* triangles is reduced. This is illustrated in Figure 2.3. In Figure 2.3, the triangulation on the right has reduced number of skinny triangles compared to the one on the left. A triangulation with fewer number of skinny triangles is called a good quality triangulation. In Figure 2.2, the leftmost triangle has low aspect ratio and is a thin triangle. Similarly the middle one is a fat triangle with aspect-ratio close to one.

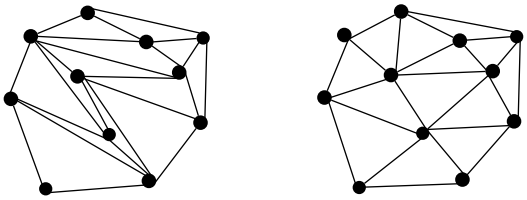


Figure 2.3: Triangulation with dense and sparse skinny triangles

2.2 Delaunay Triangulation

A triangulation with the most interesting properties is the Delauay triangulation. A Delaunay Triangulation is the dual of the Voronoi diagram [5, 8]. An interesting property of Delaunay triangulation is the fact that it maximizes the smallest angle of triangulation [4]. There is a direct characterization of Delaunay triangulation in terms of *in-circle* property, i.e. two points b and c are the end points of a Delaunay edge if and only if there is a circle through b and c that passes through no other point sites and contains no sites in its interior. Consequently, all triangles in the Delaunay triangulation for a given point site will have empty circumscribed circles. That is, no point sites lie in the interior of any triangle's circum-circle. It is always unique

as long as no four points in the given point sites are co-circular. If more than three point-sites are co-circular then there will not be unique Delaunay triangles.

In Figure 2.4, the circle through point b and c contains no other point sites inside it. Hence, it is a delaunay edge for the given point sites. Similarly for point sites *a, b, c* and *d* as shown on the left of Figure 2.4, the segment connecting point sites *c* and *a* cannot be a delaunay edge as the circle passing through point sites *c, d* and *a* is not an empty circle as it encloses a point sites *b* inside it. Hence, it is not a delaunay edge.

As Delaunay triangulation maximizes the smallest angle, it is geometrically nice and, in general, pleasing to the eye. Delaunay triangulations have a number of interesting properties other than the empty circle property which are briefly described next.

convex-hull : The exterior face of the Delaunay triangulation is the convex-hull of the point set.

Closest pair property: The closest pair of point sites are neighbours in the Delaunay triangulation. The circle having these two sites as its diameter cannot contain any other sites, and so is an empty circle.

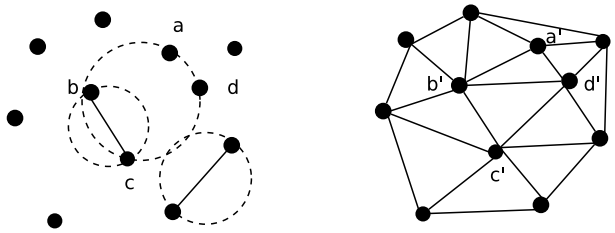


Figure 2.4: Delaunay edges shown on left and corresponding Delaunay triangulation of given point sites on right

2.3 3-D convex-hull and Delaunay Triangulations

There is a very nice relationship between Delaunay triangulation and convex-hull in three dimensions. Consider a set of points in 2-D plane. The image of these points can be projected on the surface of a 3-D paraboloid as shown in Figure 2.5. The details of the construction of paraboloid are found in [8]. The convex-hull of

the lifted points on the surface of the paraboloid gives the connectivity relationships between point sites in 2-D.

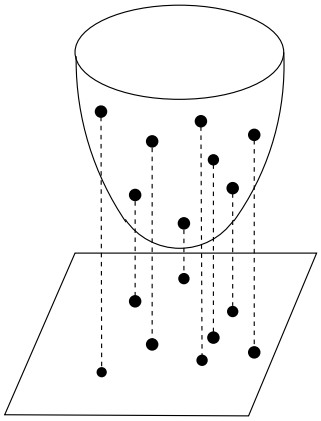


Figure 2.5: Projection of paraboloid

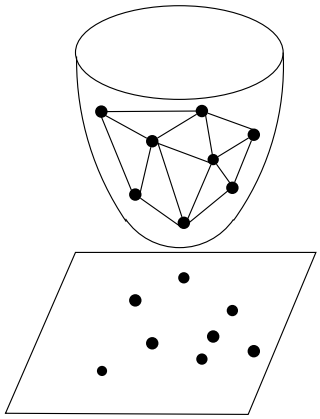


Figure 2.6: Computing convex-hull

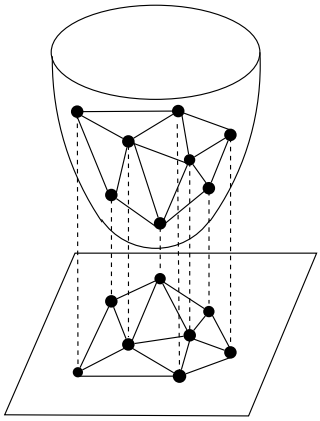


Figure 2.7: A Delaunay triangulation and its lifting to a paraboloid

The convex-hull of lifted points in 3-D is shown in Figure 2.6. Now, if these edges in 3-D convex-hull are projected back to 2-D plane, we get the Delaunay triangulation

as shown in Figure 2.7. This remarkable relationship between 3-D convex-hull and 2-D Delaunay triangulation was discovered by Brown [1]. In fact, this relationship has been generalised between m and m + 1 dimension in Eucledian space.

2.4 Algorithms for Node Relocation

An interesting problem of network design is the reconstruction of the network when nodes are allowed to change their position. Not much research work has been reported on the change of network when node position varies. Some recent work on node relocation have been considered by investigators in sensor network community. For example, Coskun [3] has investigated connected cover problem for sensor networks when some sensor nodes are allowed to change location. Rongratana [6, 7] has addressed the problem of identifying *free-regions* of a sensor node so that the connectivity of network is preserved as long as a node remains within its free-region. Since the work presented in Chapter 3 is also dealing with the relocation of nodes we present an overview of the concepts and results reported in [6, 7]; where the relocation is done for nodes of Unit disk graph.

Unit disk graph is a very useful concept for application in the sensor networks. Unit disk graph is defined when all nodes have identical transmission range which is taken without loss of generality as 1. Each sensor node becomes the vertex of the Unit disk graph. Two nodes v_1 and v_2 are connected by an edge if the distance between v_1 and v_2 is less than or equal to 1. Basically, a Unit disk graph (UDG) G(V,E) is obtained by connecting all vertices that are within the transmission range (=1). Figure 2.8 shows an example of UDG with indicated range 1. It is remarked that UDG becomes very dense if the transmission range is large. Let TD(i) denote the transmission disk of node v_i.

The free-region of a node is computed in terms of *in-free region* and *out-free region* of that node. Consider a sensor-node v_1 whose neighbor nodes in the Unit disk graph are v_2, v_3 and v_4 as shown in Figure 2.9. A node nd_j is called an *outbound node* of node nd_i if (i) nd_j lies outside the transmission disk TD(i) of nd_i and (ii) the transmission

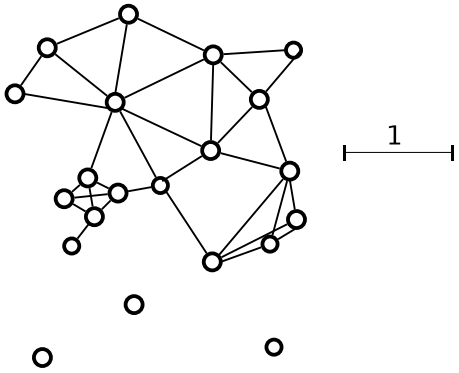
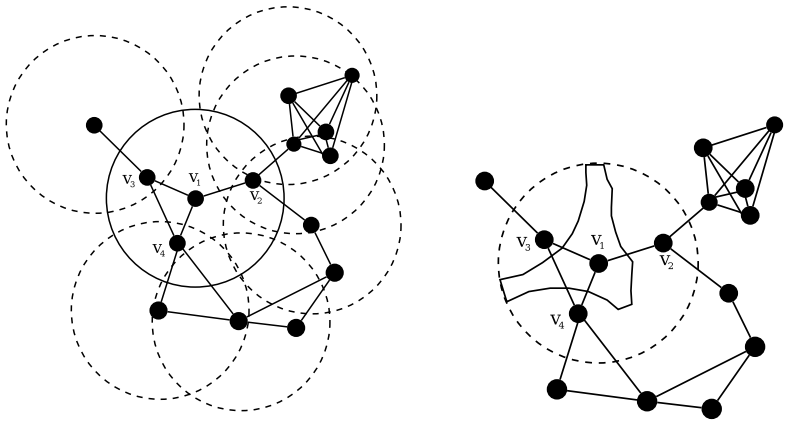


Figure 2.8: Illustrating Unit disk graph

disks TD(i) and TD(j) intersect. The disks of out-bound nodes of v_1 are drawn dashed in the Figure 2.9. The region of TD(i) that is not intersecting with the transmission

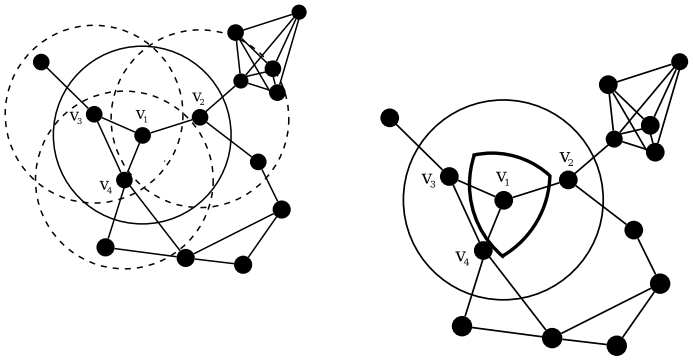


(a) The transmission disk (drawn dashed) of outbound nodes of v_i

(b) Formation of outfree region

Figure 2.9: Illustrating an out-free-region of a node

disks of outbound nodes of v_1 gives the out-free region of v_1 as shown in Figure 2.9 b. Similarly, the notion of *in-free region* is considered. In-bound nodes of v_1 are the nodes that lies within its transmission disk. The intersection of the transmission disk of inbound nodes give the in-free region as shown in Figure 2.10. The details are reported in [6, 7]. The intersection of in-free region and out-free region precisely gives the *free region* of node v_1. As long as node v_1 remains within its free-region the Unit disk graph doesn't change. Free-region of a sensor node can be computed in O(k^2) time where k is the number of out-bound and in-bound nodes of v_1 [6, 7]. It is interesting to note that the problem of computing the free region of a sensor node



(a) The transmission disk (drawn dashed) of inbound nodes of v_i

(b) Formation of in-free-region of v_i

Figure 2.10: Illustrating an in-free-region of a node

has lower bound $\Omega(klogk)$ which is proved in [6, 7] by reducing the sorting problem to the free-region computation problem. Figure 2.11 illustrates a free-region of node v_1.

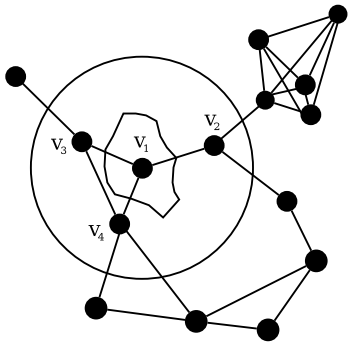


Figure 2.11: Illustrating a free-region of a node

2.5 Constrained Delaunay Triangulation

A well known variation of Delaunay triangulation is the Constrained Delaunay triangulation (CDT). In the standard Delaunay triangulation, the input consists of only point sites. In the Constrained Delaunay triangulation, the input consists of (i) a set of point sites $p_0, p_1, p_2, ..., p_{n-1}$and (ii) a set of line segments constraints $l_0, l_2, ..., l_{m-1}$. The set of line segments could be the edges of a polygon. The constrained Delaunay triangulation is defined again in terms of the empty circle test with a little "twist". The edges of a CDT are the edges of the given constraint line segments and "other

edges". While performing the empty circle test for CDT, the portion of the circle chopped by constraint line segments is ignored. Let us clarify it with an example as shown in Figure 2.12. In the standard DT, v_1 and v_3 cannot be connected to make

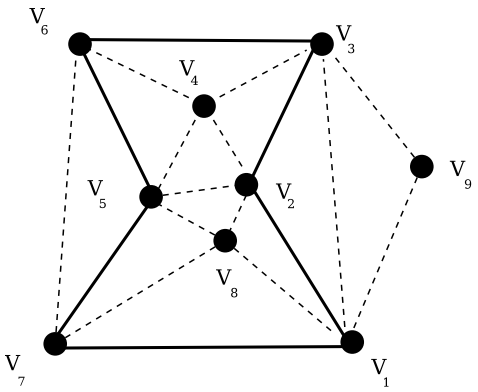


Figure 2.12: Illustrating constrained Delaunay triangulation

an edge of DT because no empty circle can be drawn with v_1 and v_3 on its boundary. In CDT the portion of the circle lying past the constraint lines v_1v_2 and v_2v_3 are ignored. So, v_1 is connected to v_3. When we make other connections we get CDT as shown in Figure 2.12.

Algorithms for constructiong CDT are reported in [2]. The algorithm reported in [2] constructs CDT in O(n logn) time by using divide and conquer algorithm. It has been remarked by several authors that CDTs have application in constructing trees. By using the algorithm given in [2], constrained minimum spanning tree can be computed in O(n logn) time. Another interesting application of CDT is in the motion planning of robots in the presence of obstacles.

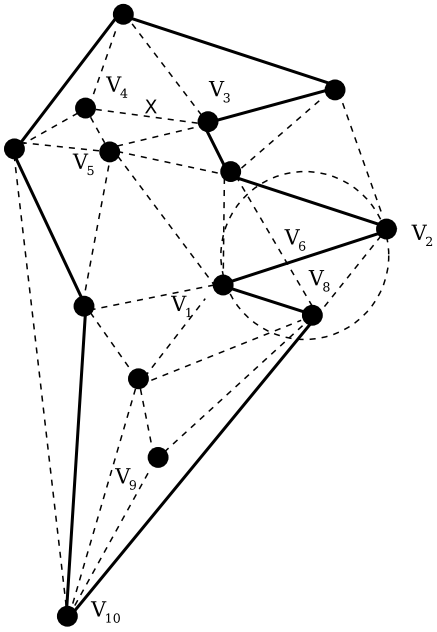
Another version of constrained Delaunay triangulation is reported in [9], where new vertices are introduced to refine the triangulation. The input is a planar straight line graph G' containing a set of free vertices W and set of line segments L. The total vertices are the vertices V' in G' and the vertices in W. Initially a triangulation is done for vertices in V'∪W. If the initial Delaunay triangulation edges intersect the edges of planar graph G' then the intersecting edges of G' are split at the middle to introduce new vertices. After the introduction of such new vertices other new vertices are introduced based on the following two conditions: (i) Encroachment condition (ii)

Skinny condition.

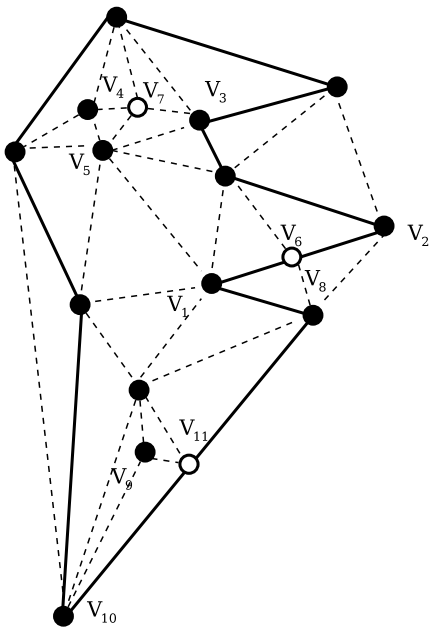
Encroachment condition: A free vertex v encroaches an edge s_1 of PSLG if *diametral circle* of s_1 contain vertex v. In Figure 2.13a, segment v_1 and v_2 is encroached by vertex v_8. Each encroached segments are split at the middle by introducing a new vertex.

Skinny Condition: If a triangle has very small angle say α then such triangles are refined by introducing new vertices. Suppose triangle $t_1 = (v_3v_4v_5)$ is a skinny triangle. Then consider a circumscribing circle c_1 of t_1. The center of t_1 is inserted as new vertex and re-triangulation is performed for the new set of vertices. An example of such refinement is shown in Figure 2.13.

This process of triangulation refinement is continued until certain fraction of new vertices are introduced or some other criteria is reached. Details are found in [9].



(a) The edge v_1v_2 intersecting with a delaunay edge and triangles $v_3v_4v_5$ and $v_8v_9v_{10}$ with one of the internal angles less than specified minimum angle



(b) Adding new vertices v_6, v_7 and v_{11} to given set of points and re-triangulating them to give a refined delaunay triangulation

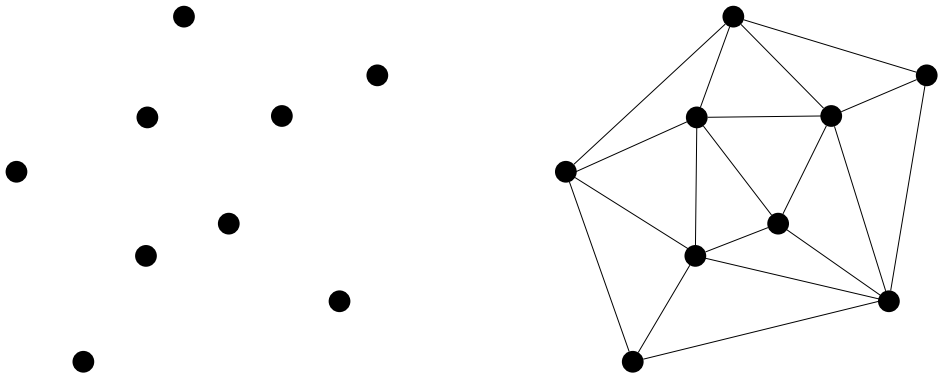
Figure 2.13: Illustrating Delaunay refinement algorithm

CHAPTER 3

ROAMING REGION FOR DELAUNAY NODES

3.1 Problem Formulation

Consider a set of nine nodes "S" as shown in Figure 3.1a. The Delaunay triangulation of these nodes is shown in Figure 3.1b. If we move a node slightly then it is



(a) a set of nine nodes (b) Delaunay triangulation

Figure 3.1: Illustrating Delaunay triangulation

very likely that the Delaunay triangulation of "S" will not change. On the other hand if we continue to move a point significantly further from its initial position then the resulting Delaunay triangulation changes. This change of Delaunay triangulation is shown in Figure 3.2. In Figure 3.2, initially node v_0 is connected directly to v_1, v_2, v_3 and v_4. When v_0 is moved to new position as shown in the figure, it will be connected to one more node which is v_5. This observation shows that it would be interesting

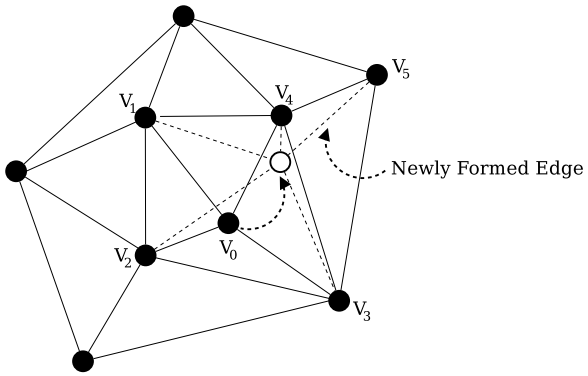


Figure 3.2: Illustrating the change of triangulation by node movement

to deteremine the connected region for a node such that the Delaunay triangulation

remains the same no matter where the node is placed in the region. To formulate this problem formally we extend the free-region concept introduced in [6] for unit disk graph.

Definition 3.1 *Consider the Delaunay triangulation of a set "S" of points in the plane. The roaming-region of a node v_i is the maximal region $R_m(i)$ in the proximity of v_i such that the Delaunay triangulation does not change when v_i is moved to any point within $R_m(i)$. An example of a roaming region is shown in Figure 3.3.*

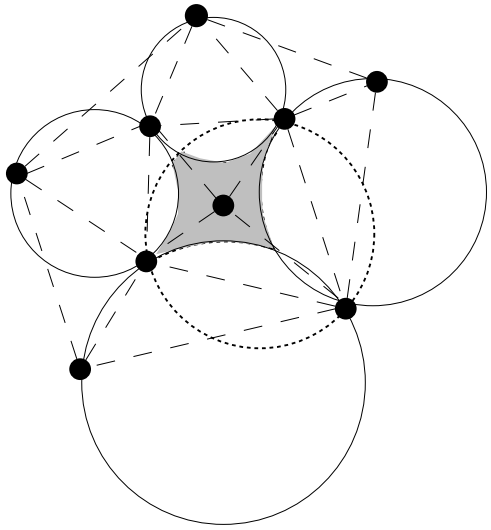


Figure 3.3: Illustrating roaming-region

3.2 Node Classification

To develop an algortihm for constructing roaming region for nodes it is necessary to distinguish them into several classes. Nodes that are on the boundary of the convex-hull of nodes are called *external nodes* and those that are inside the convex-hull are the *internal nodes*. The internal nodes can be further distinguished as *shallow-internal* and *deep-internal*.

Definition 3.2 *Consider the Delaunay Triangulation DT of given nodes. An internal node v_i is called deep-internal if all neighbors of v_i are internal nodes. In Figure 3.4, node v_0 is a deep-internal. In fact node v_0 is the only deep-internal node in the triangulation of Figure 3.4.*

Definition 3.3 *An internal node v_i is called shallow-internal if some neighbor of v_i is an external node. In Figure 3.4, four nodes v_1, v_2, v_3, and v_4 are shallow-internal nodes.*

Remark 3.1 *It is remarked that the set of nodes in a Delaunay triangulation T can be viewed as the union of three disjoint sets: (i) external nodes, (ii) deep-internal nodes and (iii) shallow-internal nodes.*

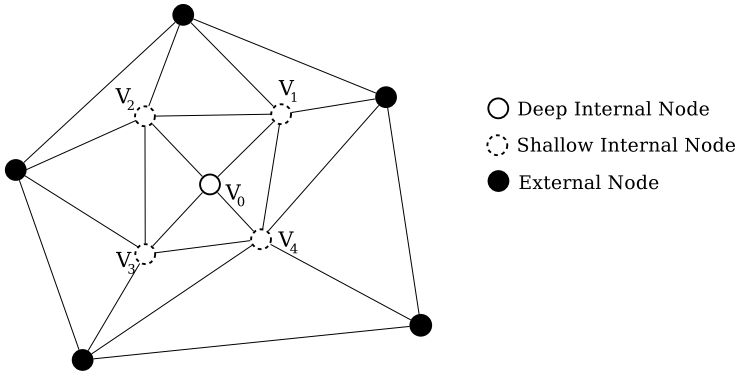


Figure 3.4: Distinguishing node types

3.3 Roaming Region for Deep-Internal Nodes

Before developing methods for computing roaming regions we recall one of the key properties of Delaunay triangulation which states that the circum-circle of any triangle of a Delaunay Triangulation is empty, i.e. the circum-circle does not contain any other node of the triangulation. This property is called "empty-circle property". Our method of computing roaming region of a node is based on the use of set intersection and set differences of circum-circles of carefully selected triangles (both Delaunay and non-Delaunay) in the proximity of the candidate node. For this purpose we start with the characterization of *radial* and *lateral* triangles for a given node v_i as follows.

Definition 3.4 *Consider a candidate Delaunay node v_i. Let $t_1, t_2, t_3, ...t_k$ be the triangles incident on v_i. If the degree of v_i is k then there are k incident triangles. The triangles sharing the sides of incident triangles opposite to v_i are the radial triangles of v_i. In Figure 3.5, radial triangles for node v_i are shown shaded grey.*

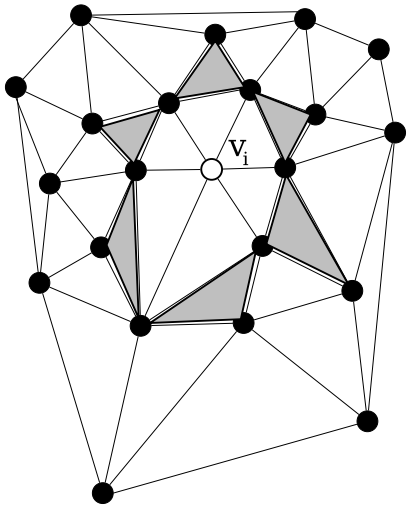


Figure 3.5: Illustrating radial triangles for deep internal node v_i

An inspection of the radial triangles of boundary and internal nodes of Delaunay triangulations shows that some nodes have radial triangles for all incident triangles while other nodes do not have radial triangles for all of them. This leads to the following remark.

Remark 3.2 *For a deep-internal node v_i there will be radial triangles for each incident triangle of v_i.*

The notion of lateral triangles is captured by considering consecutive incident triangles $t_i t_{i+1}$ for a candidate node v_i. Lateral triangles as conceptualized below are not Delaunay triangles. Figure 3.6 shows two of the lateral triangles for a Delaunay node v_i.

Definition 3.5 *Let $t_1, t_2, ...t_k$ be the triangles incident on node v_i such that they are ordered angularly around v_i. A pair of consecutive incident triangles $t_i t_{i+1}$ form a quadrilateral $v_i v_p v_q v_r$. The non-Delaunay triangle $v_p v_q v_r$ is a lateral triangle for vertex v_i. In Figure 3.6, only two out of six possible lateral triangles are shown.*

Radial Roaming Region: Let $R_1, R_2, R_3, ...R_n$ be radial disks of v_0. Let $v_1, v_2, v_3, ...v_m$ be the neighboring nodes of v_0. Let D_{max} be the convex-hull area of the point sites

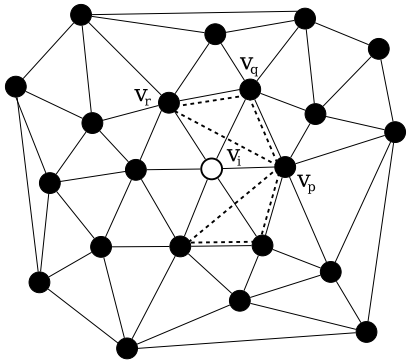


Figure 3.6: Illustrating lateral triangles for deep internal node v_i

including v_0 and neighbor nodes $v_1, v_2, v_3, ...v_m$. Then, the radial roaming region of v_i is:

$$\text{Radial Roaming Region} = \text{RR(i)} = D_{max} \text{ - } R_1 - R_2 ..... - R_n \text{ (i)}$$

In Figure 3.7, radial circum-circles R_1, R_2, R_3 and R_4 are drawn with respect to delaunay triangles D_1, D_2, D_3 and D_4 respectively. In this scenario, Delaunay triangles D_1, D_2, D_3 and D_4 are radial triangles of v_0.

To compute a free roaming region for a node v_i, considering the radial circum-circles is not enough. So, we also need to consider another concept of "lateral circum-circles". Radial circum-circles and lateral circum-circles are shown in Figure 3.7 and Figure 3.8 (dotted circles) respectively.

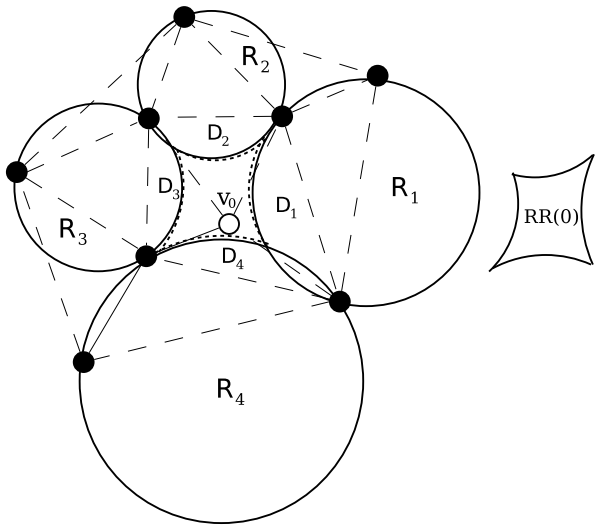


Figure 3.7: Illustrating radial roaming region for v_0

Lateral Roaming Region: Let $L_1, L_2, L_3, ...L_k$ be the lateral disks of v_i. Then lateral roaming region LR(i) for node v_i is given by

$$\text{Lateral Roaming Region} = \text{LR(i)} = \cap(L_1, L_2, L_3, ...L_k) \text{ (ii)}$$

The intersection of radial roaming region RR(i) and lateral roaming region LR(i) precisely gives the roaming region R(i) for node v_i as follows.

$$\text{R(i)} = \cap(\text{LR(i)},\text{RR(i)}) \text{ (iii)}$$

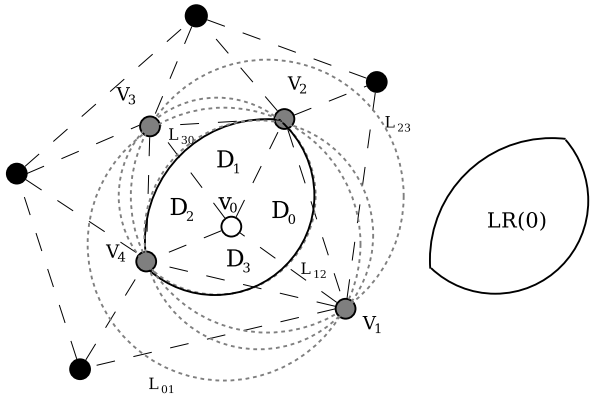


Figure 3.8: Illustrating radial roaming region for v_0

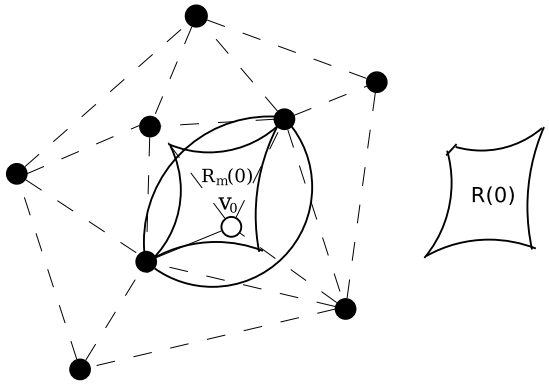


Figure 3.9: Illustrating formation of roaming region R(0)

The formation of radial roaming region RR(0) for node v_0 is elaborated in Figure 3.7. The radial disks are drawn with circles and the convex-hull region chopped by radial disks are shown on the right side of the figure. Similarly, the formation of lateral roaming region LR(0) for node v_0 is elaborated in Figure 3.8. The lateral disks are drawn with dotted circles and their intersection LR(0) is shown on the right

side of the figure. The overlay of lateral roaming region LR(0) and radial roaming region RR(0) is shown in Figure 3.9. The intersection of LR(0) and RR(0) gives the roaming region R(0) as shown on the right side of the figure.

The algorithm for computing the roaming regions for internal nodes is sketched below:

Roaming Region Algorithm for Internal Nodes

Input: (i) Delaunay Triangulation DT

(ii) Interior node v_i

Output: Roaming region R(i) for v_i

Algorithm 1 Roaming Region for Internal Node

1: a. Determine all the neighboring nodes $v_1, v_2, ..., v_m$ of v_i in DT
b. Let D_{max} be the convex-hull area of the point sites including v_i and neighboring nodes $v_1, v_2, ..., v_m$
2: (i) Determine the radial triangles $t_1, t_2, ..., t_k$ for v_i
(ii) Compute radial disks $R_1, R_2, ..., R_k$ corresponding to $t_1, t_2, ..., t_k$
3: Determine RR(i) = D_{max} - R_1 - R_2 ... -R_k
4: Identify lateral disks $L_1, L_2, ..., L_p$ for vertex v_i
5: Compute LR(i) = $\cap(L_1, L_2, ..., L_p)$
6: Determine and output R(i) = $\cap$(LR(i), RR(i))

Theorem 3.1 *The roaming region for deep-internal node v_i in a Delaunay triangulation DT can be computed in $O(n^2)$ time, where n is the number of nodes in DT.*

Proof. We assume that the given Delaunay triangulation is available in a doubly-connected edge list data structure. Step 1 can be done in O(d(v_i)) time by following the edge list incident on v_i, where d(v_i) is the degree of node v_i. Since the number of edges in a DT are linearly related to the number of vertices n, d(v_i) = n. Hence, Step 1 and Step 2 take O(n). Once radial triangles around v_i are available, radial disks can be determined in O(n) time. By navigating the doubly connected edge list data structure, lateral triangles of v_i can be determined in O(n) time. Finally, the intersection of lateral disks and the intersection of LR(i) and RR(i) can be done in O(n^2) time in a straightforward manner. □

3.4 Roaming Regions for External Nodes

While roaming regions for internal nodes are always finite and bounded, the corresponding roaming regions for external nodes could be both bounded or unbounded.

It is interesting to identify the cases where the roaming region for an external node is bounded. Consider the case in which the number of convex-hull edges is at least five as shown in Figure 3.10. Let v_i be the candidate external node for which we want to identify the roaming region. We relabel the nodes as the convex-hull boundary, in counterclockwise traversal as $v_{i-2}, v_{i-1}, v_i, v_{i+1}, v_{i+2}$.... We further denote the convex-hull edges as $e_{i-2} = (v_{i-1}, v_{i-2})$, $e_{i-1} = (v_i, v_{i-1})$, $e_i = (v_i, v_{i+1})$, $e_{i+1} = (v_{i+1}, v_{i+2})$ ... If the extension of edges e_{i+1} and e_{i-2} meet at point q_i then the roaming region will be bounded and inside the union of triangle $T_i = (v_{i-1},q_i,v_{i+1})$ and the convex-hull region of the whole point sites.

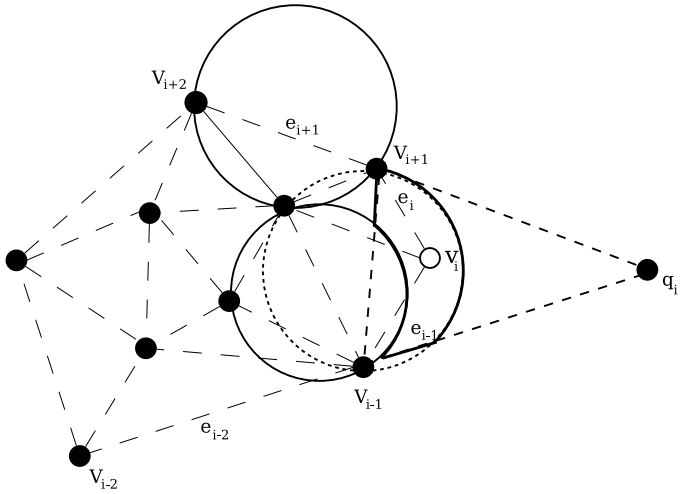


Figure 3.10: Illustrating bounded roaming region for an external node v_i

On the other hand, if the extension of edges e_{i-2} and e_{i+1} diverge as shown in Figure 3.11, then the roaming region could be unbounded for certain distribution of nodes. In order to come up with the algorithm for capturing roaming regions for external nodes we need to identify all the cases for which the roaming regions are finite.

Once such cases are properly identified, it would be straightforward to extend the algorithm for internal nodes to the nodes on the convex-hull-boundary.

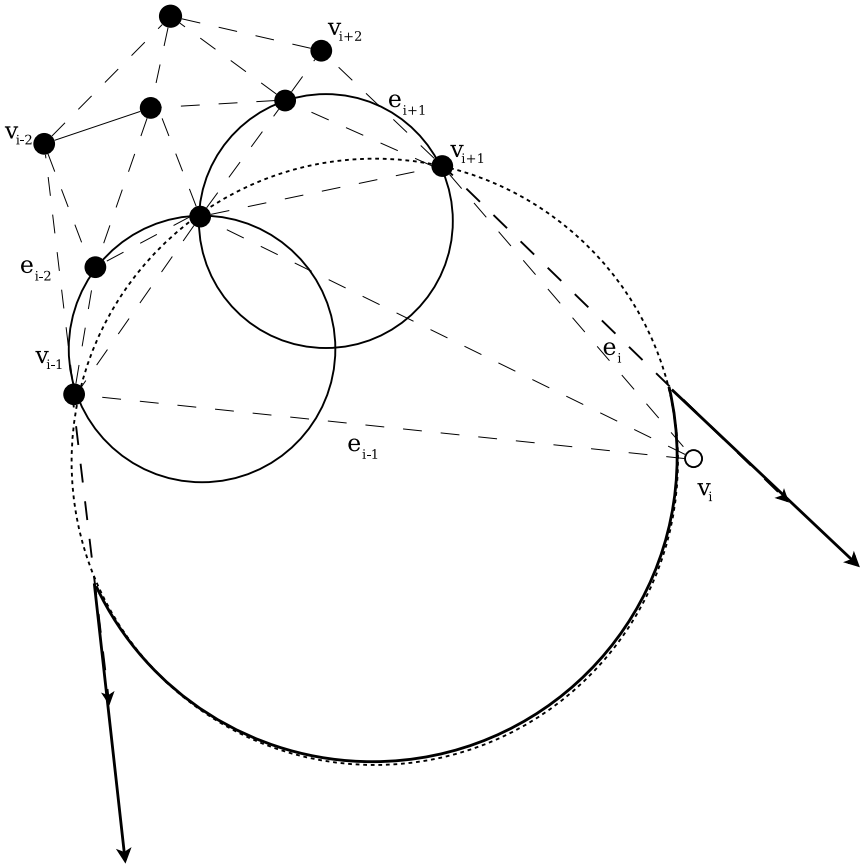


Figure 3.11: Illustrating unbounded roaming region for an external node v_i

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

This chapter describes an implementation and study of the reliability of Delaunay triangulation nodes. The programs were implemented in Java, Version 1.6.

The implementation of determining a roaming regions can be categorized into three types, i) roaming region for shallow internal nodes ii) roaming region for deep internal nodes, and iii) roaming region for external nodes. The roaming region area for a node v_i is defined by an intersection of *radial* and *lateral* roaming region associated with given node v_i.

There are three stages for determining a roaming region. In the first stage, we determine the radial region for given node v_i, while the second stage deals with the determination of the lateral roaming region; and the final stage deals with the determining an intersection of radial and lateral roaming regions resulting in a roaming region for node v_i.

4.1 Interface Description

The main Graphical User Interface(GUI) window is implemented by extending the JFrame class component in javax.swing which consists of two panels, as shown in Figure 4.1 The menu bar panel is added to the JFrame on the top, which contains the File and Tools Menu. All other panels contained within the JFrame object are constructed by using JPanel class. The whole panel is classified as left and right, where left panel contains the main display area and the coordinates of mouse cursor are displayed at the leftmost corner of the panel. Finally, the right panel contains the buttons which are used to i) display radial, lateral and roaming region, ii) select and manipulate the vertices and edges of the Delaunay graph, and iii) remove the vertices, edges and circum-circles from the GUI main panel. The right panel also consists of text areas for displaying coordinates of the current vertices of the main panel.

A user can initiate a graph by adding vertices and can grow a bigger planar graph

by adding a sequence of vertices and edges and by splitting edges and by splitting faces. Edge addition, edge split and face split can be done one at a time.

As the implementation is based on Delaunay triangulation, each time a user adds, edits or deletes vertices of a graph the Delaunay triangulation is computed and the resulting Delaunay graph is displayed in the main panel. There are several checkboxes and buttons to manipulate and generate the graph. The functionality of each menu item, checkbox and button are breifly described in Table 4.1, Table 4.2 and Table 4.3 respectively.

Table 4.1: Buttons description.

	Button Label	**Functionality**
1	Random Sites	Used to generate a Delaunay graph at the main panel with random numbers of nodes
2	Roaming Region	Used to display a roaming region for a selected node in the Delaunay graph
3	Radial Region	Used to display a radial region for a selected node in the Delaunay graph
4	Lateral Region	Used to display a lateral region for a selected node in the Delaunay graph
5	Radial & Lateral	Used to display both radial and lateral region for a selected node in the Delaunay graph
6	Quandrangulation	Used to quandrangulate a Delaunay graph displayed at main panel
7	Refresh TxtBox	Used to refresh text box at right panel displaying current co-ordinates of the vertices of the graph
8	Clear Circles	Used to remove the radial or lateral or both circumcircles for a selected node in the Delaunay graph
9	Clear All	Used to clear everything at the center panel

Table 4.2: FileMenu description.

	Menu item	**Functionality**
1	Read DCEL File	Reads a stored file containing a graph in a form of Doubly connected edge list.
2	Save DCEL File	Saves a graph in main panel of GUI to a file in a form of Doubly connected edge list.
3	ExportToXfig	Exports and saves a graph in main panel of GUI to **.eps* format.

Table 4.3: ToolsMenu description.

	Menu item	**Functionality**
1	Draw Vertex	Draws a vertex of the graph.
2	Edit Vertex	Moves the position of selected vertex by dragging mouse.
3	Delete Node	Deletes the selected vertex of a graph by updating the values to the connecting vertices.
4	Add Edge	Adds an edge to any vertex v_i of a selected face of the graph.
5	Delete Edge	Removes selected edge from the graph.
6	Split Edge	Splits the selected edge into two parts by generating a new vertex to the selected edge.
7	Split Face	Splits the face by joining two vertices with an edge.
8	Triangulation	Uses Delaunay triangulation for triangulating the graph.

4.2 Program Menu Items

A File tab and a Tools Tab are represented as menu items in the program. The File menu items enable the user to (i) read and open previously saved graph files, (ii) save a generated graph to a file and (iii) export the generated graph to the file in *.eps* format.

The Tools menu items enable users to (i) create or remove vertices of a graph, ii) move position of vertices, iii) delete vertices, iv) add or delete edges of a graph, v) perform triangulation and quandrangulation of a set of vertices, vi) split an edge of a graph and vii) split a face of a graph. A brief description of the File and Tools items are provided in Table 4.2 and Table 4.3.

Figure 4.2 and Figure 4.3 illustrate the GUI representation of the File menu and Tools menu item and selection panel to choose or save the graph G(V,E) respectively.

4.3 Illustrating Roaming Region

To determine a roaming region for a node v_i of a Delaunay graph, the *radial* and *lateral* flags should be enabled to instruct the program to perform computations of both radial as well as lateral regions with respect to a specific node v_i. To perform this action, the checkbox "Roaming Region" in "Tools menu" should be enabled. The algorithm as described in Chapter 3 is implemented in this program for determining

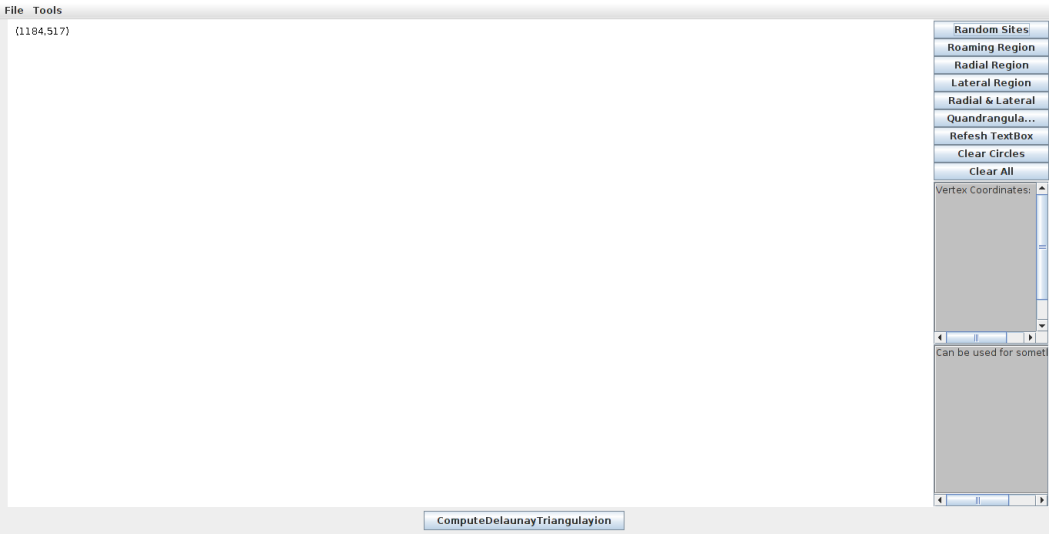


Figure 4.1: The initial display of GUI for graph construction

radial as well as *lateral* roaming regions resulting in a roaming region which is the intersection of radial and lateral roaming regions. After the user clicks "Roaming Region" button, the program will display the roaming region for a node in a separate JFrame as soon as the user clicks near by a node in the main panel of the GUI for which the roaming region is to be computed and displayed. Figure 4.4 and Figure 4.5 illustrates radial regions for deep internal node and shallow internal node respectively. Similarly Figure 4.6 and Figure 4.7 illustrates lateral regions for deep and shallow internal nodes respectively.

The program also provides an option to compute and display a roaming region for a node in three differnt steps and the steps are i) determining radial roaming region ii) determining lateral roaming region for a given node and iii) finally, determining a resulting roaming region for a given node as an area common to both radial as well as lateral roaming regions.

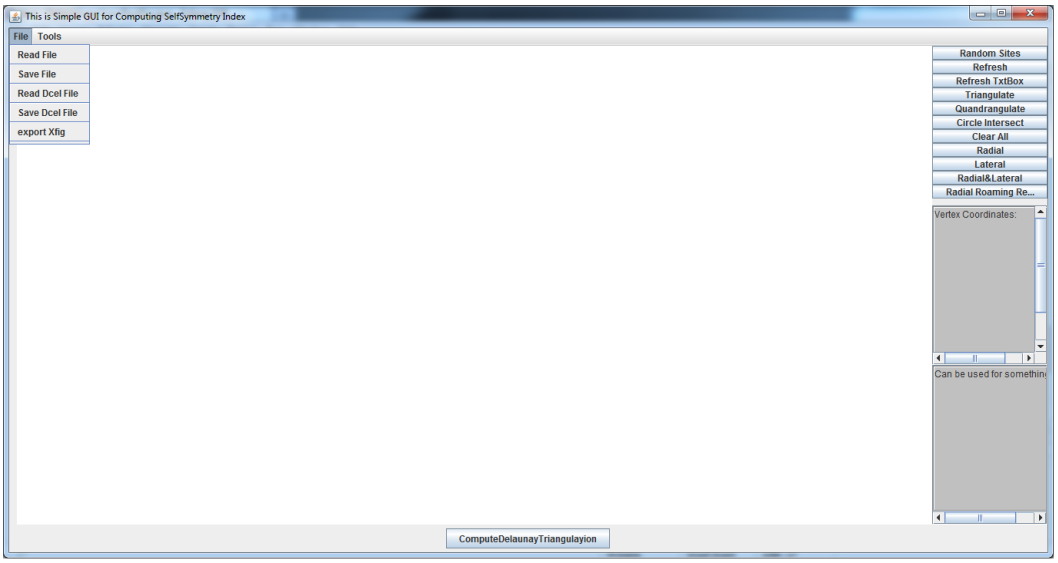


Figure 4.2: Display of file menu

4.4 Radial, Lateral and Roaming Region Computation

For an efficient computation during the implementation, all the radial and lateral circum-circles are converted to their respective polygons.

The program computes the *radial region* in the form of a polygon which is formed by a convex-hull region of neighborhood nodes of a target node v_i chopped by the radial circum-circles of the node v_i. Similarly, the lateral region is also computed in the form of a polygon as a result of intersection of two or more lateral circum-circles which encloses the target node v_i.

The roaming region for a target node v_i is computed after the computation of the radial and lateral regions in the form of polygons. Finally, the intersection of radial and lateral regions results in the roaming region for node v_i. Figure 4.8 and Figure 4.9 illustrates roaming regions for deep and shallow internal nodes respectively.

Similarly for an external node, the program identifies one of the two cases as described in Chapter 3. For a convergence case, the program computes a triangle T_i as described in Chapter 3 which bounds the roaming region of node v_i. The program

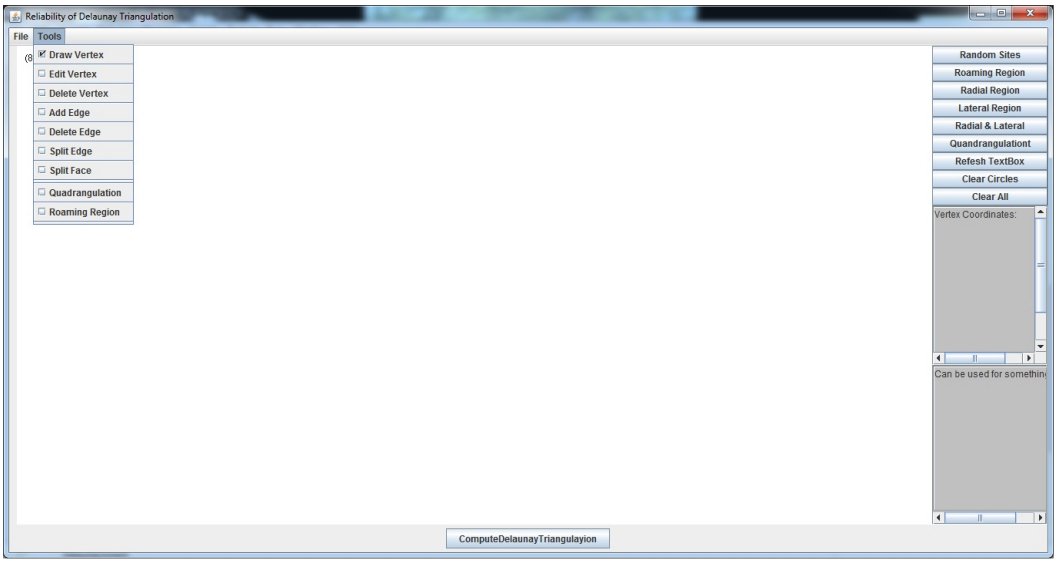


Figure 4.3: Display of tools menu

also checks for possible effects of radial and lateral circum-circles in the roaming region of v_i. Figure 4.10 illustrates an example of bounded roaming region for an external node.

For any divergence case, except for certain distribution of nodes, the program displays an unbounded roaming region for node v_i. Figure 4.11 illustrates an example of unbounded roaming region for an external node.

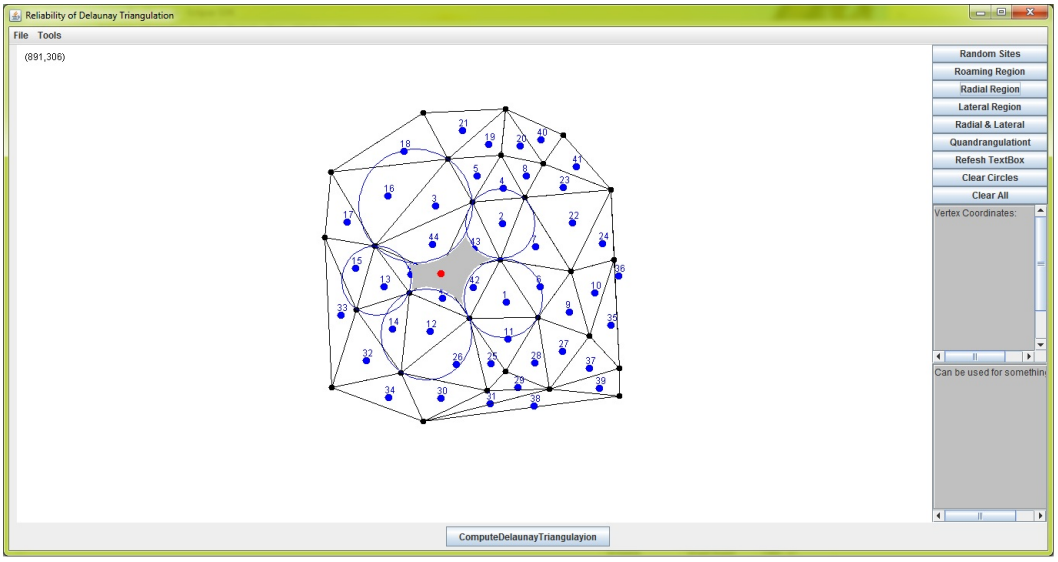


Figure 4.4: Illustrating a radial region for a deep internal node

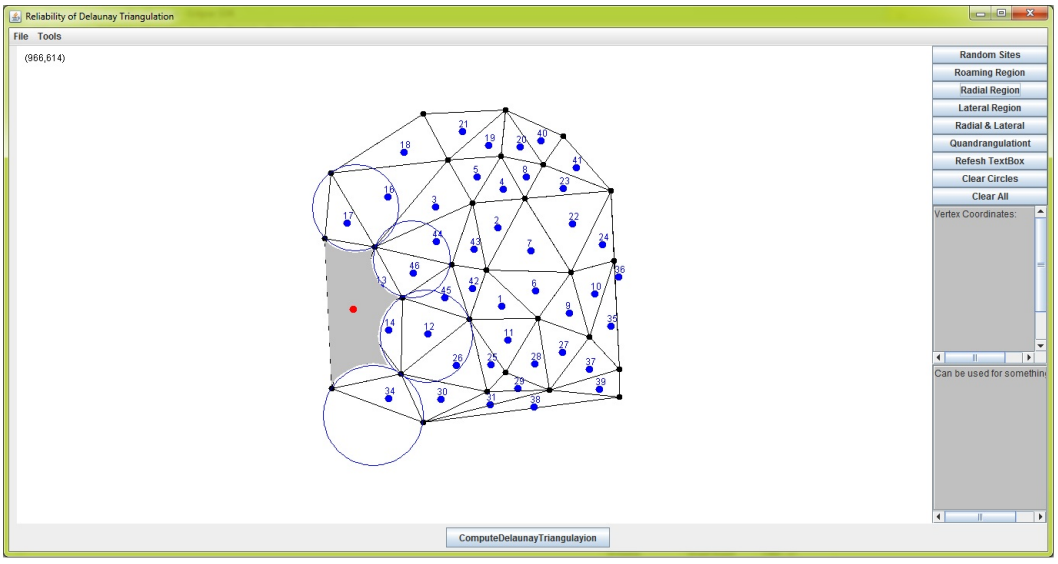


Figure 4.5: Illustrating a radial region for a shallow internal node

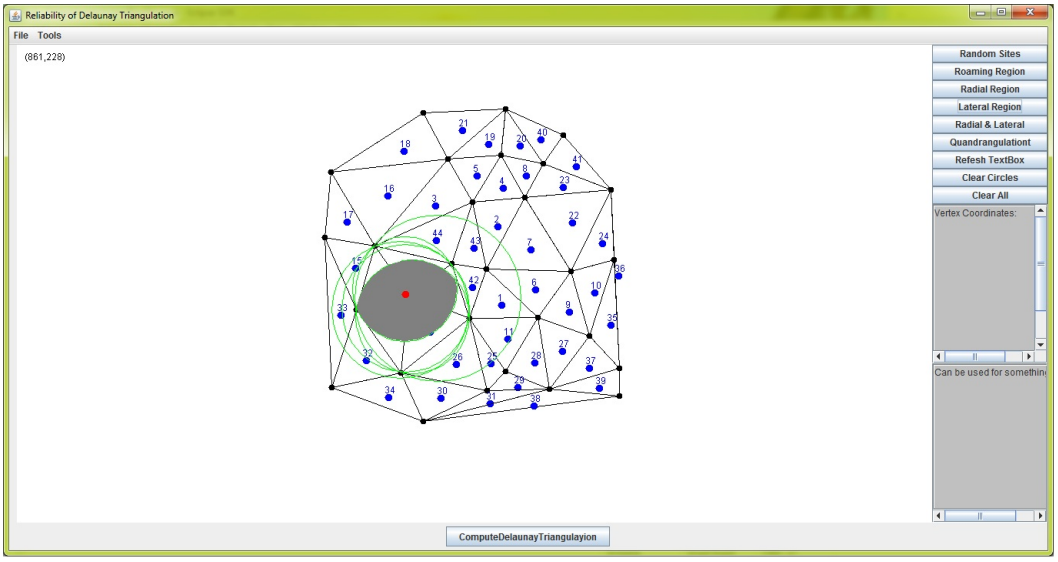


Figure 4.6: Illustrating a lateral region for a deep internal node

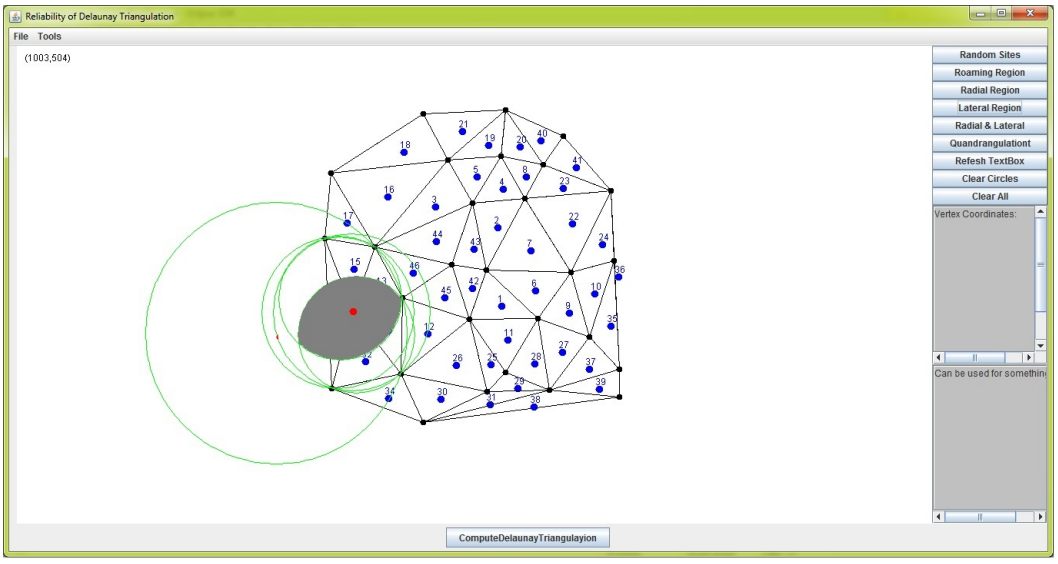


Figure 4.7: Illustrating a lateral region for a shallow internal node

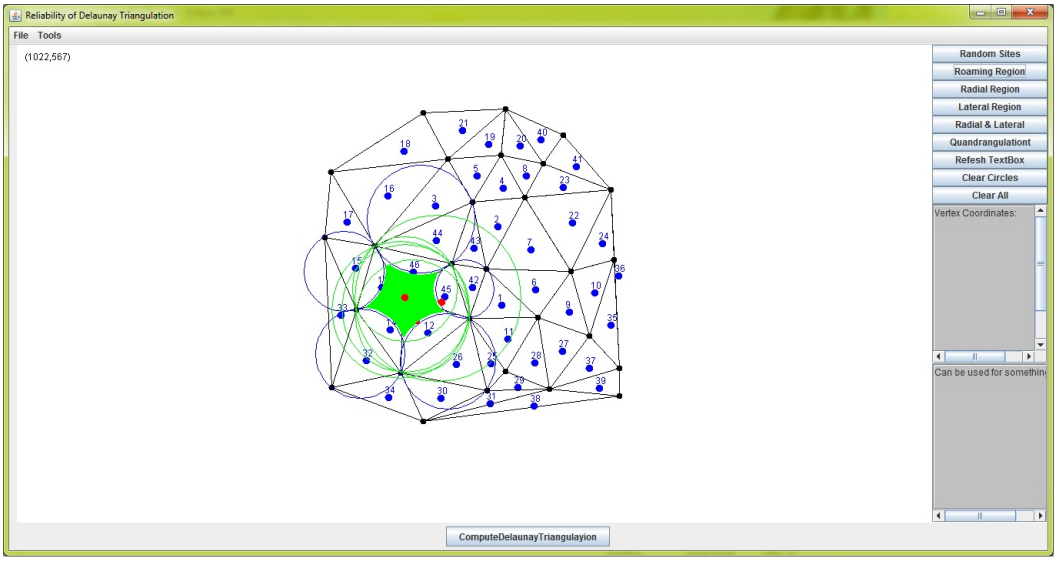


Figure 4.8: Illustrating a roaming region for a deep internal node

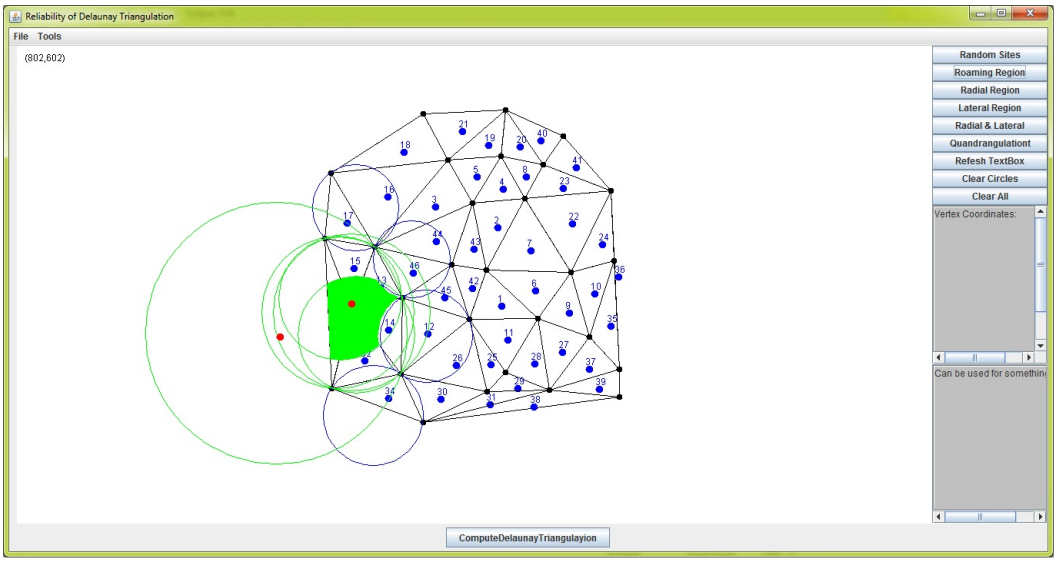


Figure 4.9: Illustrating a roaming region for a shallow internal node

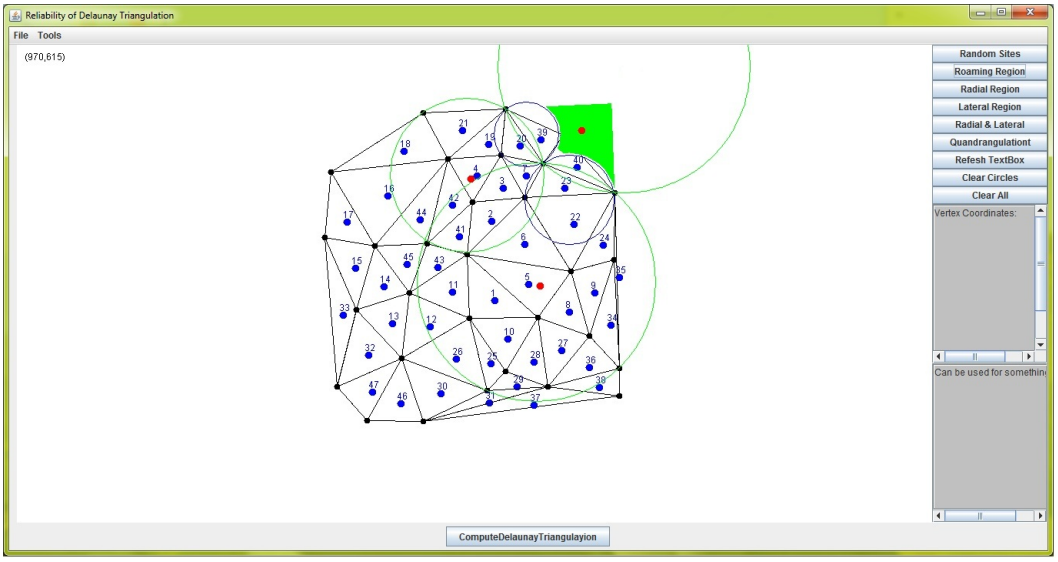


Figure 4.10: Illustrating a bounded roaming region for an external node

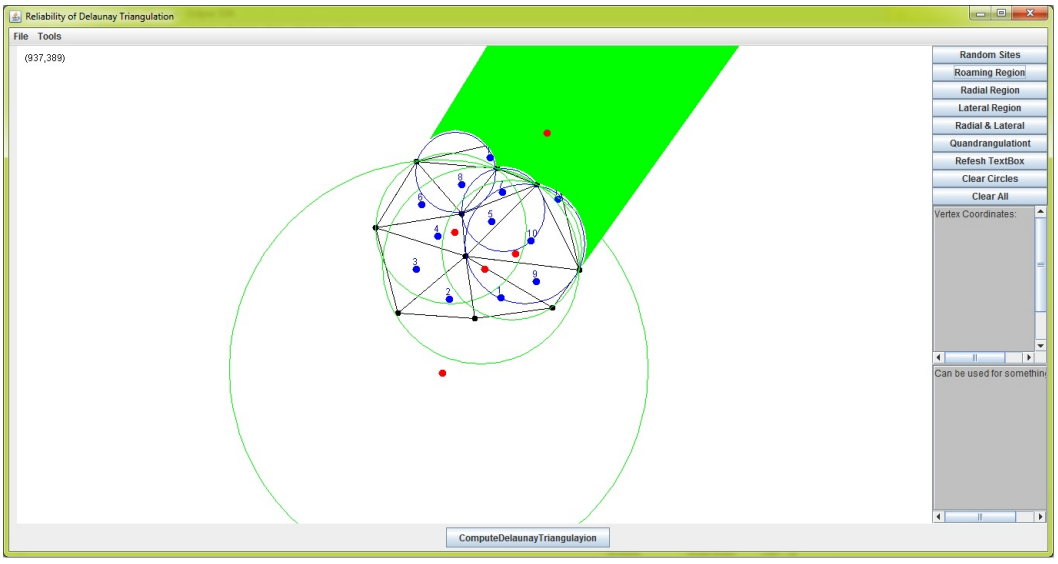


Figure 4.11: Illustrating an unbounded roaming region for an external node

CHAPTER 5

CONCLUSION

We reviewed well known algorithms for computing Delaunay triangulation of point sites in two dimensions. The properties of Delaunay triangulation were examined for capturing the proximity properties of point sites. We introduced the problem of computing roaming regions for nodes of Delaunay triangulation. To capture the roaming region of Delaunay nodes we formulated the notions of lateral roaming region and radial roaming region. We showed that the roaming region is precisely given by the intersection of radial roaming region and lateral roaming region. This characterization led us to the development of an efficient algorithm for computing the roaming region of a Delaunay node.

We considered the implementation issues of the proposed algorithm and presented an implementation of the proposed algorithm in Java programming language. The implementation can display the roaming region for internal nodes.

Several further investigations can be performed by extending the concepts introduced in this thesis. An interesting problem would be to develop an algorithm for identifying nodes with maximum reliability. Due to time constraint we only implemented the computation of roaming region for internal nodes. It would be interesting to complete the implementation of roaming region computation for all types of nodes.

REFERENCES

[1] K. Q. Brown. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5):223–228, 1979.

[2] L. Paul Chew. Constrained delaunay triangulation. *Algorithmica*, pages 97–108, 1989.

[3] Vedat Coskun. Relocating sensor nodes to maximize cumulative connected coverage in wireless sensor networks. *Sensors*, 8, 2008.

[4] B. N. Delaunay. Sur la sphre vide: Izv. akad. nauk sssr. *RNauk SSSR, Otdel. Mat. Est. Nauk*, 7:793–800, 1934.

[5] Marc van Krevel. Mark Overmars Mark de Berg, Otfried Cheong. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.

[6] Jan B. Pedersen N. Rongratana, Laxmi Gewali. Estimating the free region of a sensor node. *JCMCC*, 74:53–64, 2010.

[7] Henry Selvaraj Navin Rongratana, Laxmi Gewali. Characterizing free-regions of sensor nodes. *International Conference on Systems Engineering*, pages 375–379, 2008.

[8] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.

[9] Jim Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *NASA Ames Research Center*, 1993.

VITA

Graduate College
University of Nevada, Las Vegas

Romas James Hada

Home Address:
4441 ESCONDIDO STREET APT 1205
Las Vegas, NV 89119

Degree:
Bachelor of Computer Engineering
Institute of Engineering, Pulchowk Campus, Tribhuvan University, Nepal

Thesis Title: Roaming Region for Delaunay Triangulation

Thesis Examination Committee:
Chairperson, Dr. Laxmi Gewali, Ph.D.
Committee Member, Dr. John T. Minor, Ph.D.
Committee Member, Dr. Ajoy K. Datta, Ph.D.
Graduate Faculty Representative, Dr. Rama Venkat, Ph.D.