UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

ANALYSIS OF LONGEST-EDGE ALGORITHMS FOR 2-DIMENSIONAL MESH REFINEMENT

TESIS PARA OPTAR AL GRADO DE DOCTOR EN CIENCIAS, MENCIÓN COMPUTACIÓN

CARLOS EDUARDO BEDREGAL LIZÁRRAGA

PROFESOR GUÍA:
MARÍA CECILIA RIVARA ZÚÑIGA

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CÁRDENAS
NANCY HITSCHFELD KAHLER
CHEE YAP

Este trabajo ha sido parcialmente financiado por Beca CONICYT - Chile y por Beca del Departamento de Ciencias de la Computación de la Universidad de Chile

SANTIAGO DE CHILE
2015

Resumen

Las técnicas de generación y refinamiento de mallas no estructuradas son usadas para la descomposición de objetos geométricos. Estas técnicas son muy utilizadas en áreas como modelamiento geométrico, computación gráfica, computación científica y aplicaciones de ingeniería, entre otras, lo que les da un interés interdisciplinario.

Trabajando con triangulaciones (mallas compuestas por triángulos), el reto es generar una descomposición precisa del objeto geométrico o dominio, y al mismo tiempo satisfacer las restricciones adicionales impuestas por la aplicación, como restricciones en la forma de los elementos, el número de elementos, o la transición entre elementos de diferentes tamaños. Los algoritmos que ofrecen garantías teóricas sobre estos temas son preferidos.

Los algoritmos de arista más larga fueron diseñados para el refinamiento iterativo de triangulaciones en aplicaciones de método de elementos finitos adaptativo. Estos algoritmos está basados en la estrategia de propagación por la arista más larga. Comparados a otros algoritmos de refinamiento, los algoritmos de arista más larga rápidamente producen una descomposición del dominio (o de regiones de interés) a través de operaciones locales simples. Las triangulaciones obtenidas presentan buena densidad y la calidad de los triángulos refinados est acotada. El propósito de esta tesis es proporcionar nuevas garantías teóricas para los algoritmos de arista más larga basados en bisección y los algoritmos de arista más larga basados en refinamiento Delaunay, para la generación y el refinamiento de mallas de buena calidad en 2 dimensiones.

Nuestro estudio del algoritmo basado en bisección muestra que el algoritmo inserta un número constante de puntos por triángulo refinado, con costo asintóticamente óptimo. También mostramos que durante el proceso de refinamiento el algoritmo mejora la calidad promedio de los triángulos.Obtenemos nuevas cotas para el tamaño de la triangulación refinada y probamos que éste es a lo sumo un factor constante mayor que el tamaño de la triangulación inicial. Esta es la primera prueba completa sobre la complejidad del algoritmo.

Seguidamente estudiamos el algoritmo basado en refinamiento Delaunay y su estrategia de inserción de puntos. Demostramos que los puntos insertados por el algoritmo no pueden estar arbitrariamente cerca de puntos existentes, lo que nos permite acotar la longitud de nuevas aristas. Analizamos el mejoramiento de la calidad de triángulos para diversas cotas en el ángulo mínimo, y definimos las propiedades geométricas de los triángulos obtenidos después del refinamiento. Utilizamos las técnicas existentes para el análisis de algoritmos de refinamiento Delaunay para demostrar que el algoritmo produce triangulaciones de tamaño óptimo, con buena densidad de puntos, y con ángulos internos entre 25.66° y 128.68°.

También estudiamos las propiedades de la propagación en estos algoritmos de arista más larga. Mostramos que el número de tringulos afectados por refinamiento propagado converge rápidamente a aproximadamente dos. Esto demuestra que el refinamiento propagado representa un factor constante en el costo de refinamiento.

Abstract

Unstructured mesh generation and mesh refinement are used for the decomposition of geometric objects, widely applied in areas such as geometric modeling, computer graphics, scientific computing, and engineering among others, making this an interdisciplinary topic.

Working with triangulations (meshes composed of triangles), the challenge is to generate a triangulation that accurately decomposes the object or domain, while satisfying additional constraints imposed by the application such as element shape, number of elements, or the transition between elements of different sizes. Algorithms offering theoretical guarantees on these issues are preferable.

Longest-edge refinement algorithms were designed for the iterative refinement of triangulations in adaptive finite element applications. Based on a longest-edge propagation strategy, these algorithms quickly produce a decomposition of the domain (or regions of interest) through local operations. The refined triangulations are nicely graded, with bounds on the element quality. The purpose of this thesis is to provide new theoretical guarantees on the longest-edge bisection and longest-edge Delaunay algorithms for 2-dimensional mesh refinement.

Our study of the longest-edge bisection algorithm shows that the algorithm inserts a constant number of new points per triangle refined, making the cost of the algorithm asymptotically optimal. We show that the algorithm improves the global quality and grading of the triangulation as it is processed. We obtain new bounds on the size of the output triangulation and prove that it is at most a constant factor larger than the size of the input triangulations, providing the first complete proof of the algorithm's complexity.

We then study the longest-edge Delaunay algorithm's point insertion strategy and show that new points cannot be inserted arbitrarily close to the existing points of the triangulation, bounding the length of the new edges. We analyze the improvement of triangles for various bounds on the minimal angle, and define the geometrical properties of the triangles obtained after refinement. Then, using existing analysis techniques for Delaunay refinement algorithms, we prove that they produce size-optimal triangulations that are nicely graded and have internal angles between 25.66° and 128.68°.

We also study the properties of propagating paths in longest-edge refinement algorithms. We show that the number of triangles affected by propagated refinement quickly converges to approximately two. This proves that propagated refinement represents a constant factor in the cost of the algorithms. An important byproduct of this thesis is a methodology used for the characterization of the triangles' geometry after refinement, which helps to show how longest-edge refinement algorithms naturally produce good quality sub-triangles.

Agradecimientos

Quiero agradecer a mi familia por su amor y apoyo incondicional. En especial a mi esposa Catherine, mi porrista número uno. Gracias por alentarme constantemente, acomodarte a mis inusuales horas de trabajo, y apoyar con mi redacción en inglés. También quiero agradecer a mis padres, Lupe y Fernando, y a mi hermano Nano por sus consejos y por creer siempre en mí.

Un agradecimiento muy especial para Cecilia Rivara, mi profesora guía en esta tesis, quien me aceptó como estudiante de doctorado y me transmitió sus conocimientos. Muchas gracias por tu disposición e interés en esta investigación. Quiero agradecer tu constante apoyo a lo largo de estos años, tanto en lo académico como en lo personal.

Quiero agradecer a los miembros de la comisión, Cecilia Rivara, Nancy Hitschfeld, Benjamín Bustos y Chee Yap, por su tiempo en revisar esta tesis y por los valiosos comentarios que hicieron posible mejorar este trabajo.

A CONYCIT y al DCC por el apoyo financiero y la oportunidad de seguir mis estudios de doctorado. A los profesores del DCC por su gran aporte en mi formación académica, en especial a Gonzalo Navarro y a Jeremy Barbay. A Angélica Aguirre por su apoyo con los varios trámites hechos durante mi ausencia, y a Ren Cerro por la revisión de este documento.

A mis amigos de la Colonia, mi familia en Chile, por los gratos momentos que compartimos estos años. A mis compañeros de oficina, Cecilia, Pedro y Teresa, por las discusiones académicas y la buena compañia.

Gracias a todos los que directa o indirectamente compartieron conmigo este viaje.

Contents

Chapter 1

Introduction

Nowadays, meshes are found in a wide range of applications and disciplines, such as computer graphics, scientific computing, solid modeling, image processing, geographical information systems, and numerical simulation, among others. Meshes are also an essential part in numerical methods for the solution of partial differential equations such as the finite element method. A mesh can be regarded as a decomposition of a geometric object or domain. The interdisciplinary application of meshes has created great interest in the study of novel and efficient techniques for mesh generation and refinement.

During recent decades the community of computational geometry has been actively involved in the development of *provably good* mesh algorithms – algorithms that are mathematically guaranteed to produce satisfying meshes. For most applications, the time required to process the geometry's decomposition depends both on the size and quality of the mesh. A good quality mesh generation algorithm should produce meshes of optimal size that conform to the geometry, while satisfying constraints on the shape and size of its elements. Mesh generation algorithms that offer theoretical guarantees on these aspects are desirable.

Longest-edge refinement algorithms proposed by Rivara and collaborators [38, 40, 44, 47] provided a fast, efficient technique for the generation and refinement of meshes. These techniques work with triangular meshes in two dimensions and tetrahedral meshes in three dimensions. They were specially designed to deal with the local iterative refinement of meshes as needed by numerical techniques such as adaptive finite element methods and multigrid algorithms, ensuring the construction of good-quality irregular and nested triangulations.

In this thesis we perform an extensive analysis of the theoretical and geometrical properties of the longest-edge refinement algorithms in 2-dimensional triangular meshes. We center our analysis in the longest-edge refinement algorithms: Lepp-Bisection and Lepp-Delaunay. We consider these algorithms the two most representative and practical implementations of the longest-edge strategy. Our goal is to analyze and work with these longest-edge algorithms as they originally appeared in the literature. In so doing, we provide the theoretical grounds for future work with modified versions of these algorithms.

The properties of the longest-edge bisection of triangles, the core operation behind the refinement process, are reviewed in Chapter 2. A discussion of previous studies of longest-edge refinement algorithms is presented in Chapter 3. We then focus the analysis on the longest-edge bisection algorithms, establishing new theoretical bounds on the cost of the algorithm and providing empirical experimentation that supports our findings, presented in Chapter 4. We perform an exhaustive geometrical analysis of the point selection strategy used by the Lepp-Delaunay algorithm in Chapter 5, providing bounds on the distance among points in the output triangulations, which finally translates into a proof for size-optimality. Finally, Chapter 6, draws together the main points of the thesis and addresses future work.

1.1 Meshes

A mesh can be regarded as a decomposition of an input geometry – and its features – into a set of simple *elements*. A mesh is usually composed of triangles or quadrilaterals in two dimensions, and tetrahedra or hexahedra in three dimensions. For the 2-dimensional case, the input geometry is commonly given by a *planar straight line graph* (PSLG), a finite set of points and a set of non-crossing segments. Figure 1.1 illustrates examples of meshes of domains with various shapes.

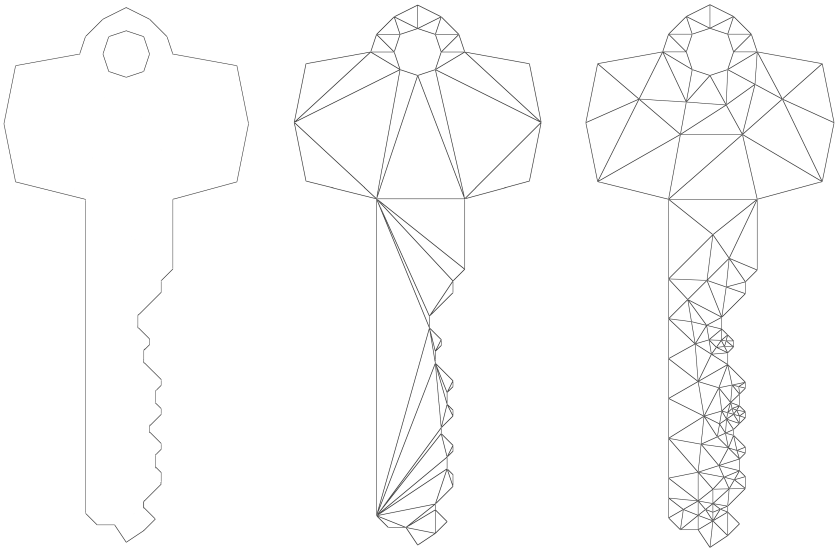
Triangulations Meshes composed of simplicial elements (triangles in two dimensions and tetrahedra in three dimensions) can more easily adapt to the requirements of many applications. In two dimensions a *triangulation* is a triangle mesh containing a set of adjacent triangles, their edges, and their vertices, and where triangles connect at common edge or a common vertex (i.e. a simplicial complex). Formally, the 2-dimensional triangulation of a set of points can be defined as follows.

Definition 1.1. Given a set of points S, a *triangulation* of S is the simplicial complex τ formed by the set triangles whose vertices correspond to S, and the union of the triangles in τ fills the convex hull of S. (Figure 1.2)

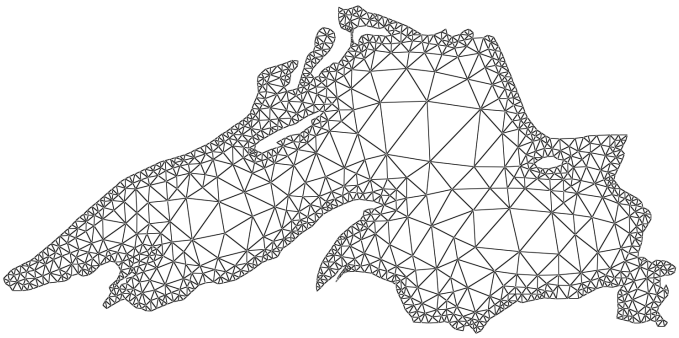
The rest of this thesis will consider the refinement algorithms that work over 2-dimensional triangulations. A study on non-simplicial meshes (using quadrilateral and hexahedral elements) can be found in the survey by Schneiders [56].

Unstructured Meshes Meshes can be categorized as structured or unstructured. In a *structured mesh* each vertex has an isomorphic local neighborhood (such as a regular square grid in two dimensions), while an *unstructured mesh* has arbitrarily varying local neighborhoods. Figure 1.3 illustrates an example of each category.

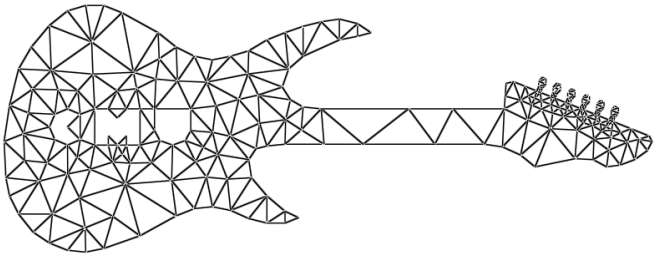
Structured meshes are simpler to store and manage since the topology can be easily calculated on-line versus unstructured meshes which explicitly store the topology information. On the other hand, structured meshes lack flexibility in capturing complex geometries, while



(a) (b) (c)



(d)



(e)

Figure 1.1: Examples of triangular meshes. (a) Domain in the shape of a key. (b) A mesh with poor quality elements. (c) The mesh was processed to have angles greater than 25°. (d) Mesh of the Lake Superior. (e) Mesh of a guitar.

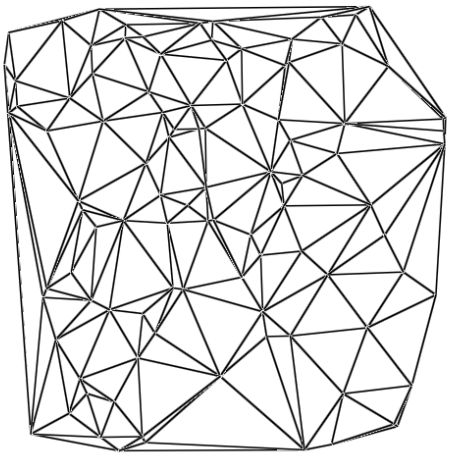
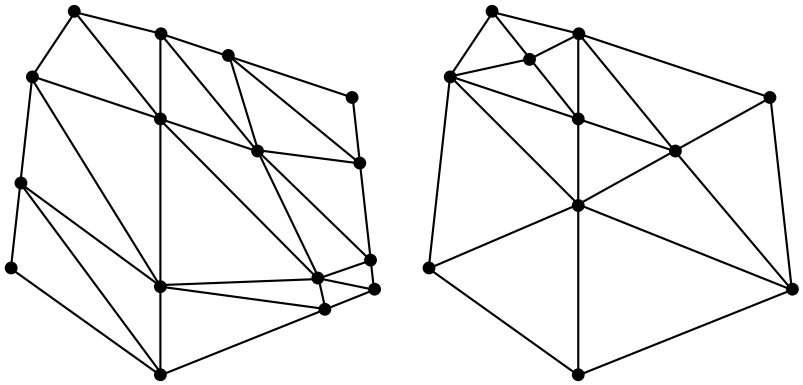


Figure 1.2: Triangulation of a set of points.



(a) (b)

Figure 1.3: (a) Structured mesh with its topology arranged as grid of triangles. (b) Unstructured mesh.

unstructured meshes are able to combine elements of different qualities and sizes, providing a more versatile decomposition.

1.2 Mesh Generation and Refinement

During the early 1980s the finite element method and its applications in many branches of engineering inspired the design of *ad-hoc* techniques for mesh generation. Later, computational geometry researchers became interested in designing algorithms that not only performed well in practice, but that also offered mathematical guarantees on the generation of satisfying meshes.

Mesh generation algorithms construct quality meshes by adding points to the initial data. Several algorithms have been proposed and studied over the last three decades. They can be roughly categorized into four classes:

- *Advancing front algorithms*, building the mesh from the geometry boundary and advancing inwards [14, 54];
- *Grid algorithms*, covering the geometry with a variable-resolution grid capable of capturing the input features [3, 7, 24].
- *Delaunay refinement algorithms*, based on circumcenter point insertions in Delaunay triangulations [25, 31, 53, 70];
- *Longest-edge refinement algorithms*, based on the longest-edge bisection of triangles (with non-Delaunay and Delaunay variants) [38, 40, 43, 45, 47].

A survey by Shewchuk [61] on unstructured mesh generation compares and discusses the advantages and properties of the first three categories. Longest-edge algorithms will be the subject of this thesis. Extensive research on practical mesh generation has been also performed. See e.g. [4, 8, 20].

Delaunay methods are arguably the most popular ones. They inherit the geometric properties of the Delaunay triangulation. This geometric structure allows the improvement of the minimum angle of the triangles, producing as coarse a mesh as possible. In spite of that, Delaunay methods based on a circumcenter strategy have to address implementation issues such as processing order, point location and robust operations. The interest in competitive algorithms with cheaper computational cost is still fueling the development of new algorithms.

1.2.1 Longest-Edge Refinement

Longest-edge refinement algorithms [38, 40, 44, 47, 65, 66] were designed to generate *refined* meshes that improve the resolution of the decomposition. Starting with a coarse (good quality) mesh, these algorithms produce a refined mesh of quality analogous to the initial mesh, achieving the element size required by the application (such as the finite element method). By incorporating a Delaunay approach, Delaunay-based longest-edge refinement algorithms can also achieve a desired element quality.

Longest-edge refinement algorithms serve as an efficient and fast alternative for mesh improvement and refinement: they show mostly linear complexity on the number of elements and use constant-time insertion operations. These algorithms are flexible, robust, provide scalable multithread implementations, and are suitable for both local and adaptive mesh refinement; properties desirable for applications such as the finite element method.

Longest-edge algorithms are practical and robust techniques for the refinement and generation of nicely graded triangulations. Originally based on the longest-edge bisection of triangles, they also incorporate local improvement strategies such as longest-edge propagating paths (Lepp-Bisection), and the Delaunay triangulation (Lepp-Delaunay).

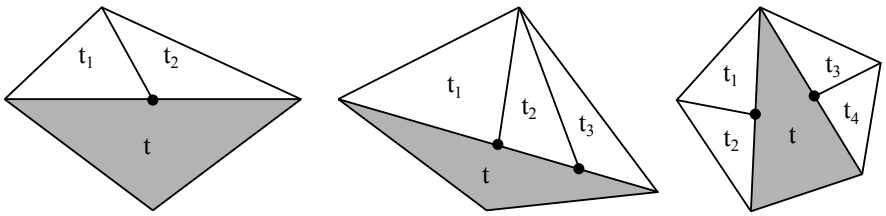


Figure 1.4: Examples of non-conforming triangles. In each case shaded triangle t shares a single edge with more than one triangle.

1.2.2 Properties of Mesh Generation Algorithms

In what follows we define some concepts and desirable properties of a mesh generation algorithm.

Conformity We say that a triangulation is a *conforming triangulation* or a *valid triangulation* if its elements entirely cover the input domain without overlapping each other, input segments are accurately represented, and the intersection of two adjacent triangles is either a common vertex or a common edge.

On the other hand, we say that a triangulation is *non-conforming* if it allows non-conforming triangles.

A triangle t is a *non-conforming triangle* if there exists a point along any edge of t that is not an endpoint of the edge. We call this point a *non-conforming point* and the edge affected by non-conforming points a *non-conforming edge*. (See Figure 1.4)

Size-Optimality The size of a mesh refers to the number of elements in the mesh. In order to improve the quality of the mesh, or to meet the element shape and size requirements, meshing algorithms introduce new points, also called *Steiner points*, that are not part of the input. The algorithm must be careful about the number of points introduced since this increases the size of the mesh and the computational cost of the application.

A mesh is considered *size-optimal* if the number of elements is within a constant factor times the minimum number in the smallest mesh offering the same shape bounds.

Geometric Quality A measure of quality provides a scalar value that estimates the suitability of an element's shape. The *quality* of a mesh is given by the quality of its elements. Common measures of quality include the internal angles (minimum or maximum angles), the aspect ratio (the length if the longest edge divided by the length of the shortest altitude), or the radius-edge ratio (the circumradius of the triangle divided by the length of the shortest edge).

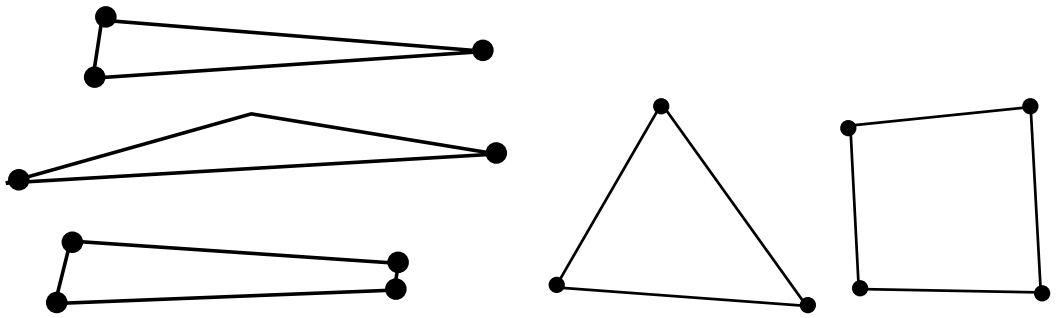


Figure 1.5: (a) Elements of bad quality with very small angles. (b) Good quality elements.

In 2-dimensional triangulations, the aforementioned measures are equivalent since they can be defined in terms of the minimum angle [27, 53]. The *minimum angle* allowed in the triangulation is a fairly general measure of quality: given an angle tolerance parameter θ, a triangle with smallest angle less than θ is regarded as a *bad quality triangle*.

The desired quality of the elements is ultimately determined by the application: some applications forbid large angles, some forbid small angles, some require fine and small-size elements, etc. A common example of elements with "ideal" shape are those nearly equilateral. Figure 1.5 illustrates some examples of elements of good and bad quality.

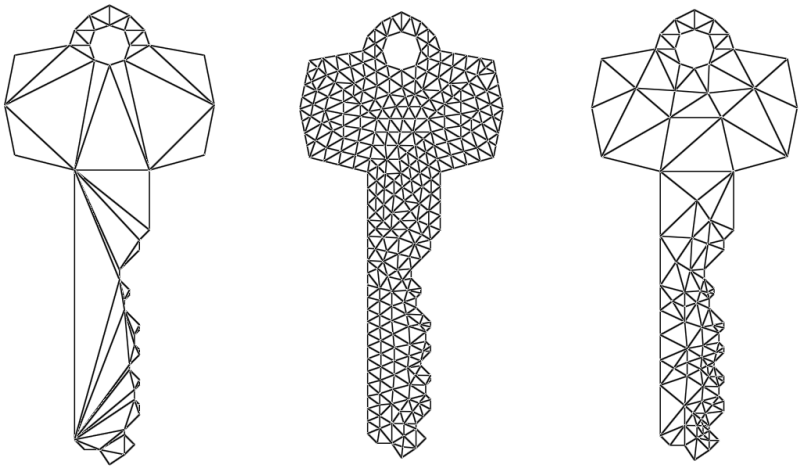
Good Grading Well *graded* meshes allow a smooth transition from small elements to large elements over a short distance without introducing small-angled elements. Because of the computational cost involved, well graded meshes are preferred over those with uniform element size. Figure 1.6 shows the effect of poor and good grading on the triangulation quality and on its number of elements.

Robustness Implementations of meshing algorithms are known to be susceptible to non-robustness introduced by numerical errors, which in turn produces topological errors and inconsistencies in the mesh. An algorithm sensitive to robustness issues tends to increase the difficulty of its implementation, threatening the complexity, termination and even accuracy of the results.

Robustness in geometric algorithms is studied by researchers in the area of Exact Geometric Computation, who are concerned about the non-robustness issues caused by numerical errors (see survey by Yap [73]).

1.2.3 Provably Good Mesh Generation

Provably good mesh generation methods provide mathematically guarantees to produce satisfying meshes. Ideally, the geometric quality of the elements is bounded and the meshes



(a) (b) (c)

Figure 1.6: (a) Triangulation with poor grading where small triangles neighboring big triangles produces bad quality elements. (b) A uniform triangulation of the same geometry showing the negative effect on the number of elements required to capture small elements (angles greater than 25°). (c) A well graded triangulation with the same shape bound as (b).

produced are size-optimal.

In two dimensions, grid methods were among the first provably good algorithms. Baker *et al.* [3] proposed the first algorithm to give shape guarantees, using an uniform square grid and producing meshes with angles between 13° and 90°. This work was later extended by Bern *et al.* [7], working over *quadtrees* meshes to guarantee well-shaped elements (with no angles smaller than 18.4°), grading, and size optimality.

Chew [31] presented the first provably good Delaunay refinement algorithm to produce uniform size-optimal triangulations with angles between 30° and 120°. Later, the seminal work of Ruppert [53] extended the ideas of Chew, presenting a Delaunay refinement algorithm to triangulate PSLGs with guarantees on grading, size and quality of the triangulation. Ruppert's algorithm offers theoretically guarantees on the minimum angle of 20.7°.

The work of Shewchuk [58, 60, 60] revisited the results of Ruppert and extended the application of the algorithm to 3-dimensional triangulations, also extending Ruppert's original methodology.

Recent studies on provably good Delaunay refinement algorithms aimed at reducing the size of the output triangulation. Ungor [69, 70] presented a strategy based on the iterative insertion of *off-centers* (instead of circumcenters). The algorithm maintains the same quality bounds of Ruppert's algorithm, but in practice, inserts about 40% fewer Steiner points than the circumcenter point selection strategy.

In this thesis we develop the first theoretical proofs on the Lepp-Bisection algorithm's size-optimality, and prove that the Lepp-Delaunay algorithm is a provably good mesh generation algorithm with outstanding practical properties.

1.3 Outline of the Thesis

The study will focus on the Lepp-Bisection and Lepp-Delaunay algorithms for 2-dimensional triangulation refinement. Our main goal is to establish new theoretical guarantees on the computational cost of these algorithms, and on the quality and size of the triangulations they produce.

Previous studies of longest-edge refinement algorithms established the geometrical properties of the algorithms and some quality bounds on the refined triangulations [38, 39, 43, 49]. These studies, reviewed in Chapter 3, guaranteed the generation of good-quality, unstructured triangulations which conform to the input geometry. Size-optimality and good grading were not theoretically guaranteed, although exhibited in practice [36, 40, 44, 49, 68]. Our previous analysis on longest-edge bisection algorithms showed that the cost of these algorithms can be bounded [5, 6].

The simplicity and locality of the longest-edge bisection operation allows fast generation of refined triangles. Chapter 2 briefly surveys previous mathematical properties and results of the longest-edge bisection of triangles, a key operation for most longest-edge refinement algorithms. We revisit the concept of *quasi-equilateral triangles*, triangles that behave like the equilateral triangle with respect to iterative longest-edge bisections. This concept is used throughout our analysis to provide proofs on the improvement of triangles.

The first contribution of this thesis is related to a classification of triangles based on the generation of quasi-equilateral triangles and non-similar triangles during iterative longest-edge refinement (discussed in Chapter 4). This taxonomy extends the one proposed by Gutierrez *et al.* [16] (reviewed in Chapter 2) for the iterative bisection of triangles. Our classification establishes the basis to define the behavior of longest-edge bisection algorithms, and facilitates a fine analysis of the algorithm.

The study of the longest-edge bisection algorithm in Chapter 4 focuses on the analysis of: (1) the refinement propagation to neighboring triangles, and (2) the cost associated with processing these triangles. This thesis provides the first bounds on the cost associated with the refinement of a triangle by longest-edge bisection. We show that the longest-edge bisection algorithm performs a constant number of bisections (and insertion of points) to produce conforming triangles. Our study of propagated refinement shows that the number of triangles affected depends on the geometry of the input, and that on average only a small number of triangles are processed (approximately two). Furthermore, we show that through the refinement process the propagation tends to affect only two triangles, making the cost of the algorithm asymptotically optimal.

We obtain new bounds on the size of the output triangulation and prove that it is at most a constant factor larger than the size of the input triangulations, providing the first complete proof of the algorithm's complexity. Our empirical analysis shows that the algorithm improves the global quality and grading of the triangulation.

The study of the Lepp-Delaunay algorithm in Chapter 5 is focused on the analysis of

the point selection strategy and the geometrical characterization of new edges created by the algorithm. We use the bound on the length of the new edges to prove the algorithm's termination and size-optimality.

We show that, due to the bound on the largest angle of terminal triangles, the distance from new points inserted by the algorithm to existing points of the triangulation (called insertion radius) is bounded by the geometry of the terminal triangles. We establish bounds on the length of the new edges based on the length of existing edges and on the circumradius of the triangles: the insertion radius is at least half the length of the smallest edge of a triangle, or at least half its circumradius.

We perform a detailed analysis of the insertion radius (exhaustive case analysis) and show the relationship between the insertion radius of new points and that of points in its neighborhood and along the propagating path. Based on this relationship we prove that the insertion radius of the points selected by the Lepp-Delaunay algorithm are at most a factor of two smaller than insertion radius of its predecessors. Using these results we show that the algorithm improves the quality of the triangles creating edges with bounded length: at most a factor of $\sqrt{3}$ smaller than the smallest feature in the input geometry. We extend this analysis to various bounds on the minimum angle, and show the improvement properties of the algorithm for each scenario.

We also develop new improvements on the point selection strategy (tuning of the algorithm) during the processing of triangles with constrained edges, reducing the number of points inserted near segments of the input and improving the algorithm's convergence. The improved point selection strategy allows the algorithm to process input geometries with constrained angles as small as 35°.

Finally, we apply the technique for the analysis of Delaunay refinement algorithms introduced by Ruppert [53] to prove that the Lepp-Delaunay algorithm produces size-optimal triangulations that are nicely graded. A strict analysis based on the local feature size function shows that the algorithm is guaranteed to meet Ruppert's conditions for a minimum angle parameter of roughly 14.48°. Based on the improvement properties of the algorithm, we extend the analysis to an angle parameter of 25.66°. In practice, size-optimality and good-grading are observed for angle parameters above 30°, although not theoretically guaranteed.

Chapter 2

Longest-Edge Bisection of Triangles

Longest-edge bisection of triangles is performed by joining the midpoint of the longest edge of a triangle with the opposite vertex. The geometrical properties of the longest-edge bisection of triangles inspired the study and development of mesh refinement techniques for adaptive finite element method [38–40]. The simplicity and locality of the longest-edge bisection operation permit a fast generation of refined, conforming triangulations with bounded minimum angles and improved average quality of triangles. In this chapter we review previous results on the longest-edge bisection of triangles.

2.1 Longest-Edge Bisection

Definition 2.1. Let $t(ABC)$ be a triangle of vertices A, B and C, and edges $AB \geq BC \geq CA$. Then, the longest-edge bisection of t is performed by joining the midpoint M of AB with the opposite vertex C (See Figure 2.1(a)).

Throughout this thesis the term bisection will refer to the longest-edge bisection of a triangle unless stated differently.

Given a triangle t, we say that the two sub-triangles obtained after the bisection of t are the *children* of triangle t. Analogously, triangle t is the *father* of these new-created triangles. In Figure 2.1(a), triangles AMC and MBC are the children of triangle ABC.

Definition 2.2. We call *iterative longest-edge bisection* (or simply *iterative bisection*) to the longest-edge bisection of a triangle, its children, the children of its children, and so on. We call *descendant* of triangle t to any new sub-triangle t^* created during the iterative longest-edge bisection of t. (See Figure 2.1(b))

Definition 2.3. Given two triangles t an t', we say that t is *similar* to t' if their angles are all identical. (See Figure 2.2)

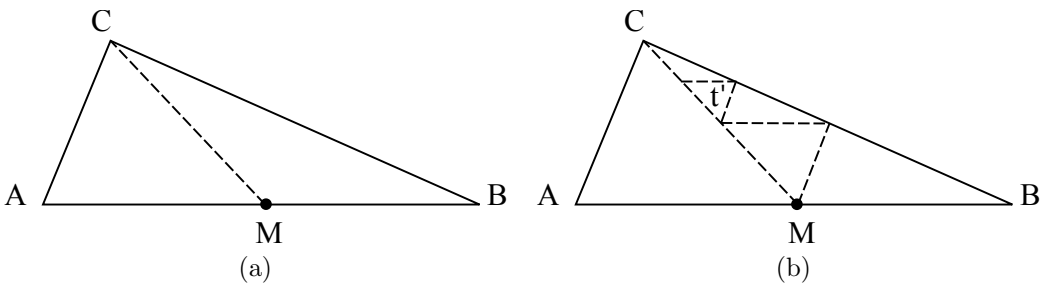


Figure 2.1: Triangle ABC with $AB \geq BC \geq CA$. M is the midpoint of AB. (a) Longest-edge bisection of triangle $t(ABC)$. (b) Repetitive longest-edge bisection of some descendants of t.

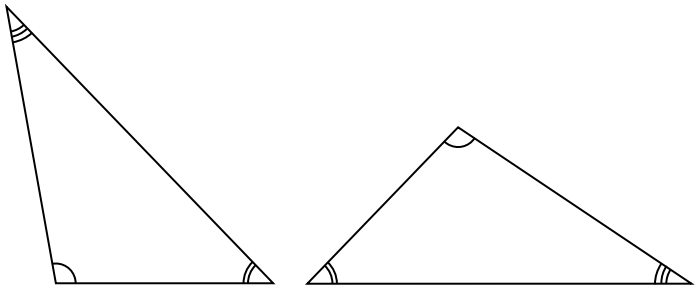


Figure 2.2: Similar triangles with identical angles.

The mathematical properties of the iterative longest-edge bisection of triangles were studied by Rosenberg and Stenger [52], Kearfott [21], Stynes [64], and Adler [1]. These results can be summarized in the following properties.

Property 2.4. *For any triangle t of smallest angle α, the iterative longest-edge bisection of t only produces triangles with any interior angle greater than or equal to $\alpha/2$.*

Property 2.5. *Every triangle generated by longest-edge bisection is similar to one of a finite number of associated non-similar triangles.*

Property 2.6. *The iterative longest-edge bisection of a triangle t monotonically increases the area of t covered with quasi-equilateral triangles.*

The angle properties of the children triangles of t obtained by iterative longest-edge bisection were further studied by Rivara and collaborators [41,47,63]. The following theorem summarizes the angle properties of the longest-edge bisection.

Theorem 2.7. *Let $t_1(AMC)$ and $t_2(MBC)$ be the children triangles of a triangle $t(ABC)$ as shown in Figure 2.3. Following this notation, then:*

a) $\alpha_1 \geq \alpha_0/2$

b) If t is obtuse, $\alpha_1 \geq \alpha_0$

c) $\beta_2 \geq 3\alpha_0/2$

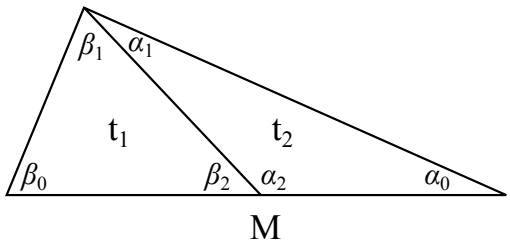


Figure 2.3: Angle properties after the longest-edge bisection of a triangle.

d) If t is obtuse, $\beta_2 \geq 2\alpha_0$

e) If t is acute $\beta_1 > \alpha_0$ if $\alpha_0 < 30°$

f) $\beta_2 = \alpha_1 + \alpha_0$

2.1.1 Quasi-Equilateral Triangles

We call a *quasi-equilateral triangle* to any triangle that behaves like an equilateral triangle with respect to its iterative longest-edge bisection, reproducing its bisection patterns.

Definition 2.8. Let $t(ABC)$ be a triangle with longest edge AB ($AB \geq BC \geq CA$) and M the midpoint of AB. Triangle t is considered quasi-equilateral if the following conditions apply:

i) $AC \geq AB/2, CM$

ii) $CM \geq CB/2$

For quasi-equilateral triangle $t(ABC)$ of Figure 2.4, the longest-edge bisection of its children, triangles AMC and MNC, is respectively performed by the edges AC and MC. Note that these bisections only produce edges parallel to the edges of initial triangle ABC. This implies that at most four similarly distinct triangles are produced:

- ABC (similar to triangles AMO, MBN),
- MBC (similar to triangle PNC),
- AMC (similar to triangle MNP), and
- MNC (similar to triangle OMC).

Any further longest-edge bisections on the descendants of t will only produce triangles similar to one of these four triangles.

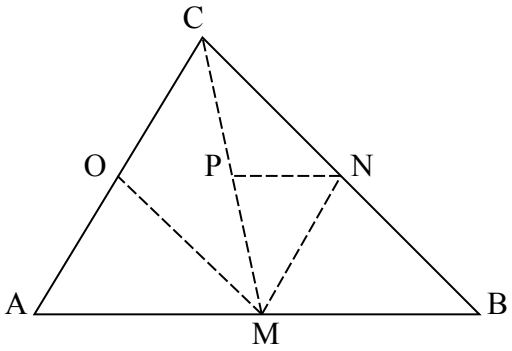


Figure 2.4: Quasi-equilateral triangle $t(ABC)$ and its non-similar descendants.

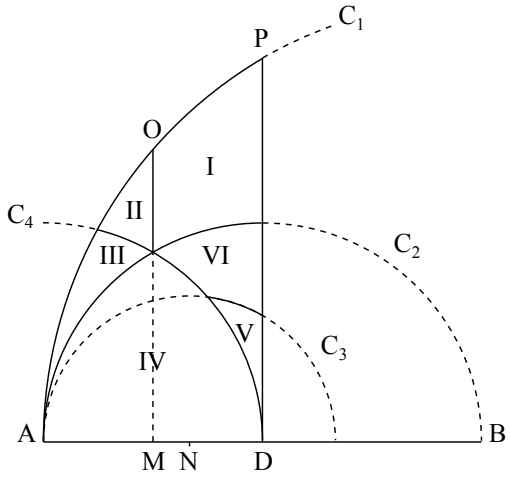


Figure 2.5: Regions defining the classification of triangles $t(ABC)$. Virtual vertex C lies in one of the regions defining a triangle t with $AB \geq BC \geq CA$.

2.2 Similarity Classes and Iterative Bisection of Triangles

Gutierrez *et al.* [16] studied the complexity of the iterative bisection of a triangle, introducing a taxonomy of triangles in the plane based on the evolution and the number of non-similar triangles produced during iterative bisection.

This study defined six classes (or *regions*) of triangles considering the geometric position where vertex C of a triangle $t(ABC)$ lies considering that $AB \geq BC \geq CA$ (Figure 2.5). Edge AB represent the longest edge of the triangle and point D its midpoint. Point M represents the midpoint of AD while point N is such that $AN = AB/3$. Both straight lines MO and DP are perpendicular to AB. Arc C_1 belongs to the circle of center B and radius AB, arc C_2 belongs to circle of center D and radius AD, arc C_3 belongs to circle of center N and radius AN, and arc C_4 belongs to circle of center A and radius AD. Then, vertex C lies in the region bounded by arc C_1 and lines DP and AD.

From the diagram shown in Figure 2.5 we can observe:

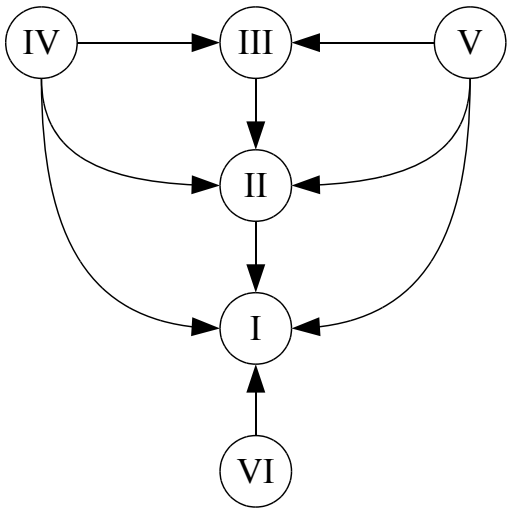


Figure 2.6: Directed graph showing the transition of the new triangles generated by iterative longest-edge bisection.

- Right-angled triangles have point C lying over arc C_2.
- The area above arc C_2 corresponds to acute triangles, while the area below it correspond to obtuse triangles.
- The equilateral triangle corresponds to triangle ABP.
- Isosceles triangles have point C lying over arc C_1 or line DP.
- Quasi-equilateral triangles belong to Regions I and VI, with Region I corresponding to acute quasi-equilateral triangles and Region VI to obtuse quasi-equilateral triangles.

Note that the generation of new non-similar triangles stops when quasi-equilateral triangles are obtained. Gutierrez *et al.* proved that the number of steps required until no new non-similar triangles are generated is upper bounded by $O(\frac{1}{\alpha})$, where α is the smallest angle of initial triangle t and the omitted constant is given by the largest angle.

Figure 2.6 shows the path that new non-similar triangles follow until their convergence to quasi-equilateral triangles. During the iterative bisection, a non-similar triangle could stay in a node several steps before moving to a different region (e.g. the bisection of a Region III triangle could produce several non-similar region III triangles before producing a non-similar region II triangle).

Figure 2.7 illustrate the classification of Figure 2.5 with three different positions for point C.

The following lemmas determine the number of new non-similar triangles produced during the iterative longest-edge bisection of triangles from each region.

Lemma 2.9. *The iterative bisection of a triangle $t(ABC)$ generates a finite number of non-similar triangles that move throughout the six regions of Figure 2.5. Furthermore, Region I*

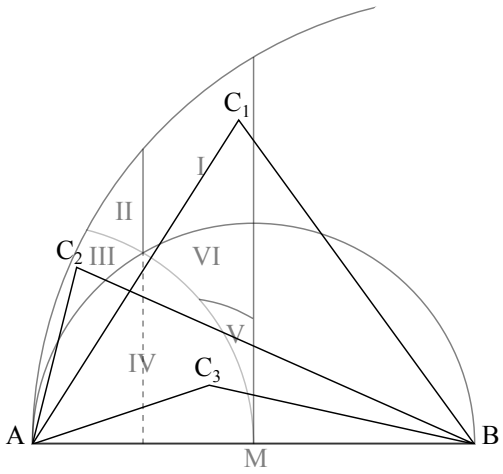


Figure 2.7: Triangles $t_1(ABC_1)$, $t_2(ABC_2)$ and $t_3(ABC_1)$ correspond to region I, III and IV respectively.

or VI triangles generate at most four non-similar triangles, all of which will still belong to either Region I or VI.

The proof of Lemma 2.9 is based on the behavior of the iterative longest-edge bisection for each region. A detailed discussion of the proof can be found in [16].

Figure 2.8 shows a region I triangle $t(ABC)$ and the sequence of triangles generated after five bisections. The initial bisection (Figure 2.8(a)) generates the two non-similar triangles $t_1(AMC)$ and $t_2(BCM)$. The bisection of t_1 shown in Figure 2.8(b) generates one non-similar triangle $t_{1,1}(OMC)$ since shaded triangle $t_{1,2}(AMO)$ is similar to initial triangle t. The bisection of t_2 shown in Figure 2.8(c) generates two triangles similar to triangles $t_{1,1}$ and $t_{1,2}$. Finally, the bisection of $t_{1,1}$ generates two triangles which are similar to triangles t_1 and t_2, terminating the generation of new non-similar triangles as shown in Figure 2.8(b). Bisections on the descendants of triangle t_2 will produce triangles similar to existing ones.

For Regions II through V triangles, some new non-similar triangles are produced until triangles of Regions I are created. Note that for some triangles from these regions the bisection of the equivalent to triangle $t_1(ADC)$ in Figure 2.8 needs to be performed either on the edge CD or AD, while the bisection of $t_{2,1}(CDN)$ can be performed on edge CN.

Lemma 2.10. *The iterative bisection of triangles from regions II, III, IV and V produces a sequence of new non-similar triangles until triangles of region I or VI are obtained.*

Lemma 2.10 refers to the convergence of the number of non-similar triangles produced iterative bisections, which finishes with the appearance of Region I or VI triangles. Figure 2.6 also illustrates this behavior.

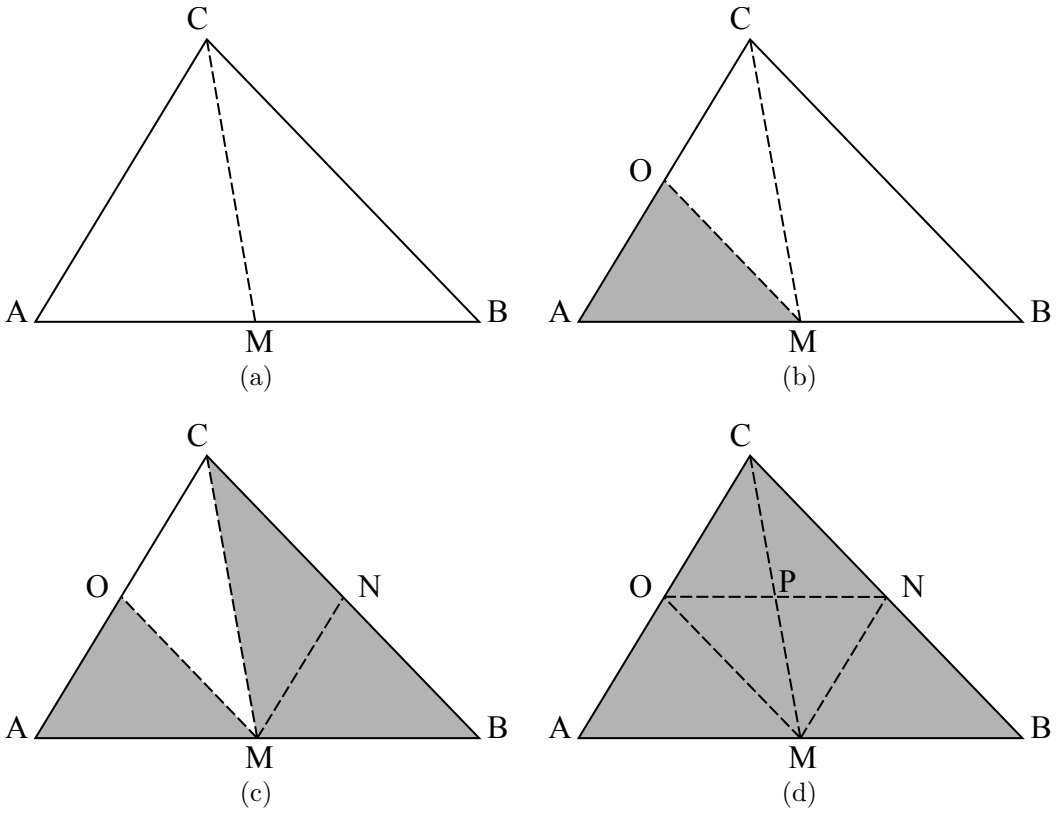


Figure 2.8: Iterative bisections of a region I triangle showing the generation of similar triangles (shaded areas).

Chapter 3

Longest-Edge Refinement Algorithms

Longest-edge refinement algorithms were designed to deal with the local iterative refinement of triangulations as needed for applications such as the finite element method. The simplicity and locality of the longest-edge refinement operation allows the fast generation of refined, conforming triangulations. Additionally, the processing order of the target triangles is irrelevant, as opposed to other well-known refinement algorithms, making the longest-edge refinement algorithms suitable for parallel environments.

The **longest-edge bisection algorithms** [38, 40, 42, 50] are used to locally refine coarse triangulations as needed by external applications. They perform a local refinement based on the bisection of triangles by their longest edge. The mathematical properties of the longest-edge bisection guarantee the generation of conforming triangulations with bounded minimum angles and improved quality of elements.

The **Lepp-Delaunay algorithm** [40, 41, 44, 45] extended the ideas of longest-edge bisection algorithms by also maintaining a Delaunay triangulation of the input. These algorithms are used to generate well-graded quality triangulations of planar straight-line graph geometries, supported by the mathematical properties of both the longest-edge bisection of triangles and the Delaunay triangulation.

Longest-edge bisection algorithms have been used in finite element software for partial differential equations [20, 29] and generalized for 3-dimensional refinement [30]. Empirical evaluations on longest-edge bisection algorithms had been previously performed [19,50,51,68], as well as for the Lepp-Delaunay algorithm in two and three dimensions [44, 49].

In this chapter we review the longest-edge refinement algorithms proposed by Rivara and collaborators. We review the properties and behavior of the original longest-edge bisection algorithm (Section 3.1). We discuss the most efficient reformulation of the algorithm, the Lepp-Bisection algorithm (Section 3.2), which is based on the longest-edge propagating paths formed by the longest-edge refinement strategy. Finally, we discuss the properties of the Lepp-Delaunay algorithm (Section 3.3), a longest-edge refinement algorithm that also maintains a Delaunay triangulation of the input and allows the elimination of bad-quality elements.

3.1 The Longest-Edge Bisection Algorithm

The original longest-edge bisection algorithm proposed by Rivara [38] was designed to work over an input conforming triangulation of acceptable geometric quality in which a set of triangles needs to be refined. A triangle is considered *refined* after it has been subdivided at least once. Then, the algorithm performs the longest-edge bisection of the *target triangles* (triangles selected for refinement) and some of their neighbors and descendants to produce a refined conforming triangulation.

After the longest-edge bisection of a triangle, the longest-edge neighbor triangle could become a non-conforming triangle (defined in Section 1.2.2). The algorithm performs additional bisections on neighbors and descendants to maintain a conforming triangulation. We call this behavior *propagation* of the refinement.

Algorithm 3.1 describes the longest-edge bisection algorithm. We present a modified description of the original algorithm presented in [38] in order to highlight the generation and treatment of non-conforming triangles. The algorithm bisects the target triangles and any non-conforming triangle in the triangulation until it is made conforming again.

Algorithm 3.1 Longest-edge bisection algorithm

Input: A quality triangulation τ and a set S_{target} of triangles to be refined
Output: A quality triangulation τ' such that each triangle $t \in S_{\text{target}}$ has been refined

```
1: for each triangle t in S_target do
2:    Add t to S_proc, the set of triangles that need to be processed
3:    while S_proc ≠ ∅ do
4:       Choose a triangle u from S_proc
5:       Triangle v is the longest-edge neighbor of u
6:       Perform the longest-edge bisection of u, producing children triangles u_1 and u_2
7:       Add to S_proc any non-conforming children of u
8:       if v is a non-conforming triangle then
9:          Add v to S_proc
10:      end if
11:      Remove u from S_proc {also remove u from S_target if it was selected for refinement}
12:   end while
13: end for
```

Figure 3.1 illustrate the refinement process of Algorithm 3.1.

Due to propagated refinement, triangles in S_{target} could be indirectly refined if they become non-conforming during the processing of set S_{proc}. This scenario does not affect the performance of the algorithm (nor the output triangulation) since the number of bisections remain the same regardless of the order in which set S_{target} is processed.

In practice, managing the non-conforming points and the associated non-conforming triangles is a complex task. As it is later discussed in Section 3.2, the Lepp-Bisection algorithm

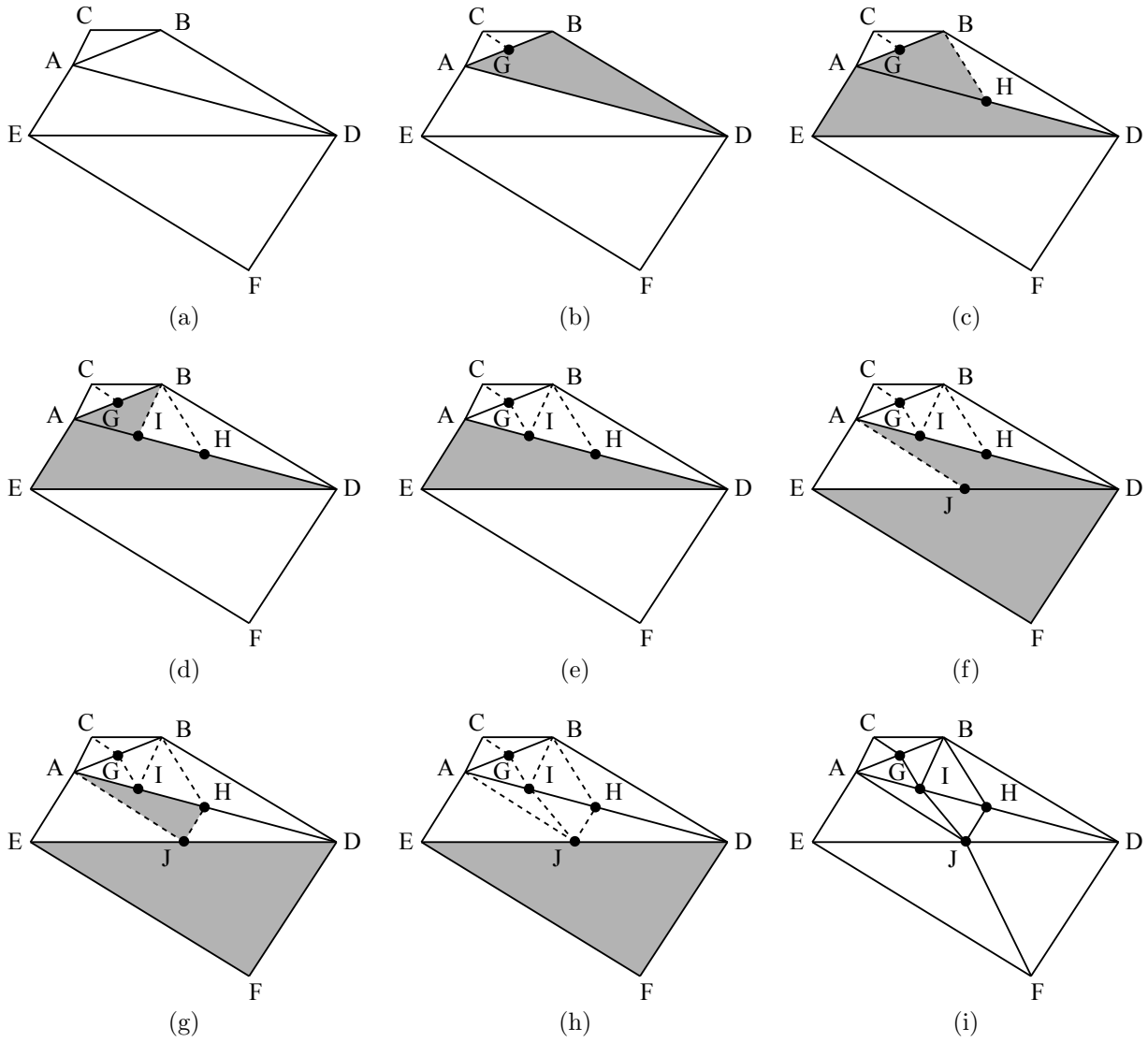


Figure 3.1: Example of the longest-edge bisection algorithm described in Algorithm 3.1 with $S_{\text{target}} = \{t(ABC)\}$. (a) Initial triangulation. (b) - (h) Refinement process showing triangles belonging to set S_{proc} (shaded in gray). (i) Final triangulation.

eliminates this burden, reformulating the longest-edge bisection algorithm to avoid the creation of non-conforming triangles during propagated refinement.

3.1.1 Robustness in Longest-Edge Refinement

An important advantage of bisection-based algorithms over known Delaunay-based refinement methods is their robustness: the longest-edge bisection strategy do not depend on complex computations. Also, the insertion of the midpoint of a terminal edge does not require additional operations to locate the triangles affected. As discussed in Section 1.2.2, robustness issues are very important in many applications where computation time and accuracy are critical [55, 73].

For example, if the points of the input triangulation have coordinates described by dyadic numbers, then the triangulation produced by the longest-edge algorithm remains dyadic.

A *dyadic number* is a rational number whose denominator is a power of 2. The set of dyadic numbers may be denoted $\mathbb{Z}[\frac{1}{2}] = \{m2^n : m, n \in \mathbb{Z}\}$.

Dyadic numbers can be efficiently represented without error using machine numbers or libraries available in several programming languages [72]. Consequently, longest-edge refinement algorithms can be easily made completely robust from numerical errors.

3.1.2 Properties of Longest-Edge Bisection Algorithms

The following properties of the longest-edge bisection algorithms were initially discussed by Rivara [38]. These properties refer to the improvement of triangles and the termination of the algorithm.

Lemma 3.1. *The iterative and arbitrary use of the algorithms only produces triangles whose smallest interior angles are always greater than or equal to $\alpha/2$, where α is the smallest interior angle of the initial triangulation. Furthermore every triangle generated is similar to one of a finite number of reference triangles.*

Proof of Lemma 3.1 follows from the properties of the iterative bisection of triangles reviewed in Section 2.1 (Fact 2.4 and Lemma 2.9).

Lemma 3.2. *Longest-edge refinement algorithms always terminate in a finite number of steps with the construction of a conforming triangulation.*

Proof of Lemma 3.2 is based on the facts that (1) the propagation of refinement moves toward bigger triangles of the triangulation, and (2) every triangulation has the smallest angle bounded.

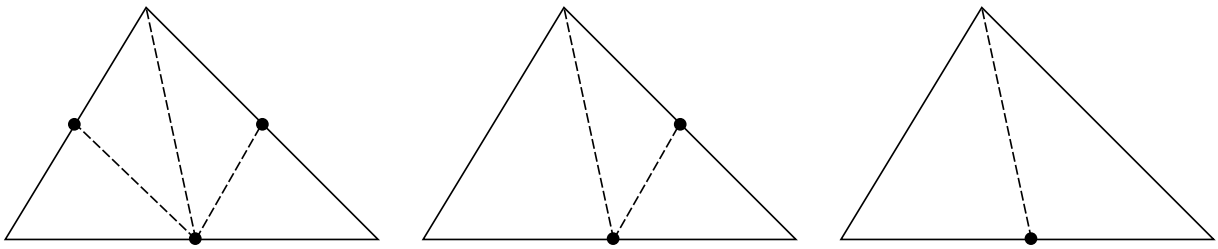


Figure 3.2: Partition patterns for the 4-Triangles refinement algorithm. (a) 4-triangles pattern. (b) 3-triangles pattern. (c) longest-edge bisection.

Lemma 3.3. *Any triangulation τ generated by means of the iterative use of the algorithms satisfies the following smoothness condition: for any pair of side-adjacent triangles $t_1, t_2 \in \tau$ (with respective diameters h_1, h_2), it holds that $\frac{min(h_1,h_2)}{max(h_1,h_2)} \geq k$, where $k > 0$ depends on the smallest angle of the initial triangulation.*

The smoothness property follows from the bound on the smallest angle of Lemma 3.1.

Lemma 3.4. *For any triangulation τ, the global iterative application of the algorithm (the bisection of all the triangles in the preceding iteration) covers, in a monotonically increasing form, the area of τ with quasi-equilateral triangles (with smallest angle $\geq 27.89°$).*

Lemma 3.4 refers to the progressive generation of more equilateral triangles during the repetitive application of the algorithm, which at the same time tends to isolate the worst angles.

3.1.3 Other Longest-Edge Bisection Algorithms

The 4-Triangles algorithm [38, 47] applies a 4-triangle partition pattern on target triangles, performing a longest-edge bisection of the triangle and connecting the midpoint of the longest-edge to the midpoints of the other two edges. Additional patterns of bisections are performed on neighboring triangles to maintain a conforming triangulation.

First, the algorithm refines a target triangle using the 4-triangle partition shown in Figure 3.2(a). Then, a local propagation to neighboring triangles is performed using either the 3-triangle partition shown in Figures 3.2(b) or a longest-edge bisection as shown in Figure 3.2(c). As in Rivara's original bisection algorithm, the propagation step assures that the output is a conforming triangulation.

This algorithm maintains the properties reviewed in 3.1, such as the bounds on minimum angles and on the number of non-similar triangles produced. Further studies on the 4-Triangle algorithm include those on mesh quality improvement [35], refinement propagation [66,68], and extensions to tetrahedral meshes [32,34] (although for the 3-dimensional scenario theoretical studies of size-optimality, termination and grading are still missing).

Other approaches based on the $n-$section of triangles include the 7-Triangles algorithm [23], longest-edge trisection [2, 33] and longest-edge quatersection [67]. Bisection approaches not based on the longest-edge criterion, such as the generalized bisection [17, 18], can also be found in the literature.

3.2 The Lepp-Bisection Algorithm

A practical drawback of Algorithm 3.1 is related to the management and storage of non-conforming triangles and non-conforming points. To solve this issue, Rivara introduced the concepts of *longest-edge propagating path*, *terminal triangles* and *terminal edges*, presenting the Lepp-Bisection algorithm [40] as an efficient reformulation of the original longest-edge bisection algorithm.

The Lepp-Bisection algorithm completely avoids producing non-conforming triangles. Starting at a target triangle, the algorithm "propagates" following a sequence of longest-edge neighbors until finding a pair of (terminal) triangles sharing the same (terminal) longest-edge. Then, the algorithm goes back over the sequence bisecting terminal triangles until the triangle that originated the propagation is bisected.

In comparison to Algorithm 3.1, the Lepp-Bisection algorithm navigates through set of triangles that need to be processed, S_{proc}, performing local bisections of pairs of triangles. This maintains the triangulation valid throughout the refinement process since non-conforming triangles are never generated.

3.2.1 Longest-Edge Propagating Paths

The Lepp strategy used by the longest-edge refinement algorithms studied in this thesis is based on the following concepts, originally introduced in [40].

Definition 3.5. An edge e_{term} is called a *terminal edge* in triangulation τ if e_{term} is the longest edge of every triangle that shares e_{term}. The triangles sharing e_{term} are called *terminal triangles*. For 2-dimensional triangulations, if e_{term} is shared by two terminal triangles then e_{term} is an interior edge; if e_{term} is shared by a single terminal triangle then e_{term} is a boundary edge.

Definition 3.6. For any triangle t_0 in τ, the *longest-edge propagating path* of t_0, $\text{Lepp}(t_0)$, is the ordered sequence $\{t_j\}_0^{N+1}$, where t_j is the neighbor triangle on the longest edge of t_{j-1}, and $\text{longest\_edge}(t_j) > \text{longest\_edge}(t_{j-1})$, for $j = 1, \ldots, N$. Edge $e_{\text{term}} = \text{longest\_edge}(t_{N+1}) = \text{longest\_edge}(t_N)$ is an interior terminal edge in τ and this condition determines N. Therefore, either e_{term} is shared by the couple of terminal triangles (t_N, t_{N+1}), or e_{term} is shared by a unique terminal triangle t_N with boundary (constrained) longest edge.

Figure 3.3 illustrates these concepts.

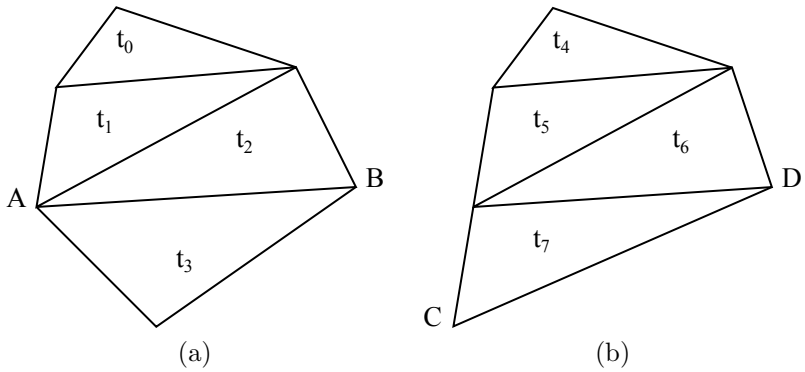


Figure 3.3: (a) AB is an interior terminal edge shared by terminal triangles $\{t_2, t_3\}$ of $\text{Lepp}(t_0) = \{t_0, t_1, t_2, t_3\}$. (b) CD is a boundary terminal edge with terminal triangle $\{t_7\}$ of $\text{Lepp}(t_4) = \{t_4, t_5, t_6, t_7\}$

3.2.2 Description of the Lepp-Bisection Algorithm

To refine a triangle t_0, the Lepp-Bisection algorithm perform two basic steps:

1) finds $\text{Lepp}(t_0)$ and the pair of terminal triangles t_N and t_{N+1} which share terminal edge e_{term}, and

2) performs the longest-edge bisection of t_N and t_{N+1} by the midpoint of e_{term}.

This process is repeated until initial triangle t_0 is eliminated. The original description of the Lepp-Bisection algorithm is presented in Algorithm 3.2.

Algorithm 3.2 Lepp-Bisection algorithm

Input: A quality triangulation τ and a set S_{target} of triangles to be refined
Output: A quality triangulation τ' such that each $t \in S_{\text{target}}$ has been refined

```
1: for each t in S_target do
2:    while t remains in τ do
3:       Find Lepp(t), terminal triangles t_i, t_j and terminal edge e_term {triangle t_j can be null
         for boundary e_term}
4:       Perform the longest-edge bisection of triangles t_i, t_j
5:       Update S_target {in case t_i or t_j were selected for refinement}
6:    end while
7: end for
```

A stack data structure can be used to avoid the repeated computation of $\text{Lepp}(t)$ needed to find the pair of terminal triangles: stored in a stack the triangles in $\text{Lepp}(t)$ as they appear, then processes the pair of (terminal) triangles on top of the stack. This modification was also discussed in [40], and effectively avoids a worst-case quadratic complexities since the triangles in the propagating path are only stored once. In practice, the average length of the propagating paths remains short, and implementations that do not contemplate the use of stacks also show good performance.

Algorithm 3.2 describes a practical implementation of the Lepp-Bisection algorithm using stacks for the management of the propagating paths.

Algorithm 3.3 Lepp-Bisection algorithm (practical implementation)

Input: A quality triangulation τ and a set S_{target} of triangles to be refined
Output: A quality triangulation τ' such that each $t \in S_{\text{target}}$ has been refined

```
 1: for each t in S_target do
 2:    Initialize a stack Q with t
 3:    while Q is not empty do
 4:       u is the triangle at the top of Q
 5:       v is the longest-edge neighbor triangle of u
 6:       if u is a terminal triangle then
 7:          Perform the longest-edge bisection of u
 8:          if v is not null then
 9:             Perform the longest-edge bisection of v {since v is also a terminal triangle}
10:          end if
11:          Remove u from Q
12:       else
13:          Add v to Q
14:       end if
15:    end while
16: end for
```

Note that:

- After bisecting a triangle, it is eliminated from the triangulation and replaced by its two descendants. This step removes the triangle from S_{target}, as well as any other triangle in the propagating path initially selected for refinement.
- The time complexity of the algorithm is $O(1)$ per split. Using standard amortization analysis we can charge adding a triangle to the stack (line 13) to the subsequent case when we bisect and remove this triangle (line 11).
- The stack requires linear space on the size of the Lepp.

Figure 3.4 illustrates refinement of triangle ABC. After finding Lepp(ABC), terminal triangles EDA and EFD are bisected by their longest edge (Figure 3.4(b)). Next, Lepp(ABC) is recomputed and terminal triangles AJD and ADB are bisected (Figure 3.4(c)). Again, Lepp(ABC) is recomputed and this time terminal triangles AJH and AHB are bisected (Figure 3.4(d)). After the last computation of Lepp(ABC) terminal triangles ABC and AIB are bisected (Figure 3.4(e)). The bisection of target triangle ABC stops the algorithm, returning the refined triangulation shown in Figure 3.4(f).

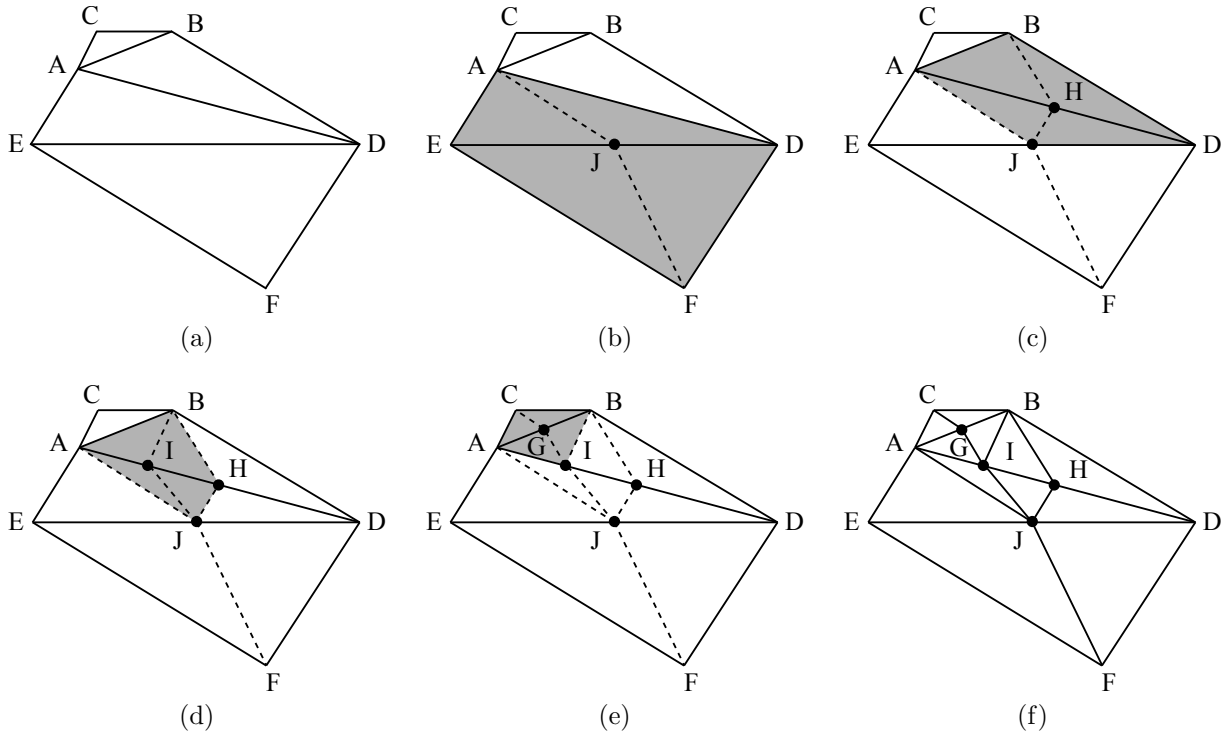


Figure 3.4: Refinement of triangle ABC using Lepp-Bisection algorithm. (a) Initial triangulation, $S_{\text{target}} = \{ABC\}$. (b) - (e) Refinement process showing terminal triangles (shaded in gray). (f) Final triangulation.

3.2.3 Iterative Bisection of Triangles

For every triangle $t \in S_{\text{target}}$ selected for refinement, the Lepp-Bisection algorithm calculates Lepp(t) and performs the longest-edge bisection of the corresponding pair of terminal triangles in the longest-edge propagating path. As described in Algorithm 3.2, this process is repeated until target triangle t is finally bisected.

It must be noted that, during this process, a triangle $t' \in$ Lepp(t) could be iteratively bisected (recall Definition 2.2) in order to eliminate non-conforming triangles and maintain a conforming triangulation. An example is shown in Figure 3.5.

3.3 The Lepp-Delaunay Algorithm

The Lepp-Delaunay algorithm extended the properties of the Lepp-Bisection algorithm by maintaining a Delaunay triangulation after bisecting the terminal triangles, effectively improving the quality of the triangulation. The algorithm deals with the automatic generation of triangulations of planar straight-line graph (PSLG) geometries, producing triangulations of bounded quality. The algorithm is supported by the mathematical properties of the longest-edge bisection of triangles and the Delaunay triangulation.

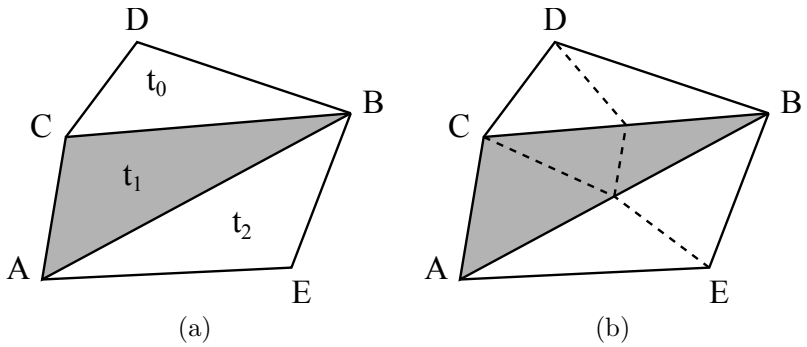


Figure 3.5: Shaded triangle $t_1(ABC)$ is iteratively bisected during refinement of target triangle $t_0(CBD)$. (a) Initial triangulation: Lepp(()t_0) = $\{t_0, t_1, t_2\}$. (b) Output triangulation: dashed lines represent new edges created by longest-edge bisection.

First, we review the concepts related to Delaunay triangulations; the geometric structure used by the Lepp-Delaunay algorithm. Then, we review the algorithm itself and some of the most important properties.

3.3.1 The Delaunay Triangulation

Delaunay refinement is based upon the geometric structure called *Delaunay triangulation*. It has been heavily used for mesh generation algorithms due to a very attractive geometric property: the Delaunay triangulation maximizes the minimum angle among all possible triangulations for a given set of points.

The Delaunay triangulation of a set of points S satisfies the *empty circumcircle property*, that is, the interior of any triangle's circumcircle does not contain any points in S. The circumcircle of a triangle is defined below.

Definition 3.7. Given a triangle t, we call *circumcircle*, or *circumscribing circle* of t to the circle which passes through every vertex of t. The *circumradius* of t is the radius of the triangle's circumcircle. The *circumcenter* of t is the point at the center of the triangle's circumcircle.

Definition 3.8. We say that a triangle is *Delaunay* if it satisfies the empty circumcircle property. Then, a *Delaunay triangulation* of a set of points S, is the triangulation of S where every triangle is Delaunay.

Figure 3.7 illustrates a Delaunay triangulation showing the empty circumcircle of every triangle.

Algorithms for the construction of 2-dimensional Delaunay triangulations in $O(n \log n)$ time can be found in the literature [12, 22, 57].

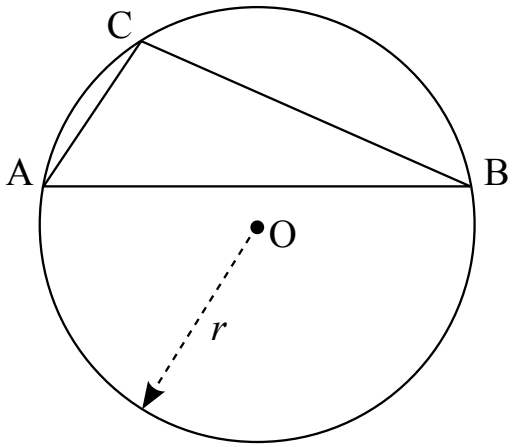


Figure 3.6: Triangle $t(ABC)$ and its circumcircle. Point O is the center of the circumcircle of radius r.

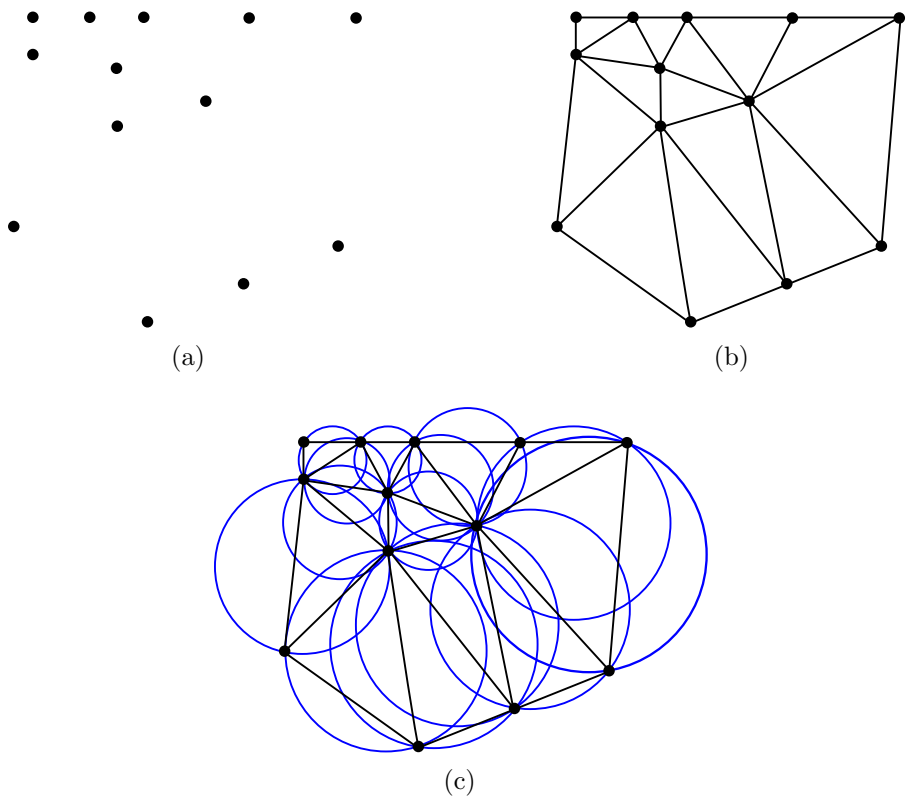


Figure 3.7: Example of a Delaunay triangulation. (a) Set of points. (b) Delaunay triangulation (c) Circumcircles (in blue) of every triangle.

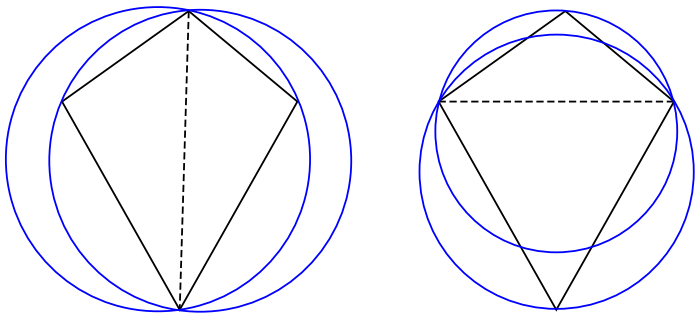


Figure 3.8: Two triangles before and after and edge-flip operation. (left) The triangles have not empty circumcircles. (right) The edge-flip operation produces two Delaunay triangles.

Edge-Flip Operations Due to its simplicity, edge-flip operations are widely used for incremental construction of Delaunay triangulations. After the algorithm adds a new point to the triangulation, it repeatedly performs edge-flip operations to locally enforce the Delaunay property .

Consider a pair of neighbor triangles sharing an edge e, the *edge-flip operation* consists of replacing e by the edge connecting the points opposite to e, as illustrated in Figure 3.8. If one of the triangles is not Delaunay, after an edge-flip operation the triangles produced are locally Delaunay. This operation increases the minimum angle.

Although in theory the number of edge-flip operations is a factor that affects the cost of insertion, in practice the number of edge-flips after the insertion of a point is a small constant [10, 15].

A review of the geometrical properties of the edge-flip operation and the proofs for the generation of Delaunay triangulations in two dimensions can be found in [9].

Constrained Delaunay Triangulations A constrained Delaunay triangulation (CDT) is similar to a Delaunay triangulation, but it works with domains composed of points and segments, ensuring that input segments are present in the triangulation. As with Delaunay triangulations, a CDT also maximizes the minimum angle among all possible triangulations for a given set of points and segments. The domain is defined by a planar straight-line graphs, defined below.

Definition 3.9. A *planar straight-line graph* (PSLG) is a set of points and segments that satisfies the following two properties: (1) the two endpoints of every segment must be present in the PSLG, and (2) segments can only intersect at their endpoints.

The CDT introduces the idea of *visibility*, relaxing the empty circumcircle property in order to respect the segments of a PSLG. Two points are visible to each other if there is no input segment intersecting a line connecting these two points. Then, a triangle is *constrained Delaunay* if its circumcircle does not contain points that are visible from the interior of the triangle. Finally, a *constrained Delaunay triangulation* of a PSLG is a triangulation in which every triangle is constrained Delaunay.

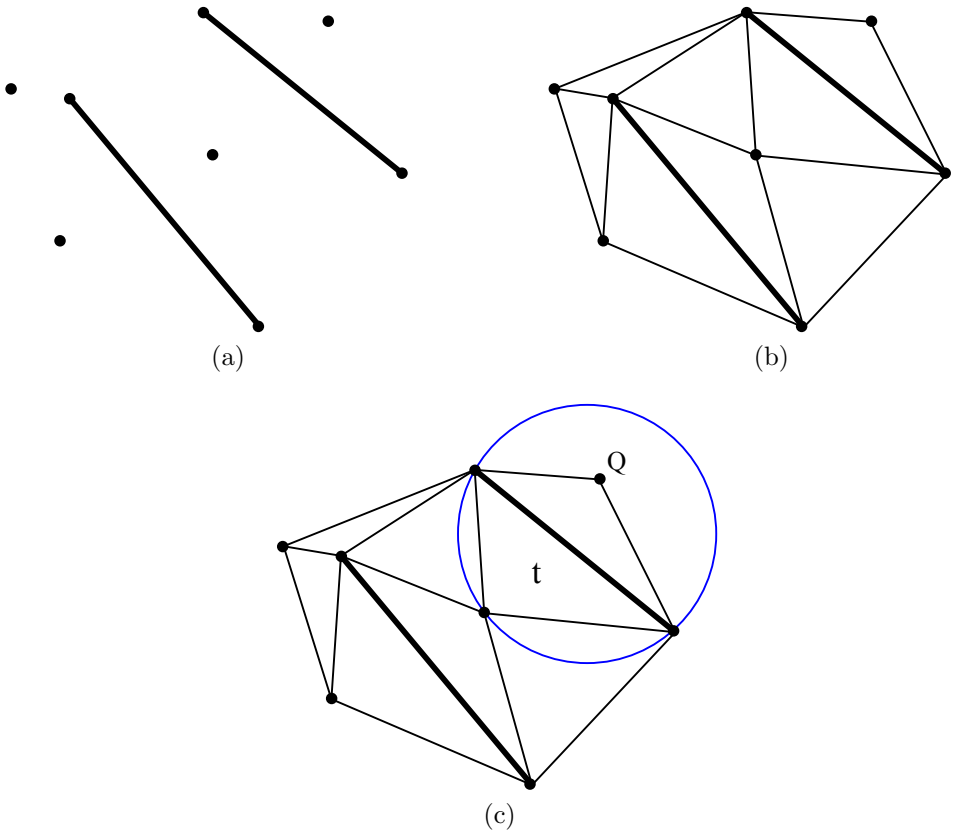


Figure 3.9: Example of a constrained Delaunay triangulation. (a) Set of points and segments. (b) A constrained Delaunay triangulation (bold lines represent segments). (c) Even though points Q lies inside the circumcircles of triangle t, Q is not visible from the interior of the triangle.

Figure 3.9 illustrates a constrained Delaunay triangulation of a set of points and segments. In Figure 3.9(c) point Q lies inside the circumcircle of triangle t. Triangle is *constrained Delaunay* and is accepted in the triangulation because point Q is not visible from the interior of t.

Constrained Delaunay triangulations can be constructed using edge-flip operations, making sure that segments are respected in the triangulation. Chew [31] presented an $O(n \log n)$ algorithm for the construction of CDTs.

3.3.2 Description of the Lepp-Delaunay Algorithm

Given an input PSLG $\mathcal{P}$ and a quality parameter ϵ, the Lepp-Delaunay algorithm produces a conforming constrained Delaunay triangulation τ of $\mathcal{P}$. Triangulation τ contains every point in $\mathcal{P}$, and every segment in $\mathcal{P}$ is a union of edges en τ.

The algorithm starts with the constrained Delaunay triangulation (CDT) of $\mathcal{P}$, progres-

sively creating new points and edges to attain the desired quality. The edges of τ forming part of an input segment are called *constrained edges*.

Any triangle in τ not satisfying ϵ is added to the set of target triangles to be refined. Then, for every target triangle t, the algorithm finds the terminal edge e_{term} associated with Lepp(t), and performs the constrained Delaunay insertion of the midpoint of e_{term}. Under certain circumstances, if there is a constrained edge associated with the terminal triangles, the algorithm inserts the midpoint of the constrained edge instead of the midpoint of e_{term}. The process is repeated until t is removed from the triangulation.

Algorithm 3.4 provides a simple description of the Lepp-Delaunay algorithm. Algorithm 3.5 describes the procedure for the point selection strategy used by the algorithm.

Algorithm 3.4 Lepp-Delaunay Algorithm

Input: PSLG polygon $\mathcal{P}$, quality parameter ϵ
Output: Constrained Delaunay Triangulation τ of quality defined by ϵ

```
1: Construct τ the CDT of P.
2: Find S_target ∈ τ, the set of bad quality triangles defined by ε
3: for each t in S_target do
4:    if t has longest edge or midsize edge constrained then
5:       Perform constrained Delaunay insertion of midpoint of the constrained edge
6:    else
7:       while t remains in τ do
8:          Find Lepp(t), terminal triangles t_i, t_j and terminal edge e_term {triangle t_j can be
            null for boundary e_term}
9:          Select point P using Point Selection Strategy
10:         Perform constrained Delaunay insertion of P into τ
11:      end while
12:   end if
13:   Update S_target {since additional target triangles could have been eliminated}
14: end for
```

Algorithm 3.5 Point Selection Strategy (midpoint)

Input: Terminal triangles t_i, t_j, terminal edge e_{term}
Output: Point P

```
1: Select point P, midpoint of e_term
2: if e_term is not constrained then
3:    if midsize edge of t_i is constrained and the longest-edge bisection of t_i produces a
      triangle with smallest angle less than 30° then
4:       P is the midpoint of the constrained midsize edge of t_i
5:    else
6:       Perform the same verification for triangle t_j
7:    end if
8: end if
```

Similar to the Lepp-Bisection algorithm, the computation of Lepp(t) can be efficiently managed by a stack data structure (Algorithm 3.3 in Section 3.2.2).

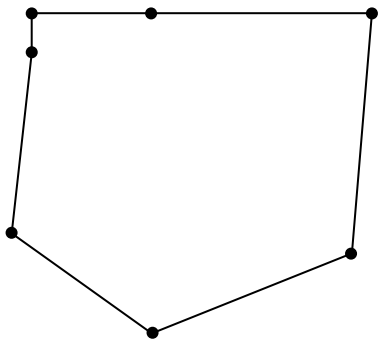


Figure 3.10: Example of a simple PSLG.

Figure 3.11 illustrates the Lepp-Delaunay algorithm in action. The input is given by the PSLG shown in Figure 3.10. Quality parameter ϵ is the minimum angle allowed in the triangulation, $\theta = 30°$.

Figure 3.11(a) shows the constrained Delaunay triangulation of the PSLG. Triangles t_0 and t_1 have angles below θ and are selected as target triangles. We select t_0 as the first target triangle to be refined. The algorithm is *order-independent*, so the actual order in which target triangles are processed is not relevant.

The algorithm computes Lepp(t_0) and selects the midpoint of terminal edge shared by terminal triangles t_2 and t_3 (Figure 3.11(b)). Figure 3.11(c) shows the triangulation obtained after the insertion of point 1. At this point of the process target triangle t_1 has been improved and is no longer considered a target triangle. Since t_0 has not been eliminated, the algorithm recomputes Lepp(t_0), selecting the midpoint of the longest edge of terminal triangle $t_{2'}$ (Figure 3.11(d)). Figure 3.11(e) illustrates the triangulation after the insertion of point 2. The process is repeated again, recomputing Lepp(t_0) and selecting the midpoint of terminal edge shared by terminal triangles $t_{1'}$ and $t_{2''}$ (Figure 3.11(f)). Figure 3.11(g) illustrates the triangulation after the insertion of point 3. Next time the algorithm computes Lepp(t_0), t_0 itself is a terminal triangle. The midpoint of the longest edge of t_0 is not selected for insertion because t_0 has constrained midsize edge and its longest-edge bisection produces a triangle with an angle smaller than 30°. In this case, the algorithm selects the midpoint of the constrained midsize edge (Figure 3.11(h)). The insertion of point 4 finally eliminates target triangle t_0 and terminates the process. Figure 3.11(i) illustrates the output triangulation with angles greater than 30°.

Some of the properties of the Lepp-Delaunay algorithm include:

Bounded Quality The set of target triangles S_{target} is defined by the *quality parameter* ϵ. The triangles in τ not satisfying ϵ are called *bad quality triangles* and included to S_{target} in order to be improved by the algorithm. A common quality parameter used in mesh refinement is the minimum angle θ allowed in the triangulation. Any triangle with smallest angle $\alpha \leq \theta$ is selected for refinement, therefore the output triangulation has no internal angles smaller than θ.

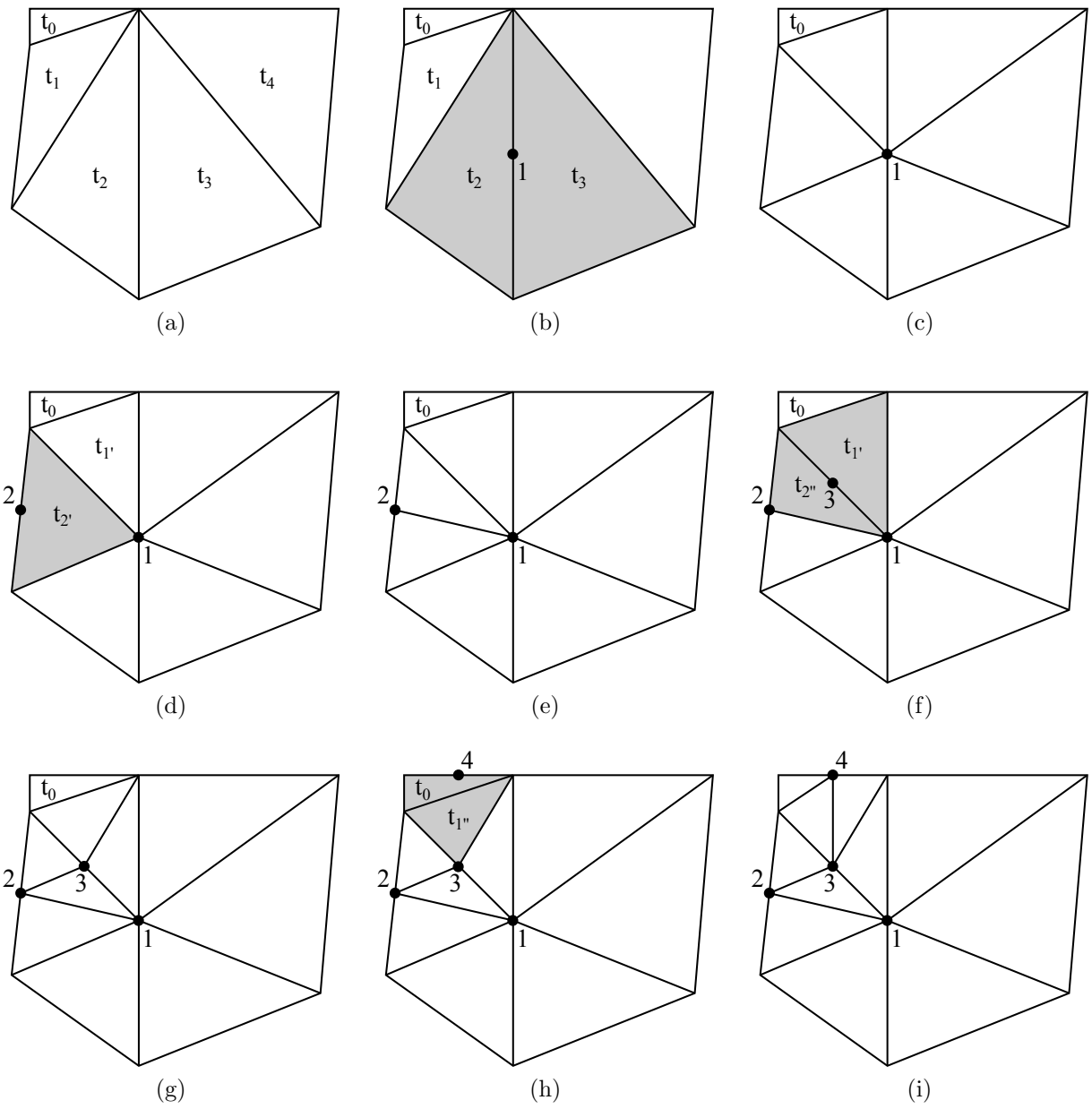


Figure 3.11: The complete refinement process of the Lepp-Delaunay algorithm for the PSLG in Figure 3.10. Quality parameter is the minimum angle allowed $\theta = 30°$. (a) Constrained Delaunay triangulation of the input PSLG. (b) - (h) Snapshots of each step of the process showing terminal triangles (shaded in gray) and the insertion of midpoints. (i) Final triangulation.

Point Selection Strategies For non-constrained terminal triangles the point selected for insertion corresponds to the midpoint of the terminal edge. The initial insertion of this point into the triangulation is equivalent to the longest-edge bisection of the terminal triangles. This is followed by a sequence of local edge-flip operations performed to produce a Delaunay triangulation.

Under certain circumstances, if a terminal triangle has a constrained midsize edge, the algorithm selects the midpoint of the midsize edge of the triangle. The initial insertion of this point is equivalent to the bisection of the triangle by its midsize edge, which is followed by a sequence of edge-flip operations to preserve the Delaunay property. As previously stated, the number of edge-flip performed after the insertion of a point is a small constant in practice.

The point selection strategy related to constrained edges was introduced to avoid infinite processing cycles, ensuring the algorithm's termination.

As we will discuss later in the thesis, the treatment given to constrained edges also improves the point distribution surrounding constrained edges. The angle threshold of 30° used to validate midsize-edge bisection implicitly bounds the distance from new points to constrained edges. This strategy is analogous to the treatment of *encroached edges* introduced by Ruppert [53].

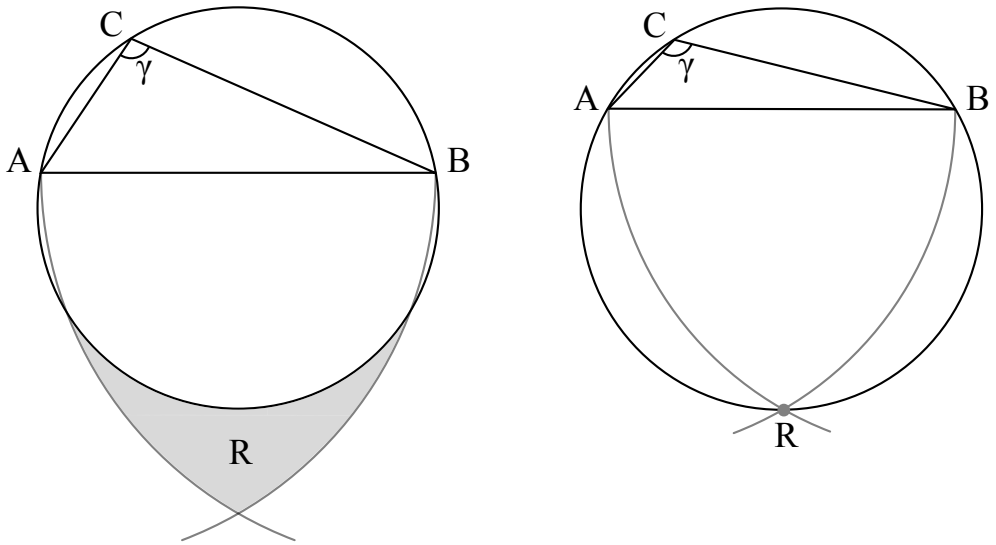
Implementation Issues As previously described, the Lepp-Delaunay algorithm selects the midpoint of an edge – either terminal or constrained – to be inserted into the triangulation.

Selecting the midpoint of an existing edge is a *robust* operation and does not require additional computations to find the triangles containing the point being inserted. This introduces an advantage over most Delaunay refinement algorithms, where point location operations penalize the computational cost of the insertion.

Furthermore, the midpoint strategy is not affected by robustness issues associated with the circumcircle test, as it is the case for circumcenter-based strategies commonly used by most Delaunay refinement algorithms.

Order-Independent For each bad quality triangle, the Lepp-Delaunay algorithm *propagates* the refinement towards a sequence of increasing neighboring triangles in order to maintain a good graded conforming triangulation. The Lepp propagation strategy inherited from the Lepp-Bisection algorithm makes the algorithm independent of the order in which the triangles are processed, or *order independent*, a property that simplifies the implementation of the algorithm and allows near-linear speed-up on parallel environments [51].

Generation of Quasi-Equilateral Triangles Another property inherited from the longest-edge bisection of triangles allows the Lepp-Delaunay algorithm to monotonically increase the area covered by quasi-equilateral triangles (Lemma 4.15 in Section 4.7.2).



(a) (b)

Figure 3.12: (a) Region R defines the area for point E of the longest-edge neighbor ABE. (b) Region R becomes a point when $\gamma = 120°$. In this case triangle ABE is necessarily equilateral.

Another factor that favors the generation of better quality triangles is the Delaunay insertion of points: any additional edge-flip operation performed by algorithm will locally maximize the minimum angles of the triangles.

3.3.3 Delaunay Terminal Triangles

We call *Delaunay terminal triangles* to the pair of terminal triangles of a Lepp. Since the triangulation satisfies the Delaunay property, the following important properties hold for any pair of terminal triangles [40].

Proposition 3.10. *Every pair of Delaunay terminal triangles in a triangulation τ have their largest angle $\gamma \leq 120$.*

Proposition 3.11. *Let t_i and t_j be a pair of Delaunay terminal triangles. Then, either one of them is an obtuse triangle, or both are acute triangles.*

The demonstration is based on the following observation. Let R be the region for the candidate longest-edge neighbor of triangle ABC. This region is reduced to a point when $\gamma = 120°$. Otherwise, for triangles with $\gamma > 120°$ no longest-edge neighbor could satisfy the Delaunay property. Figure 3.12 illustrates these scenarios.

The next corollary follows from the previous propositions.

Corollary 3.12. *For any pair of Delaunay terminal triangles t_i and t_j, t_i is an obtuse triangle if and only if its largest angle $\gamma_{t_i} \leq 120°$ and t_j is an acute triangle.*

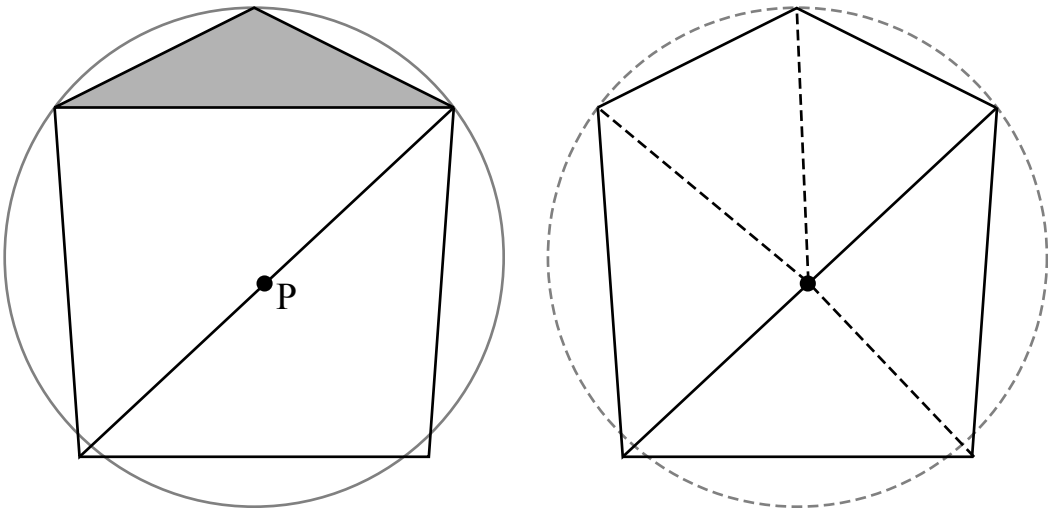


Figure 3.13: Elimination of obtuse triangles with $\gamma > 120°$. (a) Shaded triangle is candidate for refinement. Gray triangle corresponds to its circumcircle. Point P is the point selected for insertion. (b) Triangulation produced by the algorithm.

3.3.4 Elimination of Too-Obtuse Triangles

We call a triangle with largest angle greater than 120° a *too-obtuse triangle*. By Proposition 3.10, a too-obtuse triangle cannot become terminal triangle and consequently is never eliminated by the insertion of its longest edge's midpoint. Instead, these triangles are indirectly eliminated by edge-flip operations.

Any triangle t with largest angle $\gamma > 120°$ is indirectly eliminated by the terminal-edge or boundary point insertion over a pair of terminal triangles $(t_i, t_j) \in \text{Lepp}(t)$ and $t_i, t_j \neq t$. Whenever the midpoint of a terminal edge lies inside the circumcircle of t, an edge-flip operation is performed to maintain the Delaunay property, eliminating the bad-quality triangle and locally improving the triangulation; see Figure 3.13.

The following corollary refers to the elimination of too-obtuse triangles.

Corollary 3.13. *A target triangle t with largest angle $\gamma_t > 120°$ is indirectly improved by terminal-edge or boundary point insertion over another triangle in* $\text{Lepp}(t)$.

In practice, too-obtuse triangles are eliminated early during the processing of the target triangles. Too-obtuse triangles are often associated with large circumcircles, hence they are likely to be indirectly improved when a nearby triangle is processed.

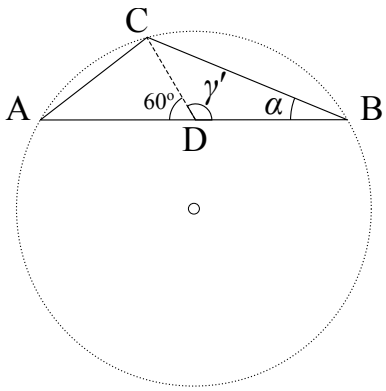


Figure 3.14: Obtuse terminal triangle $t(ABC)$ with largest angle $\gamma = 120°$ and smallest angle $\alpha = 22.24°$. After the insertion of point D, angle $\gamma' = 120°$.

3.3.5 A Rare Scenario of Infinite Processing Cycles

There is a special configuration of triangles that produces an infinite processing cycle for Algorithm 3.4 during the bisection of a pair of terminal triangles. This behavior is associated with the generation of a new bad quality triangle which is similar to an obtuse target triangle. Studies on this rare scenario [41, 46, 49] analyzed the conditions that produces the infinite cycle. The following theorem summarizes these results.

Theorem 3.14. *Terminal-edge point insertion operations over an obtuse triangle t with smallest angle $22.24° < \alpha < 30°$, whose medium-size edge becomes a terminal edge throughout the refinement process, produces a (smaller) similar triangle to t. This property induces an infinite processing cycle.*

Proof. Consider the obtuse possible terminal triangle $t(ABC)$ with largest angle $\gamma = 120°$; see Figure 3.14. After the insertion of point D (midpoint of longest edge AB), midsize edge BC could become terminal triangle only if the largest angle γ' of obtuse triangle $t'(DBC)$ is smaller than 120°. For a triangle t with smallest angle $\alpha = 22.24°$, $\gamma' = 120°$, the maximum possible angle for Delaunay terminal triangles. Then, for $\alpha > 22.24°$ the infinite processing cycle could be induced, while for $\alpha < 22.24°$ this scenario cannot arise. □

Figure 3.15 illustrates a configuration of triangles that produces an infinite processing cycle. Terminal triangles $t(ABC)$ shares longest edge AB with and triangle $t_1(AEB)$ and midsize edge CA with and triangle $t_2(BFC)$, as shown in Figure 3.15(a). The algorithm then introduces midpoint D_1 of terminal edge AB (Figure 3.15(b)). Then, assuming that no edge-flip operation removed edge BC, if some propagating path reaches triangle t_2, the insertion of point D_2 produces a triangle $t'(D_1BD_2)$ which is similar to triangle t (Figure 3.15(c)).

Non-cycle Assumption In order to avoid the scenario of infinite processing cycles, we use the same strategy used for refining triangles with constrained midsize edges. Following

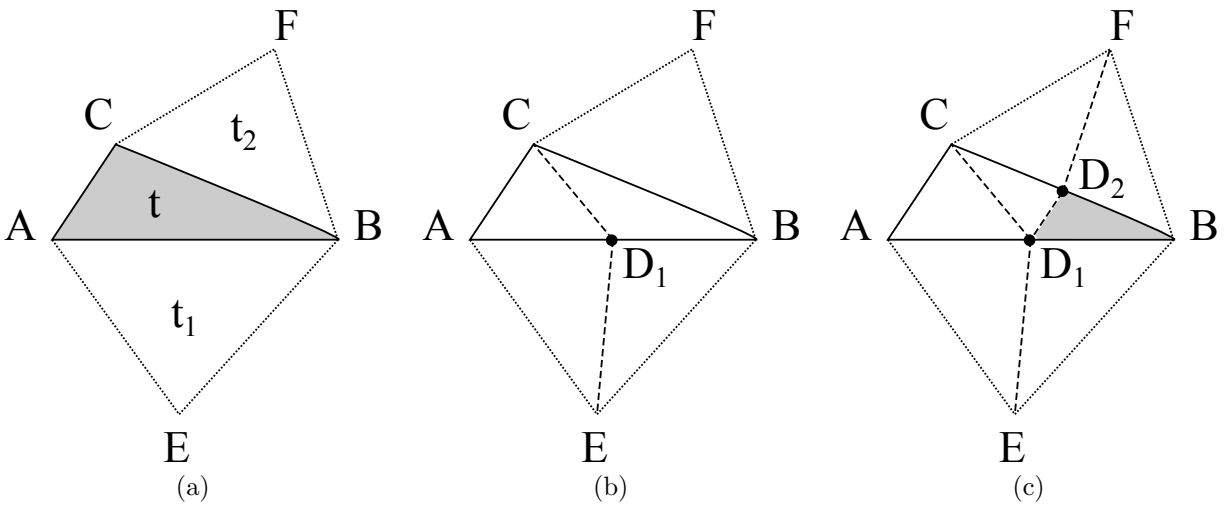


Figure 3.15: Example of an infinite processing cycle. (a) Triangles t and t_1 are terminal triangles. Triangle t_2 is the midsize-edge neighbor of t. (b) Insertion of point D_1. Edge CB becomes a terminal edge. (c) Insertion of point D_2 produces a triangle (shaded) similar to t.

the example in Figure 3.15, the strategy inserts the midpoint of midsize edge of t'. This operations produces new non-similar triangles, effectively avoiding the reproduction of the geometry; see Figure 3.16. The angle bounds and geometrical properties of the bisection of triangles are maintained.

3.3.6 Improvement Steps and Worsening of Triangles

Rivara [41] studied the appearance of intermediate slightly worse triangles during refinement process. This scenario arises after an edge-flip operation involving a point close to the circumcircle of a too-obtuse target triangle, causing the small angle of a neighbor triangle to be cut as shown in Figure 3.17. The same study proved that only a finite number of these triangles could appear, and that they will be consequently eliminated by the algorithm through edge-flip operations.

A similar behavior is observed in circumcenter-based Delaunay refinement algorithms such as Ruppert's algorithm [53]. This behavior arises when the circumcenter of a split triangle lies within the circumcircle of a skinny neighbor triangle, reducing the global minimum angle of the triangulation. As with the Lepp-Delaunay algorithm, the new slightly-worse triangles are eliminated after refinement.

3.3.7 Variations of the Lepp-Delaunay Algorithm

Rivara and Calderon [44] proposed a new point selection strategy for the Lepp-Delaunay algorithm in order to reduce the number of points inserted. In the *centroid strategy*, the algorithm selects the *centroid* of the pair of terminal triangles instead of the midpoint of the

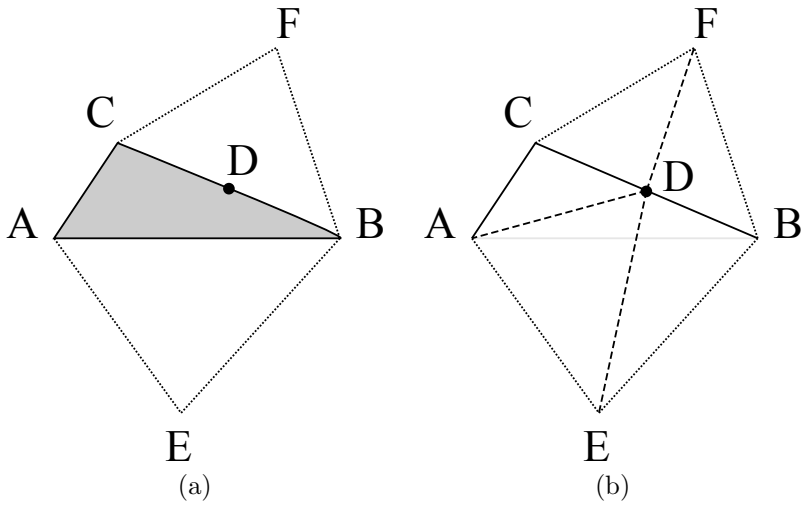


Figure 3.16: Example of the midsize-edge bisection of a triangle $t(ABC)$ that otherwise would produce an infinite processing cycle. (a) Insertion of point D, midpoint of midsize edge BC. (b) Final triangulation. No similar triangles to t are generated.

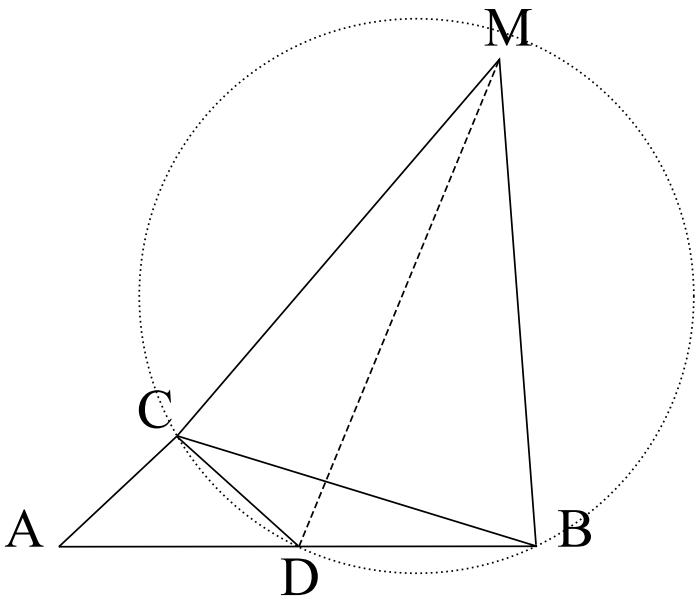


Figure 3.17: After the insertion of point D, neighbor triangle CBM has vertex M close to the circumcenter of triangle BCD. The algorithm could introduce bad quality triangles CDM and DBM if dotted edge DM is created.

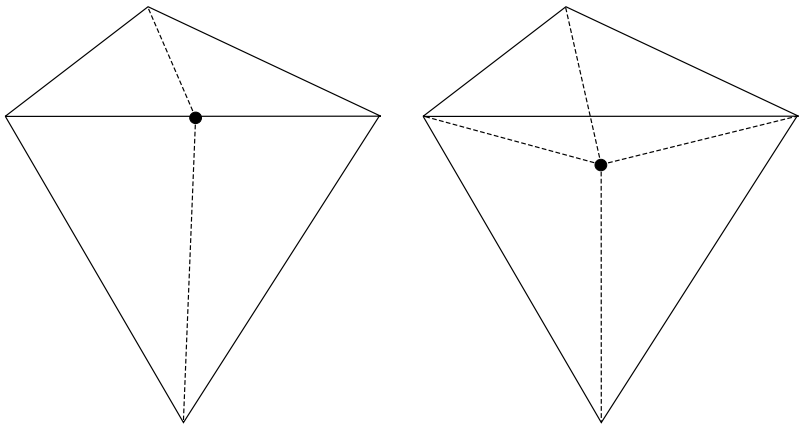


Figure 3.18: Comparison of the point selection insertion strategies for terminal triangles. (a) Midpoint strategy. (b) Centroid strategy.

terminal edge (Figure 3.18). Rivara and Calderon also included an empirical evaluation of their new strategy, showing that in practice the centroid strategy tends to insert less points than the midpoint strategy and than circumcenter-based Delaunay refinement algorithms.

Even though it is exhibited in practice, size-optimality was not theoretical guaranteed for the centroid strategy. Formal proofs of size-optimality and good grading for the Lepp-Delaunay's midpoint strategy could translate into proofs for the centroid strategy.

3.4 Conclusions

Longest-edge bisection algorithms, such as Lepp-Bisection algorithm [40, 42], perform local refinement through the bisection of triangles, maintaining the geometric quality of the mesh and increasing the area covered by nearly equilateral, good-quality triangles. A practical advantage of these algorithms is that they do not depend on complex operations such as circumcenter computations or the in-circle test. Additionally, the use of a longest-edge strategy allows a faster, local refinement of target triangles, producing well-graded triangulations.

The Longest-edge bisection refinement algorithms inherit the mathematical properties of the longest-edge bisection of triangles reviewed in Chapter 2. In addition to the non-degeneracy properties of the iterative longest-edge bisection of triangles, the algorithms offer guarantees on the minimum angle and on the number of non-similar triangles generated.

It is guaranteed that the Lepp-Bisection algorithm produces refined triangulations with angles greater than $\geq \theta/2$, where θ is the smallest angle in the initial triangulation. These algorithms provide a fast refinement process using robust operations.

The propagated refinement aims to improve the local grading of the triangulation, producing a smooth transition between target triangles and their neighbors. This is especially

observed when the bisection of a small target triangle propagates to neighboring, bigger triangles, effectively bisecting them in order to maintain and improve the grading.

The Lepp-Delaunay algorithm [40, 45] was proposed as an improvement of the original longest-edge bisection strategy. This algorithm maintains a Delaunay triangulation of the input performing the Delaunay insertion of new points. It improves the quality of the mesh and effectively removes bad quality triangles from the triangulation. The Lepp-Delaunay algorithm is supported by the properties of both the longest-edge bisection of triangles and the Delaunay triangulation.

The Delaunay terminal triangles have an important property: the largest angle is smaller than 120°. This property implicitly establishes bounds on the geometric position of the points inserted by the algorithm,and avoids the insertion of points too close to previous ones.

The algorithm is robust, and performs well in practice, producing well-graded triangulations. It can be easily implemented due to the simplicity of the point selection and insertion strategies. Additionally, the fact that the algorithm is independent of the processing order of triangles simplifies its implementation in parallel environments, yielding to efficient and scalable implementations [50, 51].

Chapter 4

The Lepp-Bisection Algorithm and Linear Time Refinement

In this chapter we study the computational cost of the Lepp-Bisection algorithm, providing a methodology for the analysis of any longest-edge bisection refinement algorithm. Our analysis is based on the number of points inserted by the algorithm during the refinement process. With this analysis we provide an upper bound on the size (number of elements) of the output triangulations. We thoroughly analyze the terminal-edge insertion strategy used by the algorithm. We also discuss the implications of both the refinement propagation and the iterative bisection of triangles performed to maintain a conforming triangulation.

We prove that both the length of the propagating paths and the number of points inserted per triangle become constant after few iterations, hence the cost of refinement becomes linear on the number of triangles selected for refinement. Our results guarantee that (1) the Lepp-Bisection algorithm generates refined triangulations of size within a constant factor of the initial triangulation, and (2) the algorithm refines a triangle in constant time when applied iteratively since it inserts a constant number of new points.

In Section 4.1 we describe the refinement problem for the Lepp-Bisection algorithm and the strategy for our analysis. We study the processing of non-conforming triangles from Section 4.3 through Section 4.5. In Section 4.3 we define a bound on the number of bisections that the Lepp-Bisection algorithm performs per triangle and the number of new points inserted. We also discuss the minimum and maximum number of points that the algorithm could produce inside any given triangle during iterative refinement.

In Section 4.7 we analyze the properties of the propagation throughout the refinement, bounding the length of the propagating paths and proving that they are monotonically reduced during the refinement of triangles. Finally, Section 4.8 presents the experimental results, showing that the algorithm performs very well in practice, in accordance with the theoretical results.

4.1 Preliminaries

We consider a conforming 2-dimensional triangulation, where pairs of neighbor triangles share either a common vertex or a common edge. We also consider that the minimum angle in the triangulation is greater than or equal to θ.

Given a triangulation τ and set $S_{\text{target}} \subseteq \tau$ of triangles selected for refinement, the *triangle-set refinement problem* consists of producing a conforming triangulation τ' such that every triangle $t \in S_{\text{target}}$ has been refined. This problem is commonly found in finite element applications, where triangle-set S_{target} corresponds to the triangles in the triangulation with unacceptable finite element error.

In this context, the goal of the refinement operation is to split the target triangles in order to produce a fine decomposition. An algorithm that provides guarantees on the quality and size of the refined triangles and on the size final triangulation is desirable.

The Lepp-Bisection algorithm introduces a number of N_{ref} new mandatory points required to refine the triangles in S_{target} (which we call *target refinement*), and a number of N_{prop} new propagated points required to maintain the mesh valid (which we call *propagated refinement*).

We analyze the computational cost of the algorithm considering the following two factors:

(1) $N = N_{ref} + N_{prop}$, the number of new points inserted into the triangulation, and

(2) the cost of the insertion operation.

Using an appropriate data structure to provide constant-time access to neighborhood information we can achieve constant-time point insertion operations. Then, the cost of the algorithm becomes linear in N, independent of the size of the triangulation and the number of iterations performed [38].

Since the target triangles in S_{target} only require one bisection to be refined, then N_{ref} is bounded by $\Theta(|S_{\text{target}}|)$. The non-trivial part of the study is then to define a bound on N_{prop} since we also have to take into consideration the propagated refinement. We establish a theoretical bound on N_{prop} through the study of the point selection strategy and the number of points inserted during propagated refinement.

4.1.1 Order-Independence and Equivalency

The longest-edge bisection strategy yields to refinement algorithms that can process triangles in any specific order. Supported by this property, we show that reformulations of the original longest-edge bisection algorithm produce the same refined triangulations for a given input.

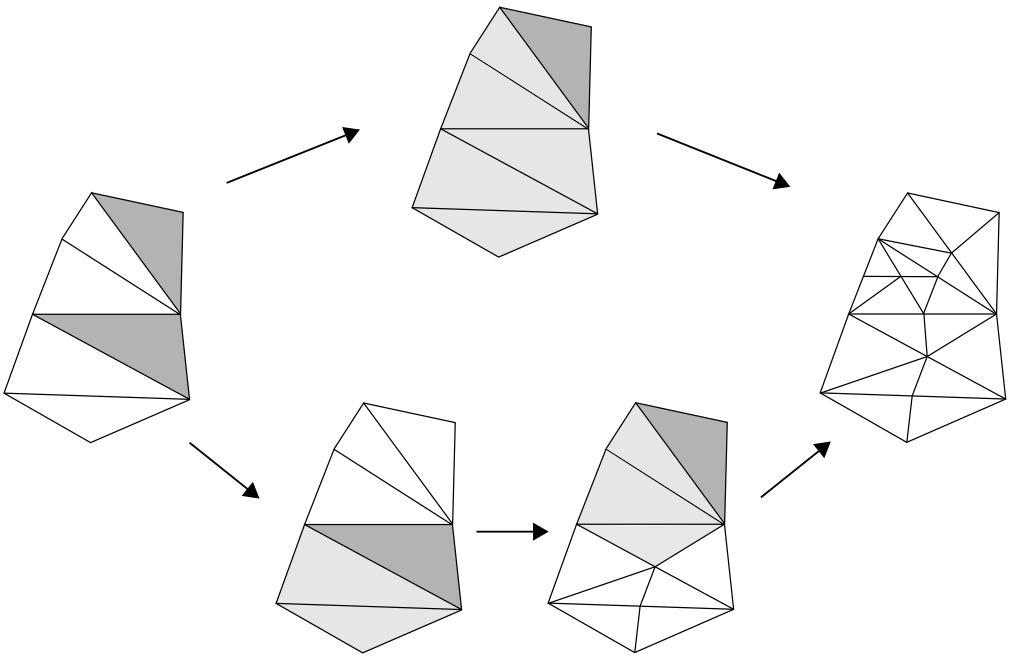


Figure 4.1: Processing target triangles in different order with the Lepp-Bisection algorithm. (left) Initial triangulation. Target triangles are shaded in dark gray . (middle) The two possible processing orders of target triangles. Lepps are shaded in light gray. (right) Output triangulation.

First, we must ensure that the refinement algorithm consistently selects the longest edge of a triangle for non-unique longest edges (i.e. isosceles and equilateral triangles). Suppose a triangulation τ with m edges $e_1, \ldots, e_m$, where the length of e_i is $\updownarrow_i$. A total ordering of these edges, $e_1 < e_2 < e_3 < \ldots < e_m$, is *admissible* if it satisfies $\updownarrow_1 \leq \updownarrow_2 \leq \updownarrow_3 \leq \ldots \leq \updownarrow_m$. We say that an algorithm is *consistent* if there exists an admissible ordering of the edges of τ such that the longest edge a triangle with edges $\{e_i, e_j, e_k\}$, where $i < j < k$, is e_k.

Given that the longest edge of a triangle is unique, the longest-edge bisection of a triangle produces a unique pair of children triangles. As a result, bisecting the triangles in a different order does not alter the resulting refined triangulation. Furthermore, the uniqueness of the longest edge also defines a unique longest-edge propagating path.

Fact 4.1. *Working with a consistent longest-edge bisection algorithm, the order in which triangles are bisected does not alter the resulting triangulation.*

Fact 4.2. *The order in which a consistent longest-edge bisection algorithm processes the set of target triangles, S_{target}, does not alter the output triangulation.*

Then, we say that an algorithm is *order independent* if the triangulation produced by the algorithm is the same regardless of the order in which target triangles in S_{target} are processed. Figure 4.1 illustrates two processing orders of target triangles using the Lepp-Bisection algorithm. The output triangulations are exactly the same.

In the original longest-edge bisection algorithm (Algorithm 3.1), refinement continues as long as S_{proc} contains triangles that need to be bisected. The Lepp-Bisection algorithm

(Algorithm 3.2) delays the bisection of triangles that become non-conforming in order to maintain a conforming triangulation throughout the whole process. These triangles are stored in order following the sequence of longest-edge neighbors, or *longest-edge propagating path*, until finding the pair of triangles sharing a common longest edge, hence called *terminal triangles*. The process is repeated, backtracking over the sequence until the triangle that originated the propagation is refined.

We introduce the idea of equivalent algorithms to define this relationship.

Definition 4.3. Let M and N be two longest-edge bisection algorithms. We say that M and N are *equivalent*, $\mathcal{M} \equiv \mathcal{N}$, if for an input triangulation τ the output triangulations $\tau'_{\mathcal{M}}$ and $\tau'_{\mathcal{N}}$ produced by M and N respectively are the same.

Note that the difference between the original longest-edge bisection algorithm and the Lepp-Bisection algorithm is the order in which the triangles in set S_{proc} are processed. Then, due to the Facts 4.2 and 4.1, both algorithms produce the same output triangulations and are equivalent. (For an example see Figures 3.1 and 3.4)

Lemma 4.4. *The Lepp-Bisection algorithm and the original longest-edge bisection algorithm are equivalent.*

4.1.2 Reformulation of the Longest-Edge Bisection Algorithm

Our analysis of the cost of refinement per triangle is based on how a non-conforming triangle is processed. We present a reformulation of the longest-edge bisection algorithm to facilitate this analysis. We describe an order for processing triangles in S_{proc} such that a non-conforming triangle is iteratively bisected until obtaining a conforming triangulation in its interior.

As discussed Section 4.1.1, the order in which triangles from S_{proc} are processed does not affect the output of the algorithm. For theoretical purposes, we provide a reformulation of the longest-edge bisection algorithm to illustrate how the longest-edge strategy processes individual triangles. Compared to Algorithm 3.1 the main difference is the order in which the algorithm processes the non-conforming triangles. In this reformulation, all the non-conforming descendants of a triangle are processed before moving on to non-conforming neighbors.

We can simulate this priority of triangles using a stack data structure (as in Algorithm 3.3), and is equivalent to performing iterative bisections of the selected triangle until a conforming triangulation is produced in its interior. Algorithm 4.1 describes this reformulation.

Note that the algorithm described in Algorithm 4.1 is equivalent to both the original longest-edge bisection algorithm (Algorithm 3.1) and the Lepp-Bisection algorithm (Algorithm 3.2), since changing the order in which tringles are processed does not affect the final triangulation.

Algorithm 4.1 Longest-edge bisection algorithm (iterative bisection)

Input: A quality triangulation τ and a set S_{target} of triangles to be refined
Output: A quality triangulation τ' such that each triangle $t \in S_{\text{target}}$ has been refined

```
1: for each t in S_target do
2:    Initialize a stack Q with t
3:    while Q is not empty do
4:       u is the triangle at the top of Q
5:       Triangle v is the longest-edge neighbor of u
6:       Perform the longest-edge bisection of u, producing children triangles u_1 and u_2
7:       Remove u from Q {also remove u from S_target if it was selected for refinement}
8:       if v is a non-conforming triangle then
9:          Add v to Q
10:      end if
11:      Add to Q any non-conforming children of u
12:   end while
13: end for
```

4.2 Geometrical Characterization of the Similarity Regions

When processing the non-conforming triangle, the number of bisections required to obtain a conforming triangulation is affected by the similarity class of the triangle (reviewed in Section 2.2). We propose an extended classification based on the classification proposed by Gutierrez *et al.* [16], defining seven similarity regions. Our classification is based on the patterns of iterative bisection performed by the longest-edge bisection algorithm. Figure 4.2 illustrates this classification, which covers every possible scenario.

Now let $t(ABC)$ be triangle with CA, BC and AB corresponding to the shortest, midsize and longest edge ($CA \leq BC \leq AB$); and angles α, β and γ corresponding to the smallest, midsize and largest angle ($\alpha \leq \beta \leq \gamma$) as shown in Figure 4.3.

We define a geometrical characterization of each similarity region by defining the range of possible angle values that a triangle from a given region could adopt. In Table 4.1 we show the minimum and maximum angle values for the smallest, midsize and largest angle. In the last column we show the characterization based on the relationship among edges after a bisection, which represent the defining properties of each region. Not shown in the table is the property to exactly define Regions V and VI: for Region V triangles it holds that $CD \leq CB/2$, while for Region VI triangles it holds that $CD \geq CB/2$.

We also define the same geometrical characterization of the first descendants of a triangle from each region. Following the representation of triangle t in Figure 4.3(a) we say that the *left child* of t corresponds to the sub-triangle containing point A (associated with midsize angle β), while the *right child* of t corresponds to the sub-triangle containing point B (associated with smallest angle α) as shown in Figure 4.3(b). Tables 4.2 and 4.2 show, respectively, the geometry of the left child and the right child of a triangle.

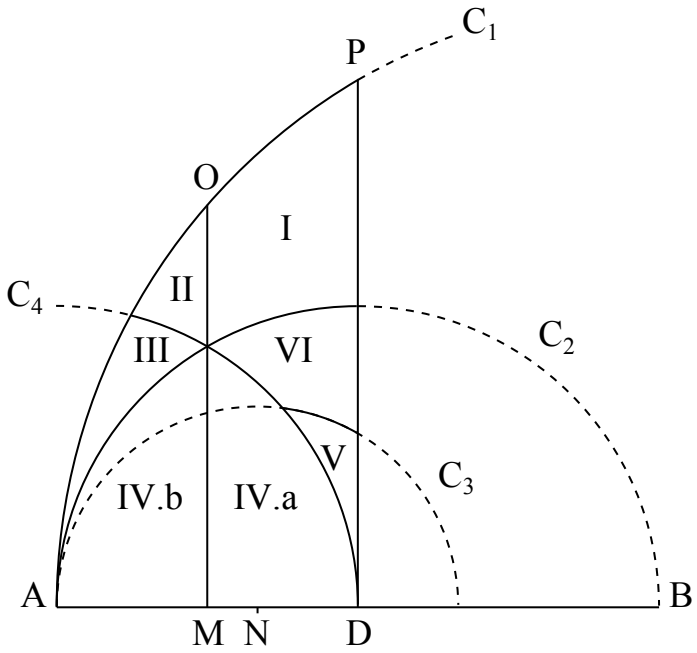


Figure 4.2: Extended geometric classification of triangles $t(ABC)$ for the analysis of the longest-edge bisection algorithm.

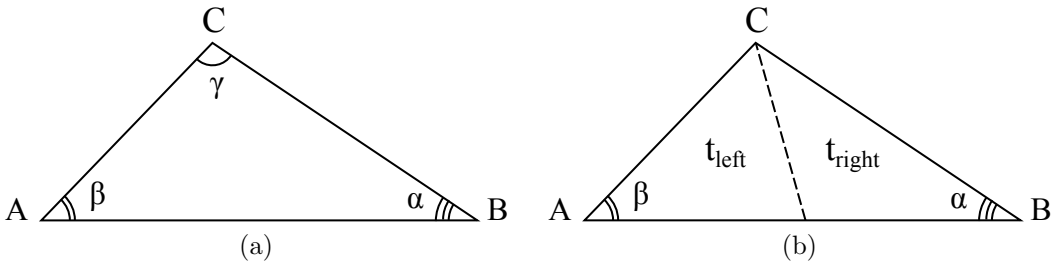


Figure 4.3: (a) Triangle $t(ABC)$. (b) Sub-triangles obtained after the bisection of t.

Table 4.1: Geometrical characterization of the triangles of each similarity region. (angles given in degrees)

Region	α (min)	α (max)	β (min)	β (max)	γ (min)	γ (max)	Edges
I	30	60	45	69.30	60	90	$AD \leq CD \leq CA$
II	28.96	41.41	60	75.52	69.30	90	$AD \leq CA \leq CD$
III	> 0	30	60	90	75.52	90	$CA \leq AD \leq CD$
IV.a	> 0	30	> 0	60	90	< 180	$CD \leq CA \leq AD$
IV.b	> 0	30	> 0	60	90	< 180	$CA \leq CD \leq AD$
V	> 0	30	> 0	41.41	110.7	< 180	$CD \leq AD \leq CA$
VI	27.89	45	30	60	90	120	$CD \leq AD \leq CA$

Table 4.2: Geometrical characterization of the left child per region. (angles given in degrees)

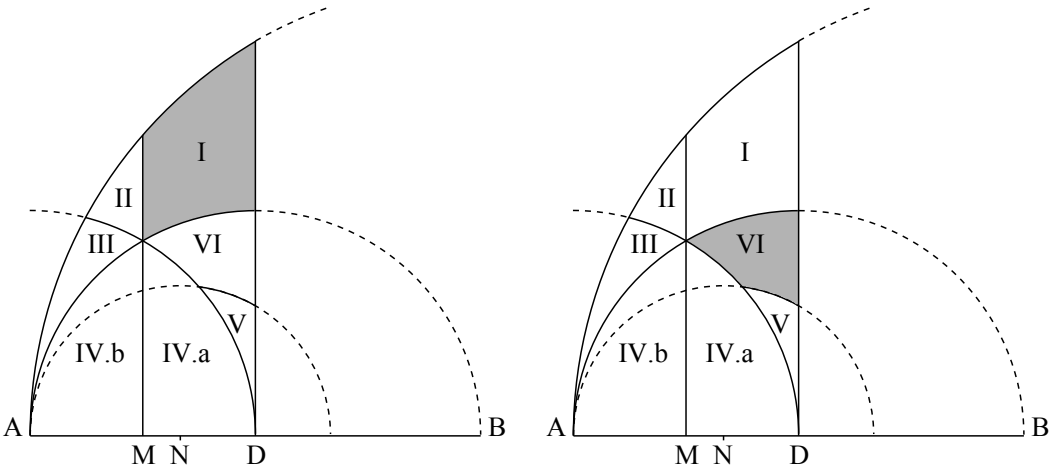
	Left Child						
Region	α (min)	α (max)	β (min)	β (max)	γ (min)	γ (max)	Child's Region
I	30	60	45	69.30	60	90	I
II	41.41	60	52.24	69.30	60	75.52	I
III	> 0	60	52.24	90	60	90	I, II, III
IV.a	> 0	60	> 0	90	60	< 180	any
IV.b	> 0	60	> 0	90	60	< 180	any
V	> 0	41.41	60	90	69.30	90	II, III
VI	30	60	45	69.30	60	90	I

Table 4.3: Geometrical characterization of the right child per region. (angles given in degrees)

	Right Child						
Region	α (min)	α (max)	β (min)	β (max)	γ (min)	γ (max)	Child's Region
I	27.89	45	30	45	90	120	VI
II	23.28	30	28.96	41.41	110.70	127.76	V
III	> 0	30	> 0	30	120	< 180	V
IV.a	> 0	30	> 0	90	90	< 180	IV.a, IV.b, V
IV.b	> 0	30	> 0	30	120	< 180	IV.a, V
V	> 0	30	41.41	90	90	110.70	IV.a, IV.b
VI	27.89	45	30	60	90	120	VI

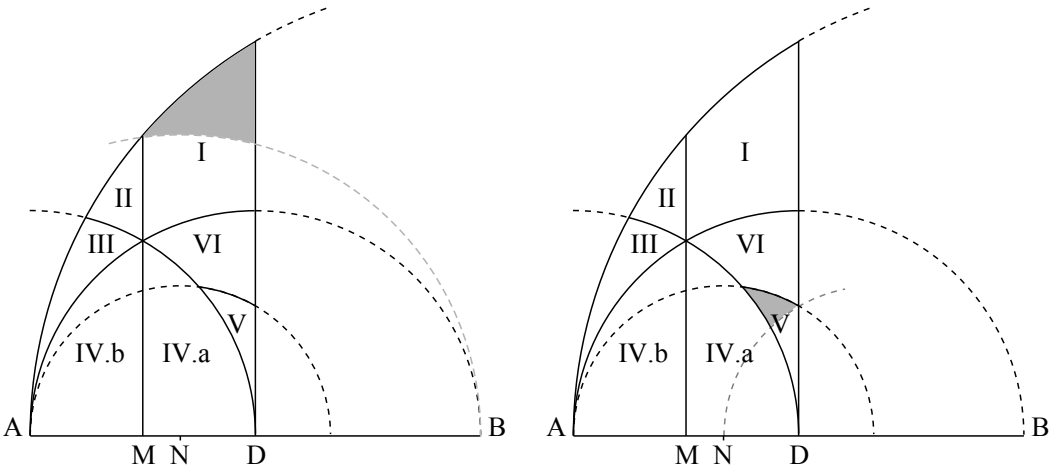
After the geometrical characterization of every region and its children triangles, we further extend the diagram shown in Figure 4.2 to identify the corresponding regions for the left and right children triangles. Figures 4.4, 4.5 and 4.6 show the areas where virtual vertex C lies for the children of Region I, II and III triangles respectively. Figures 4.7, 4.8, 4.9 and 4.10 show the areas where virtual vertex C lies for the children of Region IV.a, IV.b, V and VI triangles respectively.

As observed in Figures 4.4 and 4.10, the areas for children of Region I and VI triangles (i.e. quasi-equilateral triangles) overlap completely. Moreover, the left child of a quasi-equilateral triangle corresponds to a Region I triangle, and the right child corresponds to a Region VI triangle. Then, the following fact is true for any quasi-equilateral triangle.



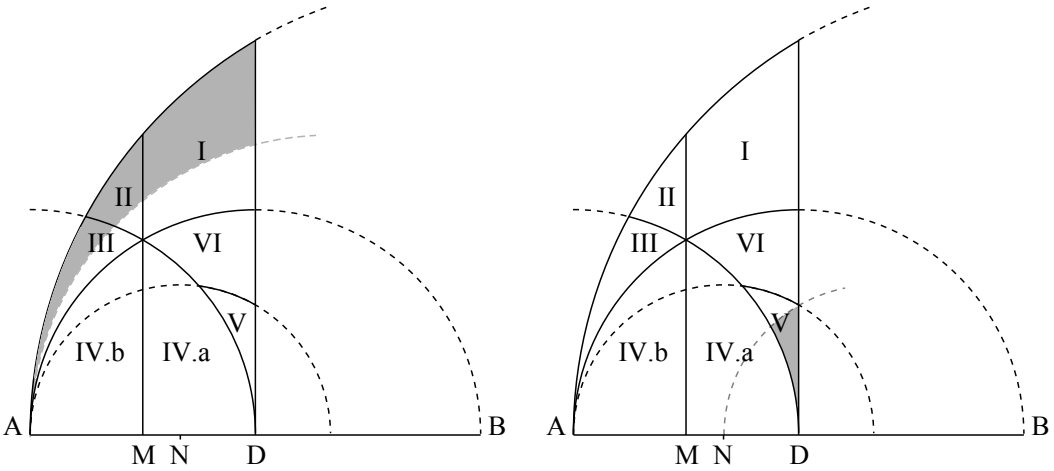
(a) Region I: left child (b) Region I: right child

Figure 4.4: Shaded areas represent the regions for each child of a Region I triangle.



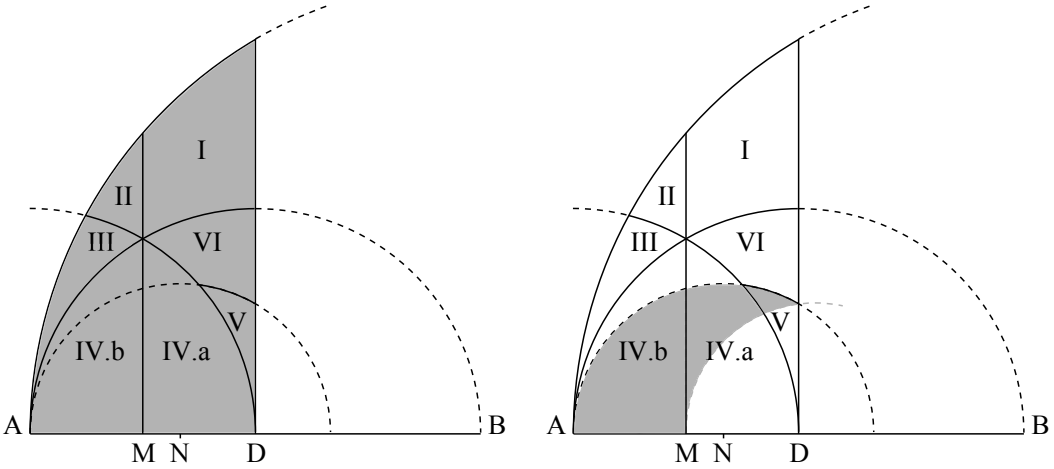
(a) Region II: left child (b) Region II: right child

Figure 4.5: Shaded areas represent the regions for each child of a Region II triangle.



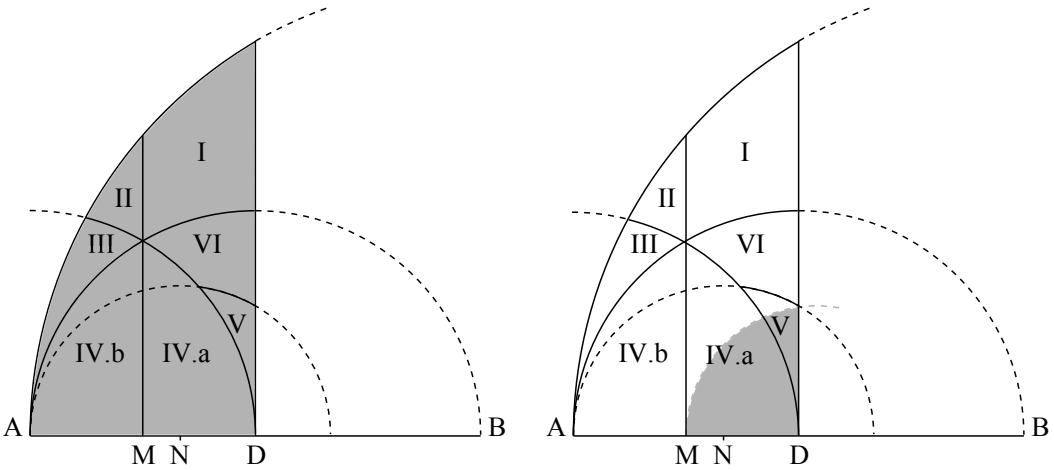
(a) Region III: left child (b) Region III: right child

Figure 4.6: Shaded areas represent the regions for each child of a Region III triangle.



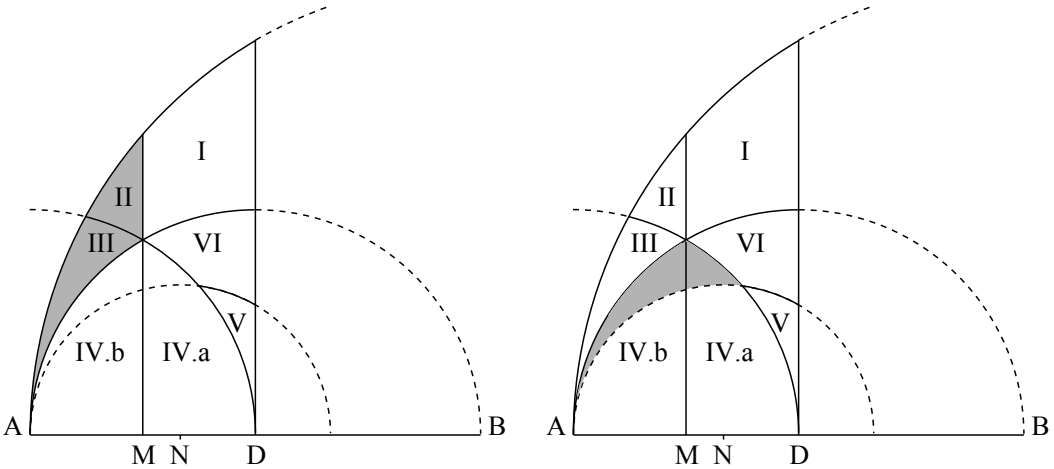
(a) Region IV.a: left child (b) Region IV.a: right child

Figure 4.7: Shaded areas represent the regions for each child of a Region IV.a triangle.



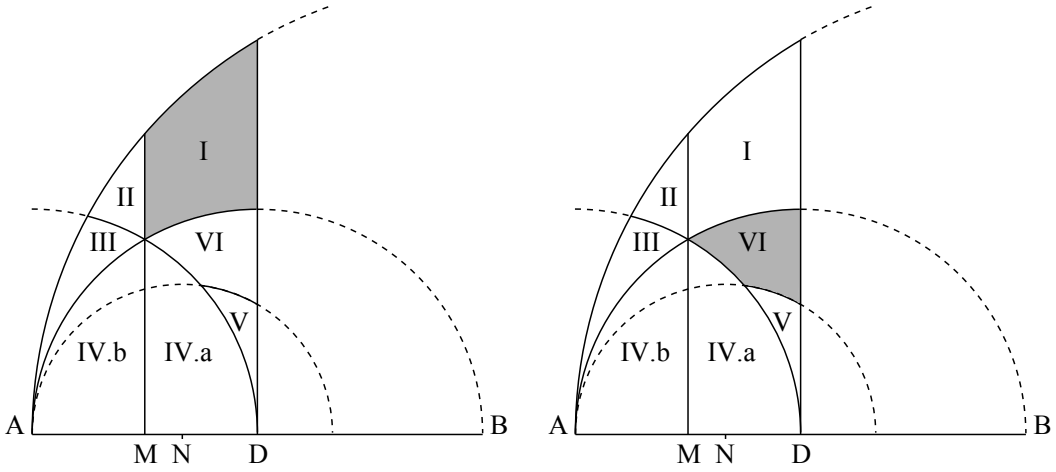
(a) Region IV.b: left child (b) Region IV.b: right child

Figure 4.8: Shaded areas represent the regions for each child of a Region IV.b triangle.



(a) Region V: left child (b) Region V: right child

Figure 4.9: Shaded areas represent the regions for each child of a Region V triangle.



(a) Region VI: left child (b) Region VI: right child

Figure 4.10: Shaded areas represent the regions for each child of a Region VI triangle.

Fact 4.5. *The children of a quasi-equilateral triangle are also quasi-equilateral. Furthermore, the left child is acute while the right child is obtuse.*

We summarize the results on the children triangles' regions with the directed graph shown in Figure 4.11, representing the transition from one region to another after a longest-edge bisection (e.g. a Region II triangle will produce Region I and V triangles). Figure 4.11 also extends the initial results of Gutierrez *et al.* presented in Figure 2.6, illustrating with more detail the evolution of regions during iterative refinement and the convergence to quasi-equilateral triangles (self-loops are explicitly shown).

4.3 Processing Non-Conforming Points

In this section we analyze the local treatment of non-conforming points. We study the behavior of the longest-edge bisection algorithm described in Algorithm 4.1 during the processing of a non-conforming triangle.

Consider a triangle $t \in S_{\text{target}}$ and its longest-edge neighbor t'. Following Algorithm 4.1, the longest-edge bisection of t might cause t' to become non-conforming. Consider point Q the non-conforming point, midpoint of the longest edge of t, inserted after the bisection of t. In turn, point Q will correspond to the midpoint of either the shortest, midsize or longest edge of t'.

As shown in Figure 4.12, during the isolated processing of triangle $t'(ABC)$ the goal is to produce the smallest conforming triangulation inside ABC such that point Q (and the affected edge of t') becomes conforming.

We define the following three possible scenarios for the treatment of the non-conforming midpoints of the longest, midsize or shortest edge of a triangle:

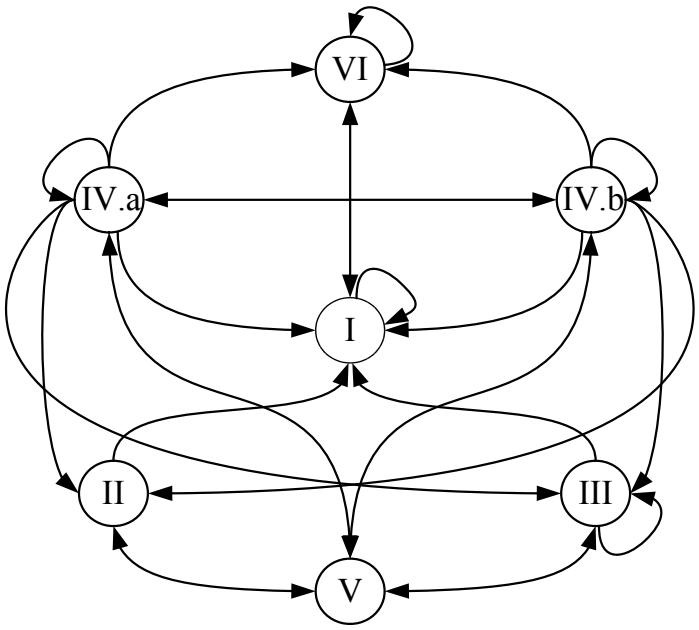


Figure 4.11: Directed graph showing the possible regions of the children triangles after performing are longest-edge bisection.

Type 1: Q is the midpoint of the longest edge of t' (Figure 4.12(a)). This case corresponds to refinement of two terminal triangles since t and t' share the same longest edge. The longest-edge bisection of t' is enough to obtain a conforming triangulation. No new points are inserted.

Type 2: Q is the midpoint of the midsize edge of t' (Figure 4.12(b)). In this case two bisections are required to obtain a valid mesh. The first bisection corresponds to the longest-edge bisection of t', inserting point Q_1 and splitting t' in two triangles. Consider t'' the non-conforming triangle descendant of t' containing non-conforming point Q. Since Q is now the midpoint of the longest edge of t'' (Type 1), the longest-edge bisection of t'' produces a conforming triangulation.

Type 3: Q is the midpoint of the shortest edge of t' (Figure 4.12(c)). After the bisection of t', its shortest edge could become the longest edge of its non-conforming descendant (Type 2), otherwise several bisections will be performed in order to eliminate the non-conforming triangle. Some points will be added over edge AB and in some scenarios over edge BC. The number of iterative bisections will depend on both the geometric quality (minimum angle) and similarity class (Section 2.2) of t'.

It must be noted that the processing of non-conforming points of Type 1 and Type 2 is not affected the triangle's quality and of its similarity class, while in the case of Type 3 several bisections could be required as illustrated in Figure 4.12(c).

Based on the above discussion for the treatment of Type 1 and Type 2 non-conforming points, the following lemma holds for the longest-edge bisection algorithm.

Lemma 4.6. *When a non-conforming triangle t has non-conforming point Q located at the*

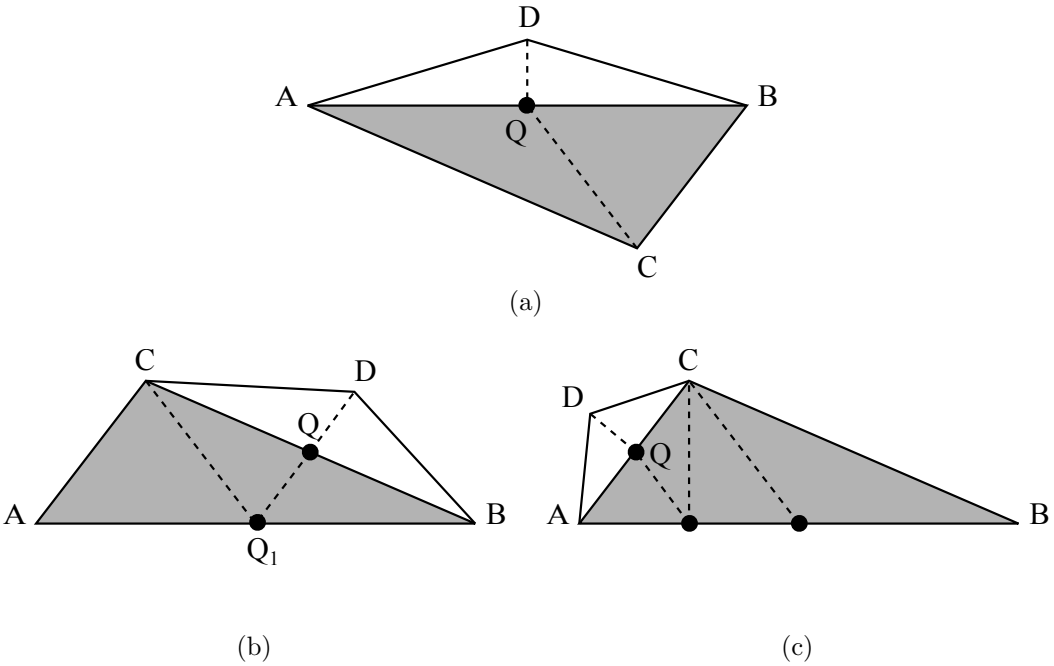


Figure 4.12: Possible scenarios for the location of non-conforming point Q along the edges of triangle t' (shaded). (a) Type 1: Q is the midpoint of the longest edge. (b) Type 2: Q is the midpoint of the midsize edge. (c) Type 3: Example of a bisection pattern when Q is the midpoint of the shortest edge.

midpoint of either its longest or midsize edge, at most two bisections are performed (and at most one new point is created) in order to produce a conforming triangulation inside t.

Opposite to Lemma 4.6, when a the non-conforming point Q lies on its shortest edge (Type 3), then at least two bisections are needed to produce a conforming triangulation inside t (inserting at least one new point). Furthermore, the number of bisections will be determined by the similarity class of t and its geometric quality as we discuss in Section 4.4.

4.4 Processing the Non-Conforming Midpoint of the Shortest Edge of a Triangle

As discussed in Section 4.3, when a triangle is bisected its longest-edge neighbor could become a non-conforming triangle. Given the non-conforming triangle, at most two iterative bisections are required to obtain a conforming triangulation if the non-conforming point corresponds to the midpoint of its longest or midsize edges (Lemma 4.6).

In this section we study the behavior of the longest-edge bisection algorithm during the processing of the non-conforming midpoint of the shortest edge of a triangle t. Based on the geometric quality of each similarity class discussed in Section 4.2, we define the iterative bisection patterns for each similarity class of non-conforming triangle t. We also present

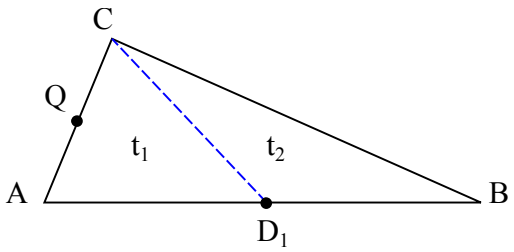


Figure 4.13: Non-conforming triangle $t(ABC)$ is produced after the insertion of point Q (midpoint of shortest edge CA). Sub-triangles $t_1(AD_1C)$ and $t_2(CD_1B)$ are obtained after the initial longest-edge bisection of t.

new bounds on the number of bisection that the algorithm performs when processing these non-conforming triangles.

It must be noted that a bound on the number of bisections is equivalent to a bound on the number of triangles and points in the triangulation since the number of new triangles produced is equal to the number of bisections performed.

Let $t(ABC)$ be a non-conforming triangle with smallest angle α and edges $AB \geq BC \geq CA$. Consider non-conforming point Q, the midpoint of the triangles shortest edge CA (see Figure 4.13). Point D_1 is inserted after the mandatory longest-edge bisection of t, producing children sub-triangles $t_1(AD_1C)$ and $t_2(CD_1B)$.

Given the non-conforming triangle t of Figure 4.13 and sub-triangle t_1 (left children of t), the non-conforming triangle containing point Q, then the number of bisections that the longest-edge bisection algorithm performs in order to produce a conforming triangulation inside t will be determined by the triangle's similarity class as follows:

1. **Region I ($AD_1 \leq CD_1 \leq CA$):** Edge CA becomes the longest edge of triangle t_1. The longest-edge bisection of t_1 is enough to produce a conforming triangulation. Triangle t_2 remains unaffected and only one new point is inserted. (See Figure 4.14(a)).

2. **Region II ($AD_1 \leq CA \leq CD_1$):** Triangle t_1 belongs to Region I and point Q is the midpoint of to its midsize edge (Type 2). Two bisections are performed: the first one inserts point D_2 over edge CD_1 and the second one connects points D_2 and Q. After inserting point D_2, neighbor triangle t_2 becomes non-conforming, with non-conforming point D_2 being the midpoint of its midsize edge CD_1 (Type 2). Two additional bisections are performed on t_2 in order to produce a conforming triangulation. Three new points are inserted during this process. (See Figure 4.14(b))

3. **Region III ($CA \leq AD_1 \leq CD_1$):** Triangle t_1 will either belong to Region I, II or III.

   - If t_1 belongs to Region I or II, the number of bisections and points produced inside t_1 correspond to the scenarios for Region described above.
   - If t_1 still belongs to Region III, after no more than $k_1 = \lceil \log(\frac{\pi}{6\alpha}) / \log(\frac{3}{2}) \rceil$ recursive bisections of the non-conforming descendant of t_1 containing point Q, called a *step*,

the algorithm will face a triangle from Region I or II. [1]

The bound on k_1 is obtained from the following observations: (i) for the non-conforming triangle produced on the $i-$step, say $t_{1,i}$, it holds that $\alpha_{t_{1,i}} \geq \frac{3}{2}\alpha_{t_{1,i-1}}$ (Theorem 2.7 part *c*), and (ii) triangles with smallest angle $\alpha_{t_{1,i}} < 30°$ could still belong to Region III.

In either scenario at least one point will be inserted over edge CD_1, making sub-triangle t_2 non-conforming. For $k_1 = 0$, the non-conforming point corresponds to the midpoint of the midsize edge of t_2 (Type 2), so two bisections will be enough to produce a conforming triangulation. Figure 4.14(c) shows a Region III triangle when t_1 is a Region I triangle ($k_1 = 0$).

For $k_1 > 0$, $O(k_1^2)$ additional bisections are performed on some of the descendants of t_1 due to internal propagation. In the worst case, $O(k_1^2)$ new points will be inserted to produce a conforming triangulation (see Figure 4.17(b) for an example of this behavior).

4. **Region IV.a ($CD_1 \leq CA \leq AD_1$):** Edge CA becomes the midsize edge of triangle t_1 so point Q corresponds to a Type 2 non-conforming point. Two bisections are performed on t_1 to produce a conforming triangulation. Triangle t_2 remains unaffected. Two new points are produced. (See Figure 4.14(d))

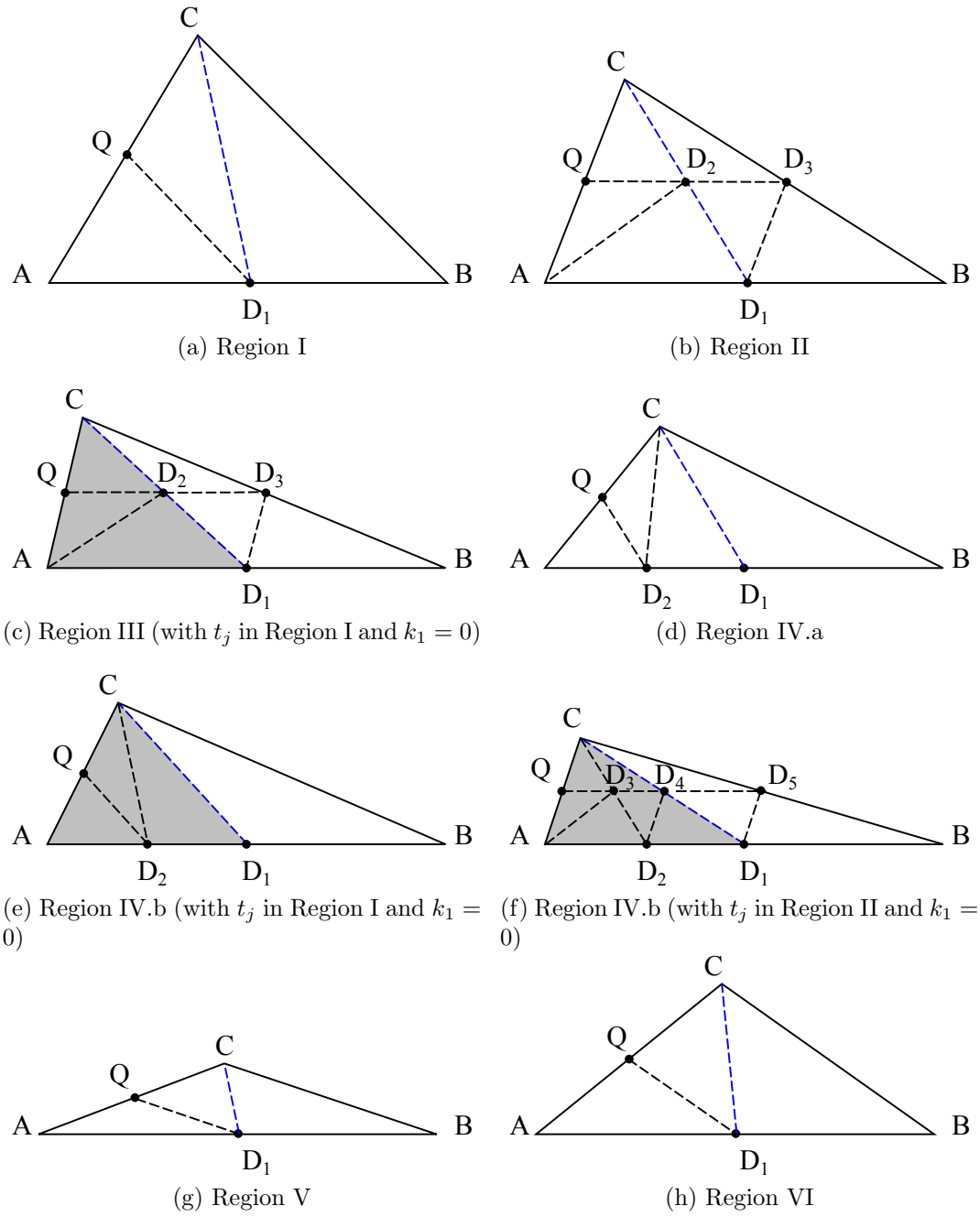
5. **Region IV.b ($CA \leq CD_1 \leq AD_1$):** Triangle t_1 could belong to any region. If t_1 is a Region IV.b triangle, then after no more than $k_2 = \lceil \log(\frac{\pi}{6\alpha}) \rceil$ steps the algorithm will face a triangle from any other region than IV.b.

   k_2 is obtained from the following observations: (i) for the non-conforming triangle produced on the $i-$step, say $t_{1,i}$, it holds that $\alpha_{t_{1,i}} = 2\alpha_{t_{1,i-1}}$ (Theorem 2.7 part *d*), and (ii) triangles with smallest angle $\alpha_{t_{1,i}} < 30°$ could belong to Region IV.b.

   The number of bisections will depend on the region of the first non-conforming descendant not belonging to Region IV.b as follows:

   - If on the $j-$th step ($j < k$) the non-conforming triangle $t_{1,j}$ corresponds to a Region I, VI.a, V or VI triangle, then the total number of bisections and new points inserted is j plus those required by $t_{1,j}$ according to its region (three in the worst case). Figure 4.14(e) shows a Region IV.b triangle when t_1 is a Region I triangle ($k_2 = 0$).
   - Otherwise, if on the j-th step $t_{1,j}$ corresponds to a Region II or III triangle, then every triangle descendant of t_1 will be iteratively bisected $O(k_2^2)$ in order to maintain a conforming triangulation. Then the total number of bisections and new points inserted is $O(k_2^2)$, which includes the processing of $t_{1,j}$ ($O(k_1^2)$ in the worst case when it belongs to Region III). Figure 4.14(f) shows a Region IV.b triangle when t_1 is a Region II triangle ($k_2 = 0$).

6. **Region V and VI ($CD_1 \leq AD_1 \leq CA$):** The behavior for these triangles is the same as the case for Region I triangles since the affected edge CA becomes the longest edge of t_1. Only one bisection is enough to produce a conforming triangulation. No new points are inserted. (See Figures 4.14(g) and 4.14(h))



(a) Region I

(b) Region II

(c) Region III (with t_j in Region I and $k_1 = 0$)

(d) Region IV.a

(e) Region IV.b (with t_j in Region I and $k_1 = 0$)

(f) Region IV.b (with t_j in Region II and $k_1 = 0$)

(g) Region V

(h) Region VI

Figure 4.14: Bisection required for a triangle $t(ABC)$ to produce a conforming triangulation when non-conforming point Q corresponds to the midpoint of the shortest edge. Shaded triangles could be represented with a triangle from another region.

The following lemma summarizes the cost associated with the processing of the non-conforming midpoint of the shortest edge of a triangle.

Lemma 4.7. *When a non-conforming triangle t with smallest angle α has non-conforming point Q located at the midpoint of its shortest edge, the number of new points and triangles inserted by the longest-edge bisection algorithm in order to produce a conforming triangulation inside t is:*

(1) For the triangles of Region I, V and VI, 3 triangles and 1 new point are produced.

(2) For the triangles of Region II, 6 triangles and 3 new points are produced.

(3) For the triangles of Region III and IV.b, at most $O(\kappa^2)$ triangles and points are produced, where $\kappa = \lceil \log(\frac{\pi}{6\alpha}) / \log(\frac{3}{2}) \rceil$.

(4) For the triangles of Region IV.a, 4 triangles and 2 new points are produced.

4.5 Processing Multiple Non-Conforming Points

So far we have analyzed the treatment of the non-conforming midpoints of a triangle's edge, however, in some scenarios multiple non-conforming points could lie along an edge. The goal is to process the non-conforming points so the algorithm produces the minimum conforming triangulation inside the triangle. In this section we study this scenario, and how non-conforming edges with multiple non-conforming points are processed.

The algorithm could produce non-conforming edges with multiple non-conforming points after processing the non-conforming midpoint of the shortest edge of a Region III, IV.a or IV.b triangle. In order to produce a conforming triangulation inside these triangles the algorithm could insert several points along the longest edge (and in some cases also along the midsize edge) as discussed in Section 4.4.

Consider the non-conforming triangle $t'(ABC)$ with smallest angle α'_t, and its longest-edge neighbor triangle $t(AEB)$ with smallest angle α_t. Consider that after processing triangle t' a sequence of k_3 points $\{D_1, D_2, \ldots, D_{k_3}\}$, with $k_3 \leq \lceil \log(\frac{\pi}{6\alpha'_t}) \rceil$, were inserted along the longest edge of t'. Figure 4.15(a) shows the case for $k_3 = 2$.

Due to the nature of the bisection of triangles, points in set $\{D_1, D_2, \ldots, D_{k_3}\}$ are inserted following a geometric progression: point D_1 is the midpoint of the longest-edge of t' ($AD_1 = AC/2$), point D_2 is the midpoint of edge of AD_1 ($AD_2 = AD_1/2 = AC/4$), point D_3 is the midpoint of edge AD_2, and so on.

Since triangle t became non-conforming, the algorithm processes the non-conforming points affecting t in the order they were initially produced. Each D_i non-conforming point,

[1] Our logarithms are base 2.

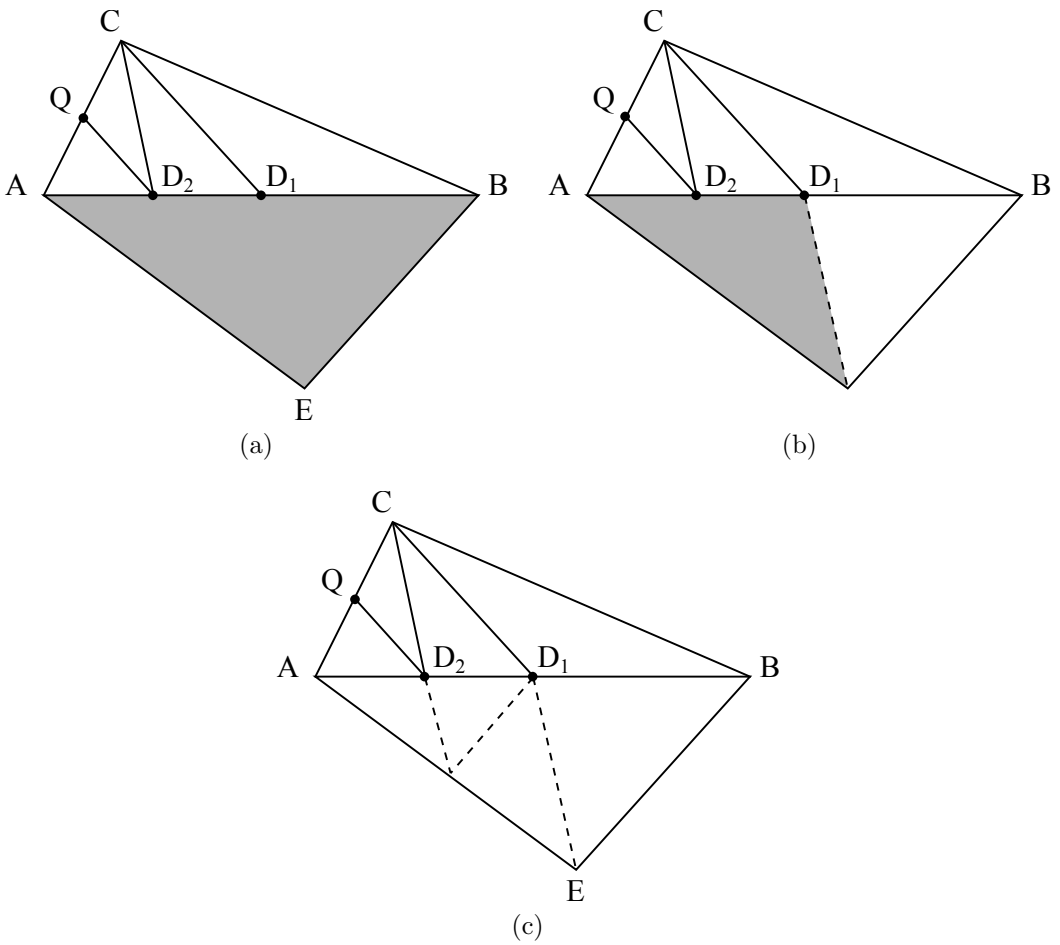


Figure 4.15: Processing multiple non-conforming points along and edge. (non-conforming triangles are shaded). (a) Initial triangulation with non-conforming triangle $t(AEB)$. (b) Bisections for point D_1. (c) Bisections for point D_2 and final triangulation.

$i = 1, \ldots, k_3$, will be processed according to the analysis presented in Section 4.1.1 and Section 4.4. Then, the algorithm will insert at most $O(\kappa_{\alpha_t}^2)$ new points in the worst case to process each non-conforming point, where κ_{α_t} depends on angle α_t (Lemma 4.7).

Figure 4.15 illustrates the processing of non-conforming triangle $t(AEB)$ with set of non-conforming points $\{D_1, D_2\}$ located along the longest edge. Point D_1, of Type 1, corresponds to the midpoint of the longest edge of triangle t so one bisection is performed to process this point (Figure 4.15b). Point D_2, of Type 2, corresponds to the midpoint of the midsize edge of non-conforming sub-triangle AED_1; two bisections are performed to process this point (Figure 4.15(c)). The resulting conforming triangulation is shown in Figure 4.15(c).

Figure 4.16 illustrates the processing of Region IV.b triangle $t(ABC)$ with $\alpha_t = 16^\circ$ and set of non-conforming points $\{D_1, D_2\}$ along the shortest edge of t. Figure 4.16(a) shows the initial triangle. After one bisection of t, the sub-triangle containing the non-conforming points belongs to Region II (Figure 4.16(b)). Point D_1, of Type 3, corresponds to the midpoint of the shortest edge, and is processed after seven bisections (Figure 4.16(c)). Point D_2, of Type 2, is processed after two bisections, but further bisections are propagated to

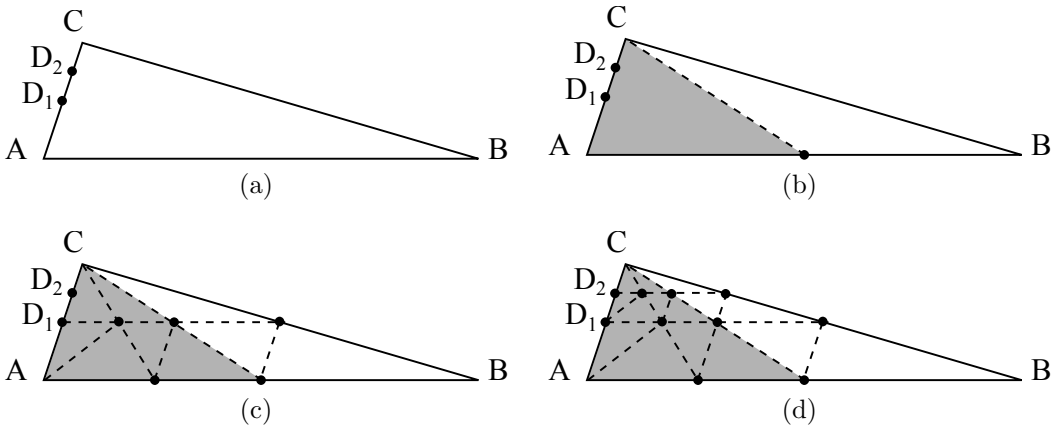


Figure 4.16: Processing multiple non-conforming points along the shortest edge of a Region IV.b triangle. (a) Initial triangulation, non-conforming triangle $t(ABC)$. (b) Shaded triangle belongs to Region II. (c) Bisections for point D_1. (d) Bisections for point D_2 and final triangulation.

other sub-triangles to produce a conforming triangulation (Figure 4.16(d)).

When the non-conforming edge corresponds to the longest or midsize edge of triangle t, the algorithm will process Type 1 or Type 2 non-conforming points, not behaving worse than the scenario described above. Furthermore, working with triangles of acceptable quality (e.g. smallest angle greater than 15°), each non-conforming point D_i will be processed in constant time.

The following lemma reflects the cost associated with processing a non-conforming edge with multiple non-conforming points.

Lemma 4.8. *Given a non-conforming triangle t with smallest angle α, and set $\{D_1, \ldots, D_i\}$ of non-conforming points located along any edge of the triangle. The longest-edge bisection algorithm inserts, in the worst case, $O(i + \kappa^2)$ new points in order to produce a conforming triangulation inside triangle t, where κ depends on α as defined in Lemma 4.7.*

4.6 Processing Non-Conforming Triangles

As discussed in Sections 4.3 and 4.4, in order to process non-conforming points, the longest-edge bisection algorithm will perform either a constant number of bisections inside a triangle (Lemmas 4.6), or in the worst case, a number of bisections that depends on the quality of the triangle (Lemma 4.7), even if the triangle is affected by multiple non-conforming points (Lemma 4.8). In this section we generalize these results, establishing the cost associated with processing any non-conforming triangle during refinement of a triangulation.

Consider the input triangulation τ with no internal angles smaller than θ, and set $S_{\text{target}} \subseteq \tau$, the set of triangles in τ selected for refinement ($|S_{\text{target}}| = n_1$). Also consider set $S_{\text{prop}} \subseteq \tau$, the set of non-conforming triangles in τ processed by propagated refinement ($|S_{\text{prop}}| = n_2$).

Then, the set of all the triangles in τ processed by the algorithm will be given by $S_{\text{proc}} = S_{\text{target}} \cup S_{\text{prop}}$, of size $n \leq n_1 + n_2$.

In order to refine set S_{target} and produce a conforming triangulation τ', the longest-edge bisection algorithm will perform in the worst case $O(\kappa^2 n)$ bisections and insertions of new points; where n is the number of triangles processed and $\kappa = \lceil \log(\frac{\pi}{6\theta}) / \log(\frac{3}{2}) \rceil$ depends on the quality of the triangulation.

Working with quality acceptable triangulations the value of κ becomes a small constant, and therefore the cost associated with the algorithm becomes linear on n.

The following lemma establishes a bound on the cost of a longest-edge refinement algorithm for the set refinement problem.

Lemma 4.9. *Given a triangulation τ with no internal angles smaller than θ, the set S_{target} of triangles in τ selected for refinement, and the set S_{prop} of triangles in τ processed due to propagated refinement; the longest-edge bisection algorithm inserts at most $O(\kappa^2 n)$ new points in order to produce a conforming, refined triangulation τ' such that every triangle in S_{target} has been refined. $n = |S_{\text{target}}| + |S_{\text{prop}}|$ is the number of triangles effectively refined, and factor $\kappa = O(\log^2 \frac{1}{\theta})$ depends on the quality of the triangulation. Furthermore, for quality acceptable triangulations κ becomes constant.*

□

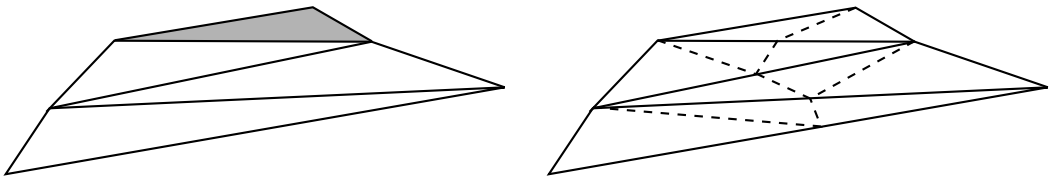
An analysis of the algorithm's propagated refinement, propagating paths, and a bound on the number of triangles effectively processed is discussed in Section 4.7.

4.6.1 Discussion of the Worst Case Scenario

For quality acceptable triangulations the number of new points produced becomes linear on n. For example, working with triangulations with $\theta \geq 30^\circ$ the algorithm would not insert more than $3n$ new points since no triangle would belong to Regions III or IV.b. Even if we allow small-angled triangles, in most scenarios they could be easily processed as shown in Figure 4.17. Our studies found that the worst case scenario corresponds to a configuration of triangles where the longest edge of a bisected triangle is the shortest edge of its neighbor, and this neighbor triangle specifically belongs to Regions III or IV.b.

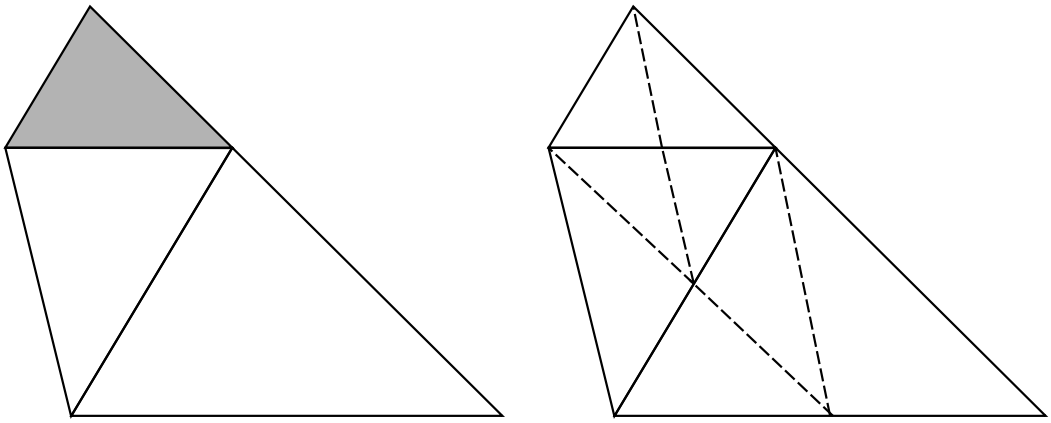
Figures 4.18 and 4.19 show a configuration of triangles where bisections are propagated through non-conforming points over the shortest edge. In Figure 4.18 every triangle belongs to Region I, and just one new point is created per triangle. On the other hand, in Figure 4.19 every triangle belongs to Region IV.b, and several bisections are performed in order to produce the output conforming triangulation.

It must be noted that the propagating path of bad quality triangles, as shown in Figure 4.19, is associated with poorly-graded triangulations. Furthermore, both figures show that



(a) Input triangulation (b) Refined triangulation

Figure 4.17: Example of bisections propagated through a triangulation of bad quality triangles. Shaded triangle is selected for refinement.



(a) Input triangulation (b) Refined triangulation

Figure 4.18: Example of bisections propagated through triangles with non-conforming points over the shortest edge when the triangle belongs to Region I ($\theta = 45°$). Shaded triangle is selected for refinement.

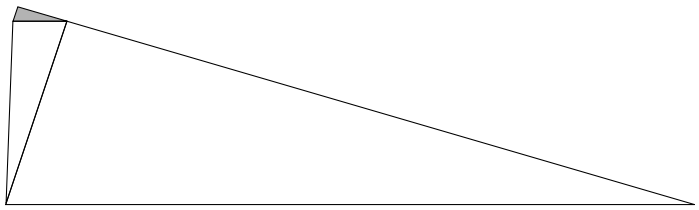
the longest-edge strategy for propagated refinement naturally improves the grading of the triangulation.

Based on these observations, it should be possible to analyze the complexity of the algorithm based not only on the number of processed triangles, n, but also on the actual structure of the instance (i.e., the distribution of bad-quality triangles and how they are affected by propagation). This suggests an adaptive analysis of the algorithm, considering the presence of this configuration of triangles as a *measure of difficulty* as it was studied in other contexts such as sorting algorithms [13] or set operations [11].

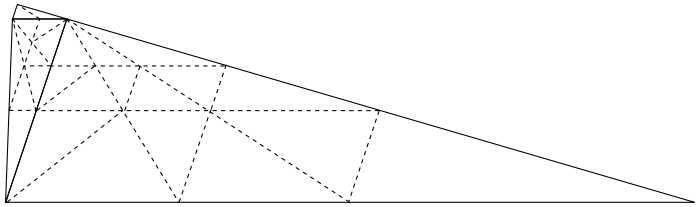
4.7 Analysis of Propagated Refinement

Our study of the longest-edge bisection algorithms considers that they produce a number of N_{prop} new points during propagated refinement of a triangle. The analysis of N_{prop} considers: (1) the number of triangles effectively refined by propagation, and (2) the number of bisections performed in each of these triangles. The latter was discussed in Section 4.6.

The cost associated with processing a triangle was established in Lemma 4.9, so the rest



(a) Initial triangulation



(b) Refined conforming triangulation

Figure 4.19: Example of bisections propagated through triangles with non-conforming points over the shortest edge. Similar triangles belong to Region IV.b, with $\theta = 16°$. The shaded triangle in (a) is initially selected for refinement.

of our analysis focuses on the number of triangles processed due to propagated refinement.

We use previous results on the propagation problem in the 4-Triangles algorithm [65, 68] (a special instance of longest-edge bisection algorithm reviewed in Section 3.1.3). These studies focused on the average number of triangles processed due to propagated refinement throughout global refinement of a triangulation, showing that the number of triangles effectively refined tends to be five as the 4-Triangles algorithm is iteratively applied and terminal triangles populate the triangulation.

In this section we study the propagated refinement for the longest-edge bisection algorithms, as well as the propagating paths and how they are affected by the presence of *terminal triangles* in the triangulation. We show that the propagated refinement is quickly reduced throughout the refinement, only affecting a few triangles.

4.7.1 Extended Propagating Paths

Whenever a triangle is selected for refinement, the algorithm also propagates the refinement to the sequence of longest-edge neighbors in order to produce a refined triangulation. The following fact is useful for our analysis.

Fact 4.10. *Longest-edge propagation is finite since the refinement only propagates to triangles with longer edges.*

Given a target triangle t in the triangulation, the size of $\text{Lepp}(t)$ changes throughout the refinement process. We analyze the evolution of the $\text{Lepp}(t)$ in order to bound the actual extent of the propagation.

Consider a triangle t in triangulation τ, and Lepp(t) the longest-edge propagating path associated with t. As described in Algorithm 3.2, during refinement of triangle t, Lepp(t) will be repeatedly computed until t is bisected. Consider $\text{Lepp}_1(t)$ the initial computation of Lepp(t), where $\text{Lepp}_1(t) \cap \tau = \text{Lepp}_1(t)$. As discussed in Section 3.2.1, $\text{Lepp}_1(t)$ only considers propagation on the longest edge of the triangles in Lepp(t). In some scenarios the $i-$th computation, $\text{Lepp}_i(t)$, could include triangles in τ that were not initially included in $\text{Lepp}_1(t)$, creating a *fork* of the initial propagating path. Figure 4.20 illustrates the different states of the longest-edge propagating path throughout refinement using the Lepp-Bisection algorithm. Figure 4.20(b) shows the initial propagating path of shaded triangle t. Figure 4.20(d) shows the propagating path including triangles of the initial triangulation that were not initially covered.

We extend the concept of longest-edge propagating path discussed in Section 3.2.1 in order to accurately identify the extent of propagated refinement in the initial triangulation.

Definition 4.11. Given a triangulation τ and a target triangle t selected for refinement, we define $\text{Lepp}^*(t)$ as the set of triangles in τ that are affected by the propagated refinement of triangle t. Formally, $\text{Lepp}^*(t) = \tau \cap \{\text{Lepp}_1(t) \cup \text{Lepp}_2(t) \cup \ldots \cup \text{Lepp}_m(t)\}$, where m is the number of iterations until t is bisected.

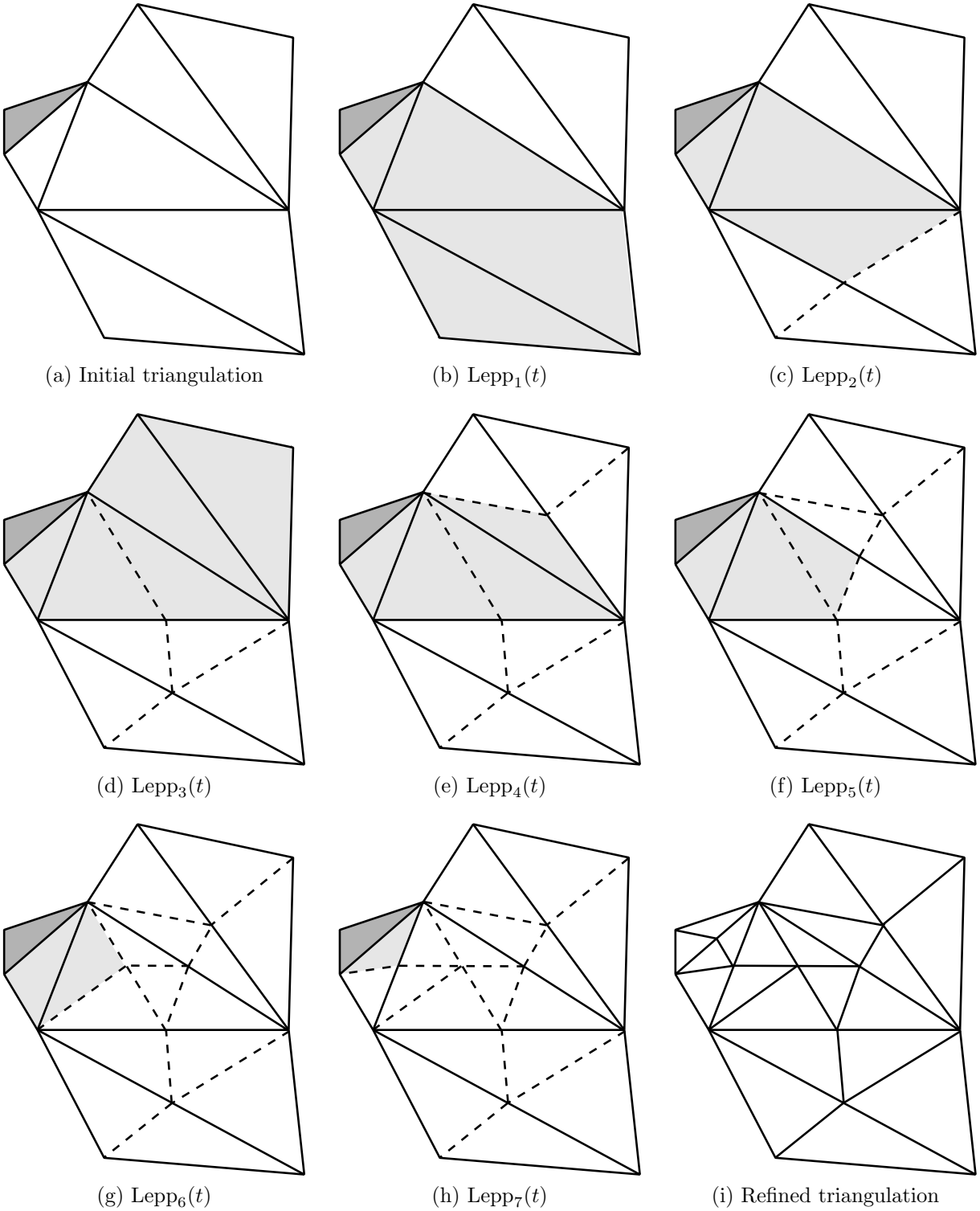
Figure 4.21 shows the different areas in the initial triangulation covered by Lepp(t) and $\text{Lepp}^*(t)$ for a triangle t. The area shaded in Figure 4.21(b) corresponds to the triangles effectively refined by the Lepp-Bisection algorithm. Furthermore, this area also corresponds to the union of every $\text{Lepp}_i(t)$, for $i = 1, \ldots, 7$, shown in Figure 4.20.

This scenario can be also identified based on the behavior of the original longest-edge bisection algorithm. After processing a non-conforming triangle, non-conforming points could be inserted along the midsize or the shortest edge of the triangle. In order to produce a conforming triangulation the algorithm will also process the neighbor triangles on the midsize or shortest edge, thus propagating the refinement to non-longest-edge neighbors according to one of the two following scenarios:

1) Following the analysis in Section 4.4, this is the case of triangles from Regions II and III, and triangles from Region IV.b when they contain a sub-triangle from Region II or III. These triangles insert points along the midsize edge when processing a non-conforming point on their shortest edge (Figure 4.22(b)).

2) Following the analysis in Section 4.5, when processing multiple non-conforming points generated by skinny triangles from Regions III or IV.b, a triangle, regardless of its region, could also insert points along the midsize or shortest edge (Figure 4.22(c)).

Given a triangle t, it is possible to navigate through its propagating path, Lepp(t), and evaluate if one of the conditions described above applies to any $t_i \in \text{Lepp}(t)$. In this way we can identify possible forks in the propagating path and reevaluate the condition in order to find $\text{Lepp}^*(t)$.

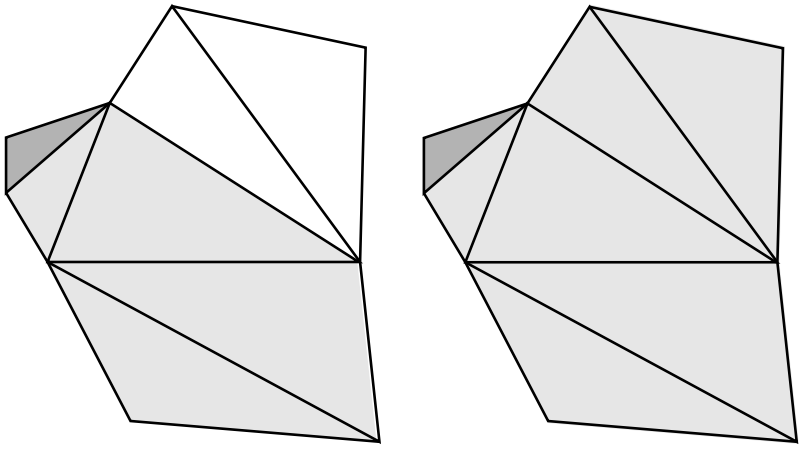


(a) Initial triangulation (b) $\text{Lepp}_1(t)$ (c) $\text{Lepp}_2(t)$

(d) $\text{Lepp}_3(t)$ (e) $\text{Lepp}_4(t)$ (f) $\text{Lepp}_5(t)$

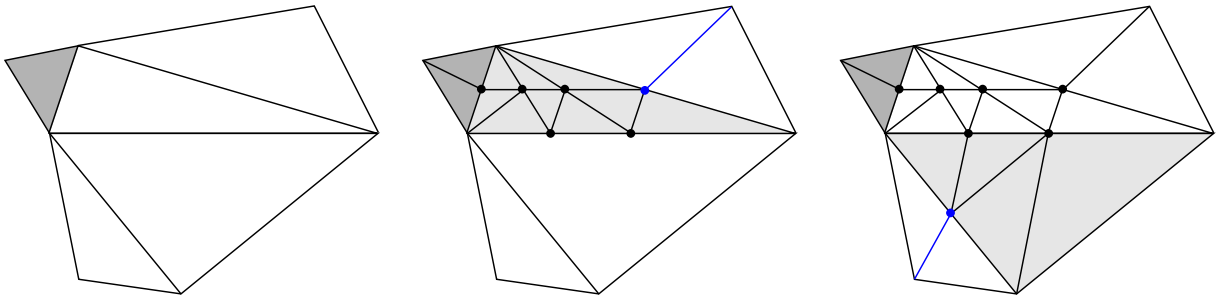
(g) $\text{Lepp}_6(t)$ (h) $\text{Lepp}_7(t)$ (i) Refined triangulation

Figure 4.20: Evolution of a triangle's Lepp during refinement with the Lepp-Bisection algorithm. (a) Initial triangulation. Triangle t (shaded in dark gray) is selected for refinement. (b)-(h) Iterations of terminal-edge bisections. Light shaded triangles belong to the current Lepp(t). (i) Refined triangulation.



(a) Lepp(t) (b) Lepp*(t)

Figure 4.21: Comparison of Lepp(t) and Lepp*(t) for a triangle t (shaded in dark gray). Triangles shaded in light gray belong to the initial propagating path.



(a) (b) (c)

Figure 4.22: Examples of propagated refinement to midsize-edge neighbor and shortest-edge neighbor. Target triangle is shaded in dark gray. (a) Initial triangulation. (b) After processing Region III triangle (shaded in light gray), midsize-edge neighbor is also refined (blue line). (c) After processing Region I triangle (shaded in light gray) with two non-conforming points along its longest edge, shortest-edge neighbor triangle is also refined (blue line).

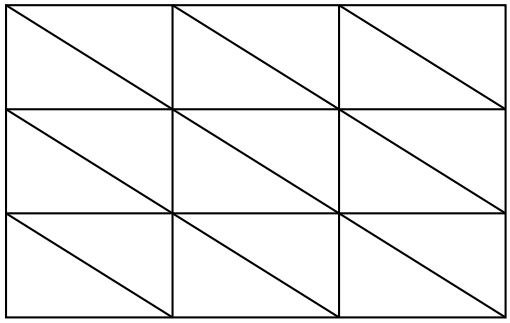


Figure 4.23: Example of a grid-oriented triangulation. Every triangle is a terminal triangle.

The *a priori* computation of the Lepp* of the target triangles could be used to improve workload assignment in multithread implementations of the Lepp-Bisection algorithm [51]. These implementations are usually sensitive to overlapping Lepp* since different processors might try to process a common triangle. Identifying each triangle's Lepp* beforehand could be used to avoid deadlocks.

4.7.2 Propagation and Terminal Triangles

By definition, the propagating paths end in a pair of terminal triangles and therefore the extent of propagated refinement is directly affected by the presence of terminal triangles in the triangulation. For a triangle t in an input triangulation τ, $|\text{Lepp}^*(t)|$, the number of triangles in τ processed by the algorithm is determined by the distribution of triangles (and their neighbors) along the triangulation.

Vilca and Rivara [71] studied the probability of encountering a terminal triangle as the algorithm propagates from one triangle $t_i \in \text{Lepp}(t)$ to the next one t_{i+1}. Since t_{i+1} is the longest-edge neighbor of triangle t_i, the expected length of the propagation can be estimated based on the probability of a distribution of triangles where two adjacent triangles do not share the same longest edge (i.e., they are not terminal triangles). The study found that, considering $|\text{Lepp}(t)|$ as a random variable, the expected number of triangles processed by propagated refinement is 3.

For example, in triangulations where every pair of triangles share a same longest edge, propagating paths will not include more than two triangles. This is the case of triangulations formed by right triangles, or grid-oriented triangulations where points are defined by an underlying grid (Figure 4.23).

By induction, the more frequent terminal triangles are in a triangulation, the shorter the average refinement propagation is since terminal triangles will represent the end of a propagating path. We are interested in measuring the occurrence of terminal triangles in a triangulation, and how it evolves during the longest-edge refinement of triangles.

In what follows we show that the percentage of terminal triangles increases as the refinement proceeds. We apply a methodology similar to the one proposed by Suarez *et al.* [66]

in the study of propagation for the 4-Triangles refinement algorithm [47]. This methodology measures the ratio of terminal triangles in a triangulation, and its evolution during iterative application of a longest-edge bisection algorithm.

Definition 4.12. Given a triangulation τ, $N(\tau)$ the total number of triangles in τ, and $T(\tau)$ the number of terminal triangles in the triangulation; the *ratio of terminal triangles* $R_T(\tau) = \frac{T(\tau)}{N(\tau)}$.

A ratio $R_T(\tau) = 1$ represents a triangulation covered by terminal triangles, with an average propagation of at most 2 triangles (a pair of terminal triangles). On the other hand, a ratio $R_T(\tau)$ approaching zero represents a triangulation covered by a few lepps, maximizing the average size of the propagating paths to $O(N(\tau))$. The scenario of single-lepp triangulation is studied in Section 4.7.3.

Later studies on the 4-Triangles refinement propagation proved that the global application of the algorithm produces triangulations with increasing proportion of terminal triangles, which tend to cover the whole triangulation, with an average length of propagating paths of five triangles [65, 68].

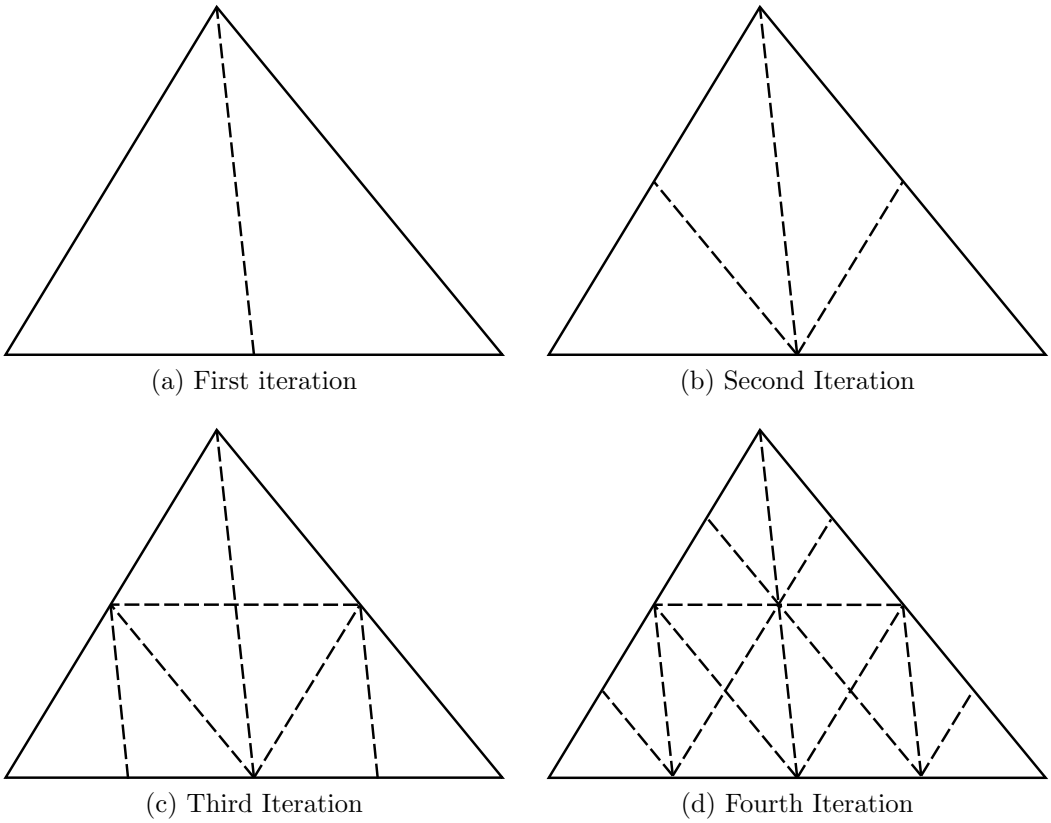
Even though the bisection and propagation patterns of the 4-Triangles algorithm are slightly different than the Lepp-Bisection algorithm or the original longest-edge bisection algorithm, the iterative application of a pure longest-edge bisection algorithm also tends to increase the ratio of terminal triangles, producing triangulations with constant-length propagating paths.

Based on the classification of triangles discussed in Section 4.2, Region I and VI triangles show the same bisection patterns as the 4-triangles partition (Figure 3.2(a)). Previous results on propagation for the 4-Triangles algorithm apply to these quasi-equilateral triangles. Furthermore, the iterative bisection of any quasi-equilateral triangle only produces terminal triangles.

Proposition 4.13. *The global iterative longest-edge bisection of any Region I or VI triangle generates triangulations covered by terminal triangles. (See Figure 4.24)*

A direct consequence of Proposition 4.13 is that, when refining a triangulation covered by quasi-equilateral triangles, the propagation will tend to affect just a few triangles, and the number of triangles processed due to propagated refinement will decrease at the same time that the triangulation is covered by more pairs of terminal triangles.

Additionally, when the longest-edge bisection algorithm processes Region I and VI triangles, only quasi-equilateral triangles are produced inside them (Lemma 2.9) as shown in Figure 4.24. Working with triangles from any other region, their iterative bisection will tend to produce Region I and VI triangles, increasing both the proportion and the area covered by quasi-equilateral triangles (Lemma 2.10). The following definition refers to the ratio of quasi-equilateral triangles in a triangulation.



(a) First iteration

(b) Second Iteration

(c) Third Iteration

(d) Fourth Iteration

Figure 4.24: Iterative refinement of a Region I triangle and all of its descendants. Every triangle is both a terminal and a quasi-equilateral triangle.

Definition 4.14. Given a triangulation τ, $N(\tau)$ the total number of triangles in τ, and $F(\tau)$ the number of quasi-equilateral triangles in the triangulation; the *ratio of quasi-equilateral triangles* $R_{QE}(\tau) = \frac{F(\tau)}{N(\tau)}$.

The iterative application of bisections using a longest-edge bisection algorithm increases R_{QE}, and so the area covered by good-quality, easy-to-process triangles (Lemma 4.9). The following proposition follows from these observations.

Proposition 4.15. *The global iterative application of the Lepp-Bisection algorithm increases the proportion of Region I and VI triangles in a triangulation τ, where $R_{QE}(\tau)$ tends to 1 as the number of iteration increases.*

Proof. Given a triangle t with smallest angle α and largest angle γ, function F_i models the number of triangles from Regions I and VI after the i-th bisection of every sub-triangle in t. The following recurrence relations represent the growth of function F_i for each similarity region:

1. Regions I and VI: $F_i = 2F_{i-1}$, and $F_0 = 1$ (Figure 4.25(a))
2. Region II:$F_i = 2F_{i-1} + 2F_{i-2}$, and $F_0 = 0$, $F_1 = 1$ (Figure 4.25(b))
3. Region III: Region III triangles will behave like Region II for $i > 1.7\log(\frac{\pi}{6\alpha})$, the number of bisections needed to obtain a Region I or II triangle. (See Figure 4.25(c))
4. Regions IV.a and IV.b: These triangles will behave like Region I, II or III triangles for $i > (\gamma - \frac{\pi}{2})/\alpha$, the number of bisections to obtain a Region I or II triangle.
5. Region V: These triangles will not behave worse than Region III.

Function F_i only considers nested bisections of the triangle, hence representing a lower bound on the algorithm's generation of quasi-equilateral triangles. We say that growth of the ratio of quasi-equilateral triangles during iterative applications of the longest-edge bisection algorithm will be faster since additional bisections are internally propagated in order to produce a conforming triangulation. This behavior translates to a faster appearance of quasi-equilateral triangles during a single refinement step. For Region III, IV and V triangles, Lemma 2.9 and Lemma 2.10 ensure the convergence to quasi-equilateral triangles.

Then, as the number of iterations tends to infinity, $\frac{F_i}{N(\tau_i)}$ tends to 1, where τ_i is the triangulation produced inside triangle t on the $i-$th iteration. $N(\tau_i)$, the number of triangles in τ_i, is represented by the recurrence relation $N_i = 2N_{i-1}$ since every triangle is bisected once on each iteration, doubling the number of triangles. □

Figure 4.25 shows the generation of triangles during the iterative longest-edge bisection. Each node corresponds to a triangle obtained after the bisection of its ancestor. Gray arrows

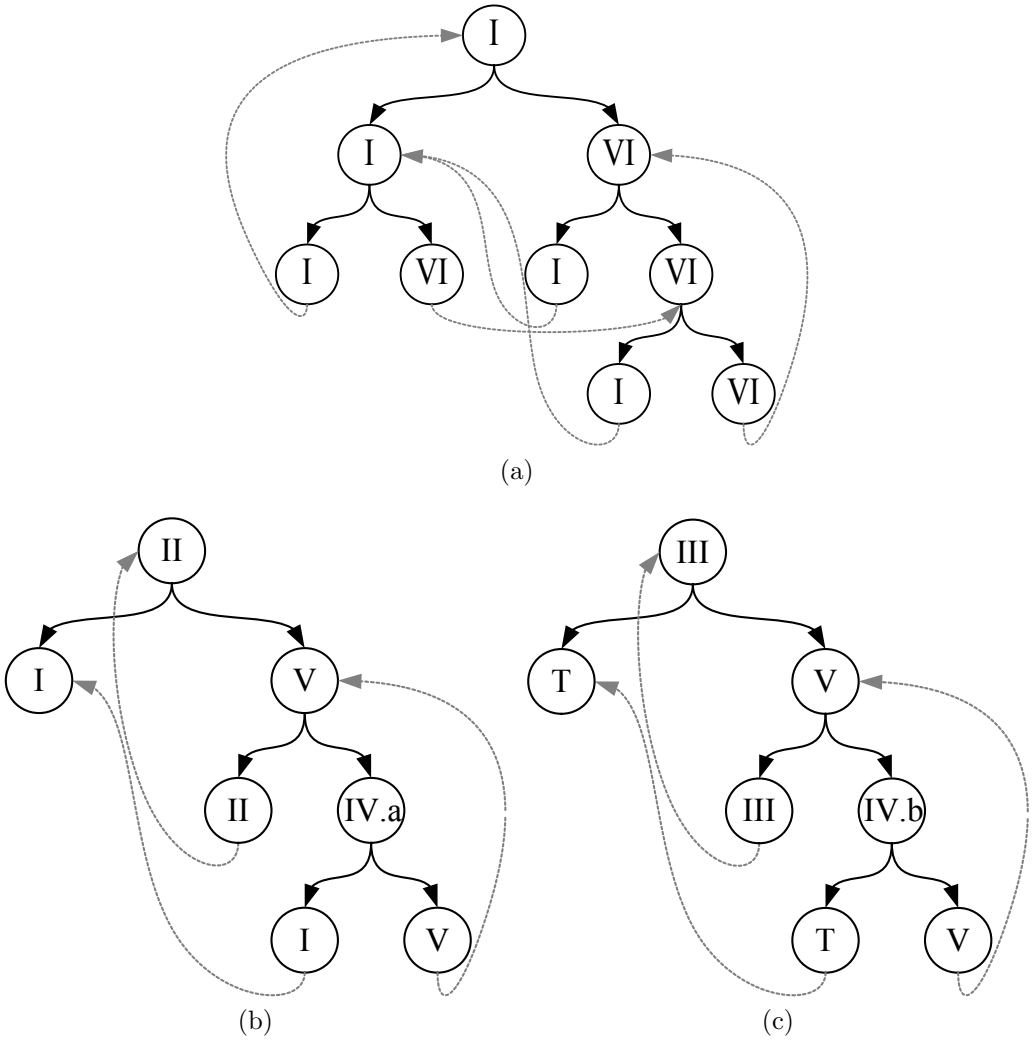


Figure 4.25: Generation of Region I and VI triangles through iterative bisection of a triangle and its descendants. Gray arrows represent the relationship between two similar triangles. (a) Region I and VI triangles. (b) Region II (IV.a and V) triangles. (c) Region III (IV.b and V) triangles. Nodes labeled with a T belongs to Region I, II or III.

show the relationship between similar triangles. Figure 4.25(a) shows the case for the quasi-equilateral triangles until no new non-similar triangles are generated. Every triangle is quasi-equilateral. Figure 4.25(b) shows the graph of a Region II triangle. The left child belongs to Region I. Right child belongs to Region V and produces a new non-similar descendant from Region IV.a. As shown in the graph, this ends the generation of new non-similar triangles. Figure 4.25(c) shows the graph of a Region III triangle. Left child is labeled T because it could belong to Region I, II, or III. Right child belongs to Region V and produces a new non-similar descendant from Region IV.b, which in turn produces two similar triangles. The generation of new non-similar triangles (nodes T in Figure 4.25(c)) stops with the appearance of a Region I or II triangle in a finite number of steps (Lemma 2.9).

For any quasi-equilateral triangle t iteratively refined it holds that, $R_{QE} = 1$. Furthermore, every descendant of t produced by the algorithm will be terminal, so $R_T = 1$, which also proves Proposition 4.13. The following theorem combines the results of Propositions 4.13 and 4.15.

Theorem 4.16. *The iterative application of the Lepp-Bisection algorithm monotonically reduces the average extent of the propagation, tending to affect only one neighbor triangle. Furthermore, the iterative application of longest-edge bisections increases the area of the triangulation covered by good quality triangles.*

Proof. From Proposition 4.15 we know that the iterative application of the algorithm increases the proportion of Regions I and VI triangles in the triangulation. From Proposition 4.13 we know that quasi-equilateral triangles only generate terminal triangles when bisected. Then, the Lepp-Bisection algorithm introduces new terminal triangles in each iteration. As the proportion of good-quality, terminal triangles increases, the average propagation minimizes to the pair of terminal triangles.

Note that Regions I and VI triangles are regarded as good quality triangles, with smallest angle $\alpha > 27.89°$. Therefore, the Lepp-Bisection algorithm not only reduces the cost of refinement after repetitive applications, but it also improves the overall quality of the triangulation due to the increasing number of quasi-equilateral triangles. Using the algorithm for global refinement of a triangulation will quickly increase the ratio of both terminal and quasi-equilateral triangles. □

4.7.3 Discussion of the Worst Case Scenario

Longest-edge bisection algorithms are commonly used for local refinement of triangulations. These algorithms quickly generate non-uniform triangulations refined over target regions, as required by applications such as the adaptive finite element method.

Due to the behavior of the algorithm, the theoretical worst possible scenario is given by a pathological input triangulation covered by quasi-isosceles triangles with small angles (i.e., Region III or IV.b triangles), where the triangles are arranged in such way that the longest edge of a triangle corresponds to the shortest edge of its neighbor (because otherwise only

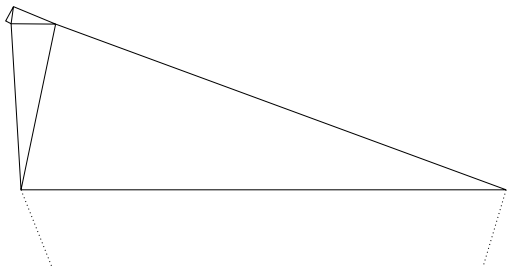


Figure 4.26: Example of a pathological input triangulation.

one or two bisections will suffice to process any non-conforming point). Figure 4.26 illustrates a triangulation with these characteristics.

The triangulation shown in Figure 4.26 will maximize the work associated with processing a triangle and the extent of the propagation. The size of the refined triangulation τ' will be within a factor of the size of the input triangulation τ that depends only on the interior smallest angle θ in τ.

In this scenario, the cost of refining each individual triangle is bounded by $O(\log^2 \frac{1}{\theta})$, and the propagation is bounded by the size of the triangulation. For example, a triangulation covered by triangles such that $\theta = 1°$, will not insert more than ≈ 25 per triangle. Although this value seems to be high, the resulting refined triangulation has a better grading and better average triangle quality.

This pathological input triangulation happens for Region III and IV.b triangles, when the propagation enters the triangle by its shortest edge – otherwise only one or two bisections would be required to eliminate the non-conforming triangle. We can measure the number of occurrences of this specific configuration and use it to estimate the *difficulty* of an instance.

Following this idea, acceptable quality triangulations would represent a difficulty close to zero, as this configuration is uncommon, and non-conforming triangles are eliminated in a constant number of bisections.

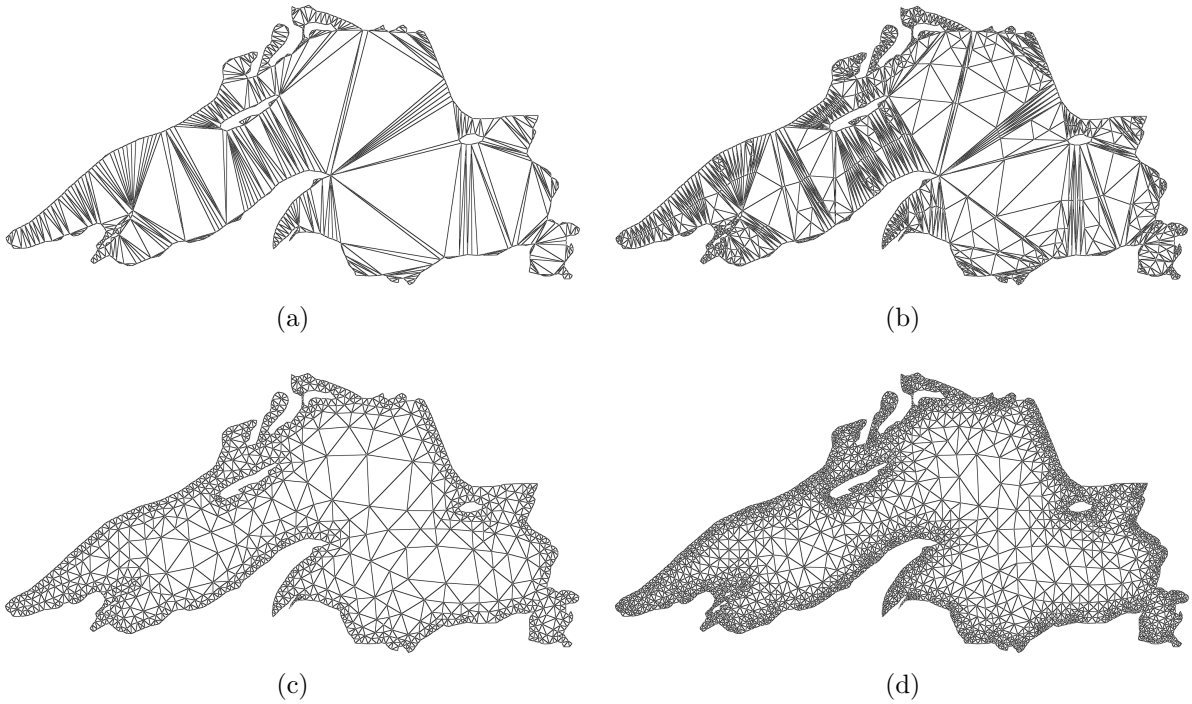
4.8 Experimental Results

In this section we empirically evaluate the performance of the Lepp-Bisection algorithm, and measure the evolution of the quality of the triangulation throughout the iterative application of the algorithm. For different triangulations we measure the cost associated with propagated refinement (average Lepp*), and the percentage of terminal triangles (R_T) and quasi-equilateral triangles (R_{QE}) on each step of the refinement.

We tested the performance of the algorithm using two Delaunay triangulations of the Lake Superior geometry: `Superior-good`, a good quality triangulation with smallest angle $\theta \geq 30°$ (Figure 4.27(c)); and `Superior-bad`, a poor quality triangulation with smallest angle

$\theta \geq 1.6^\circ$ (Figure 4.27(a)).

Although longest-edge bisection algorithms are used in practice over triangulations of acceptable quality, in order to provide a stronger empirical validation of Theorem 4.16, we also evaluated the algorithm's performance using Delaunay triangulations of randomly generated points. We generated three sets of meshes of 1,000, 5,000 and 10,000 points randomly generated inside a quadrilateral region. Each set contained 10 meshes, and the corresponding averaged results per set are labeled as `D1k`, `D5k` and `D10k`.



(a) (b) (c) (d)

Figure 4.27: Lake Superior geometry. (a) Initial triangulation `Superior-bad` (528 triangles). (b) Refined triangulation `Superior-bad` after one iteration of global refinement. (c) Initial triangulation `Superior-good` (1,875 triangles). (d) Refined triangulation `Superior-good` after one iteration of global refinement.

Table 4.4 shows the initial statistics of all the evaluated meshes. The second column refers to the number of triangles in the triangulation. The third column refers to the average smallest angle α of the triangles in the triangulation. The fourth column, Avg. Lepp*, refers to the average number of triangles locally refined, which includes the target triangle and those processed due to propagation. The fifth column refers to the proportion of terminal triangles in the mesh, while the sixth column refers to the proportion of quasi-equilateral triangles (regions I and VI).

It must be noted that even when the average α shown in Table 4.4 is good for the triangulations of randomly generated points, these triangulations contain various initial triangles with a minimum angle α close to 0°. The triangulation of `Superior-bad` also contains some initial poor quality triangles (with α close to 2°).

Table 4.4: Initial statistics of the triangulations evaluated.

Triangulation	Size	Avg. α	Avg. Lepp*	Terminal (%)	Quasi-Equilat. (%)
`Superior-bad`	528	15.28°	9.02	0.12	0.11
`Superior-good`	1,875	42.76°	3.41	39.89	93.92
`D1k`	1,981	29.53°	3.66	28.82	44.67
`D5k`	9,969	30.36°	3.71	29.38	46.37
`D10k`	19,976	30.19°	3.68	29.45	45.71

4.8.1 Iterative Global Refinement

Starting with input triangulations of Table 4.4, we iteratively and globally applied the algorithm to refine every triangle of the preceding triangulation until obtaining one million triangles. Figure 4.28 shows the evolution of four statistics over the refined triangulations during this process. Since all the meshes of random points show analogous behavior we only include results for triangulation `D1k`.

Figures 4.28(a) and 4.28(b) show that both the percentage of terminal triangles and the percentage of quasi-equilateral triangles in the mesh monotonically increases as the refinement proceeds. The same observation holds for the randomly generated triangulations, even though the initial triangulations are of poor quality.

Figure 4.28(c) shows that the average smallest angle tends to about 40° as the triangulations get populated by good quality, easy-to-process triangles. The results shown in Figure 4.28(d) validate our analysis on the reduction of propagating paths during iterative refinement. The average propagation, Lepp*, quickly converges to an average of two triangles (corresponding to a pair of terminal triangles).

Figures 4.29 and 4.30 show the appearance of quasi-equilateral and terminal triangles on triangulations `Superior-good` (after one iteration) and `Superior-bad` (after three iterations) respectively.

We note the direct effect of propagating paths on the number of new points and inserted by the algorithm. After the first iteration, triangulations `Superior-bad` and `Superior-good` grew in average by a factor of 2.4 times. This corresponded to the average length of the propagating paths. The growing rate of triangulations `D1k`, `D5k` and `D10k` converged to 2.4 after two iterations.

This empirical evaluation of the Lepp-Bisection algorithm shows the increasing generation of quasi-equilateral triangles during iterative refinement. As better quality triangles tend to cover the propagating paths future iterations are processed faster, while the triangles with the smallest angles are isolated.

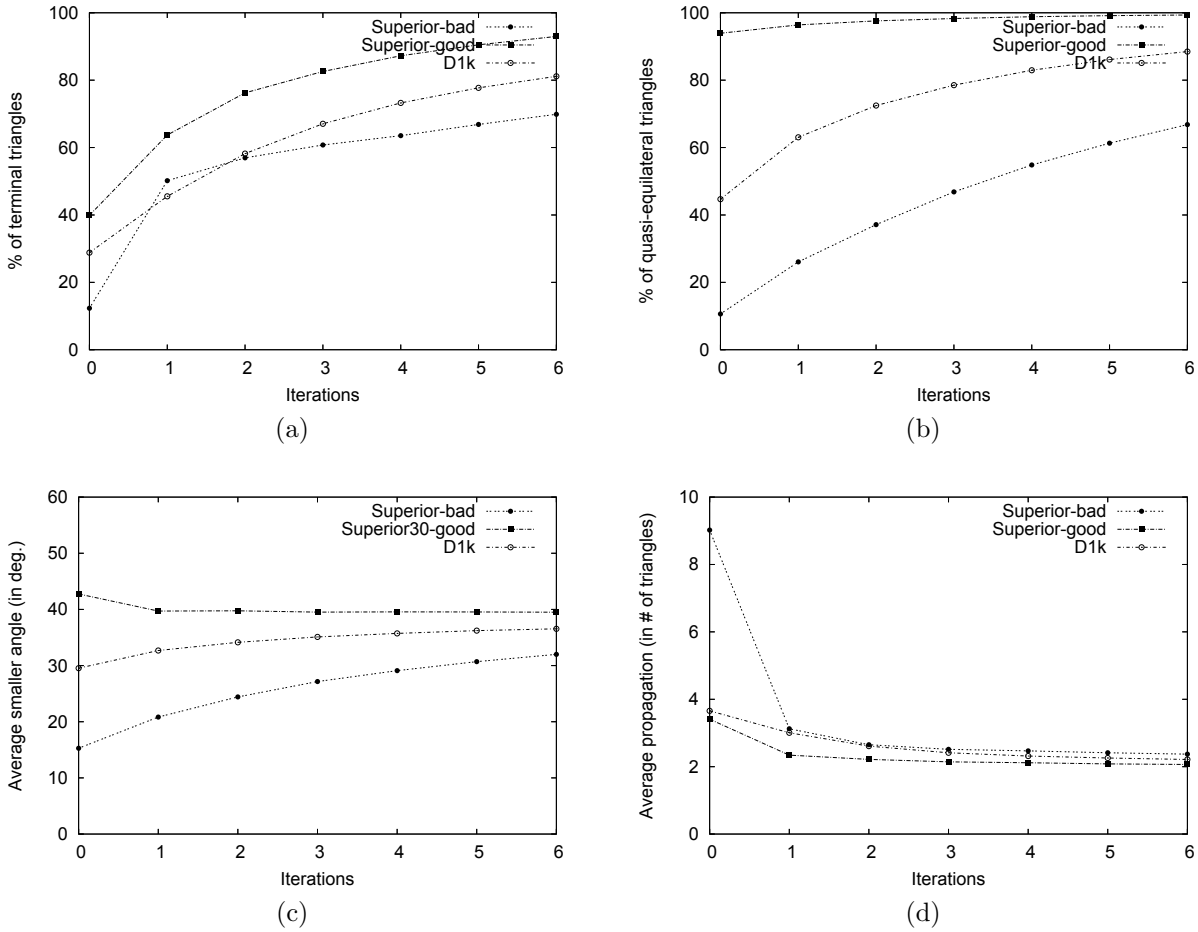


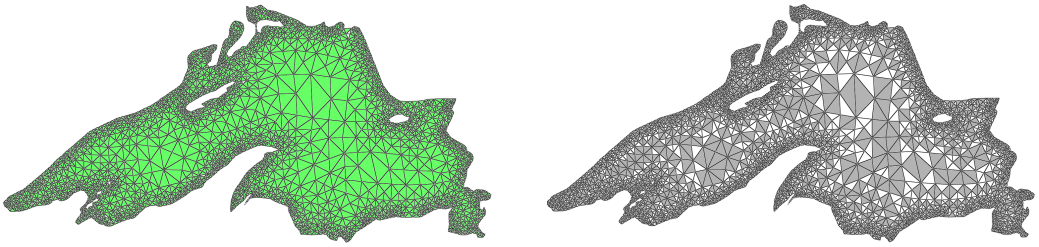
Figure 4.28: Evolution of the refined triangulations for iterative global refinement. (a) Percentage of terminal triangles. (b) Percentage of quasi-equilateral triangles. (c) Average smallest angle (d) Average number of triangles locally refined.

4.8.2 Refinement of Random Triangles

For this set of experiments, at each step we iteratively applied the algorithm to refine one random triangle of the current triangulation for the triangulations of Table 4.4. Since the selected triangle could have been previously refined, we randomly refine a number of triangles equal to the size of the initial triangulation to increase the probability of selecting both refined and unrefined triangles.

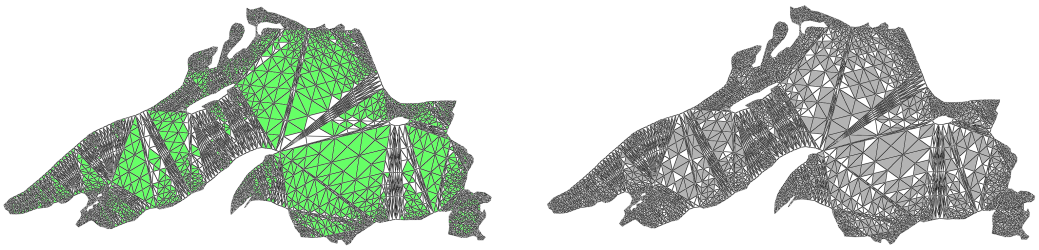
Table 4.5 presents the same statistics as Table 4.4 for the final refined triangulations. An initial comparison of both tables shows an increase of good quality and terminal triangles, as well as the reduction of the Lepps, to an average of three triangles.

For quality-acceptable triangulations, even when local refinement is applied randomly, the overall quality of the triangulation is still improved, with an average minimum angle tending to 40°.



(a) (b)

Figure 4.29: `Superior-good` triangulation after one iterations of global refinement. (a) Area covered by quasi-equilateral triangles (green). (b) Area covered by terminal triangles (gray).



(a) (b)

Figure 4.30: `Superior-bad` triangulation after three iterations of global refinement. (a) Area covered by quasi-equilateral triangles (green). (b) Area covered by terminal triangles (gray).

4.8.3 Iterative Refinement of a Triangle

Following the discussion of Section 4.7.2, in this set of experiments we evaluated the generation of quasi-equilateral triangles and terminal triangles of the iterative application of the Lepp-Bisection algorithm considering individual triangles from each similarity region proposed in Section 4.2. In order to show the angle-improvement property of the longest-edge bisection, we also tracked the evolution of the smallest angles of each in the triangulation.

At each iteration we perform a global refinement of the triangulation and measure the number and area covered by quasi-equilateral triangles, the number of terminal triangles, and the average smallest angle. We selected five representative triangles from each region and averaged the results. Figure 4.31 shows the evolution of the triangulations throughout 10 iterations of global refinement.

Table 4.5: Statistics of the output triangulations after refinement of random triangles.

Triangulation	Size	Avg. α	Avg. Lepp*	Terminal (%)	Quasi-Equilat. (%)
`Superior-bad`	1,923	22.51°	2.99	48.52	32.37
`Superior-good`	5,078	39.93°	2.91	41.78	97.49
`D1k`	5,670	33.85°	3.10	41.64	71.13
`D5k`	28,479	34.23°	3.13	41.11	72.38
`D10k`	56,885	34.62°	3.15	40.57	73.39

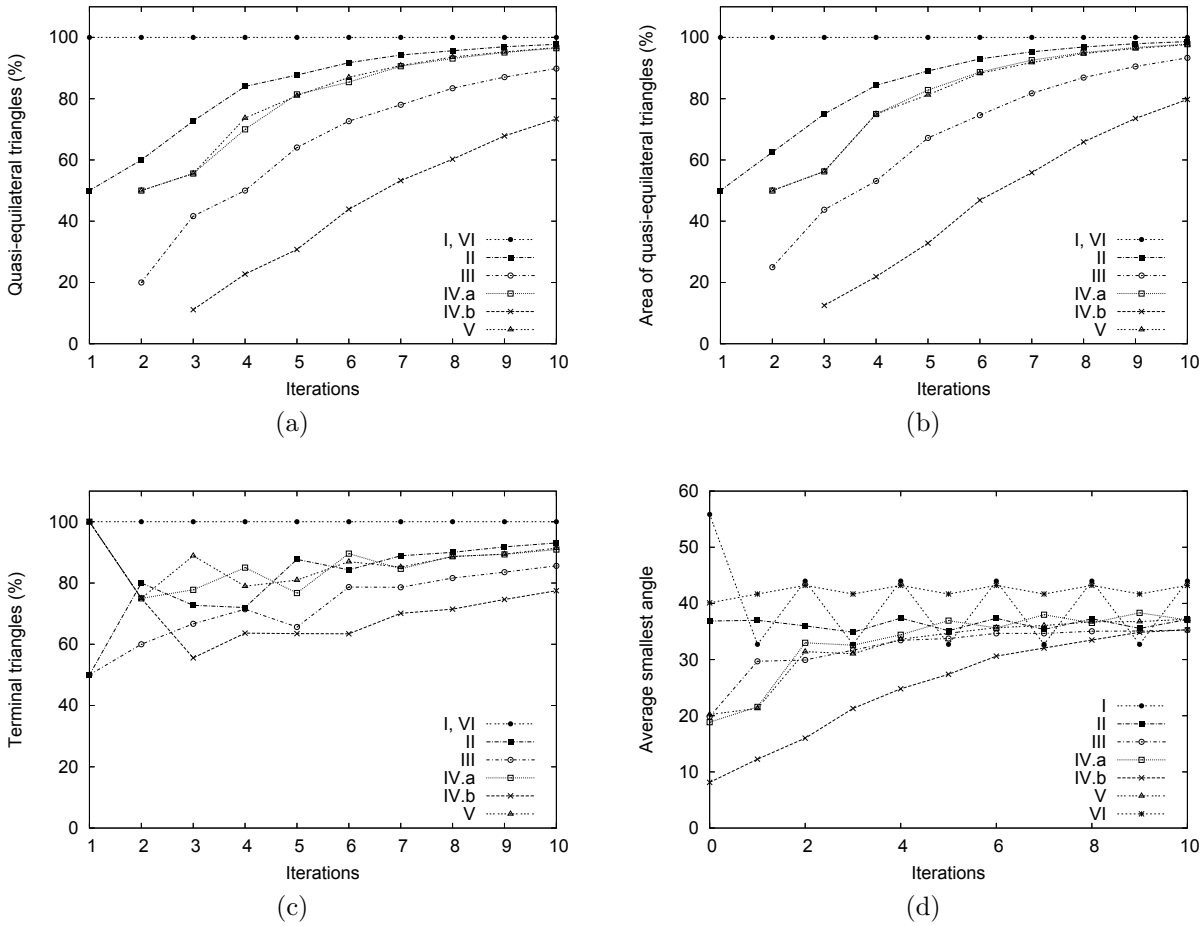


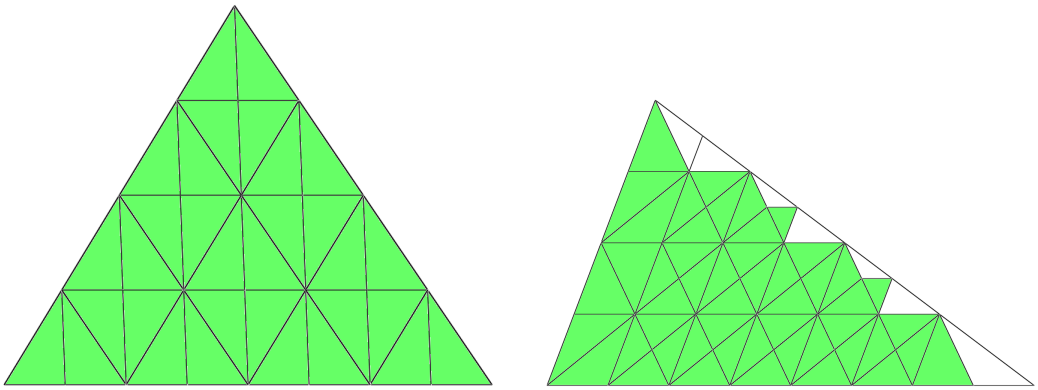
Figure 4.31: Evolution of triangulations during global refinement. (a) Percentage of quasi-equilateral triangles. (b) Percentage of area covered by quasi-equilateral triangles. (c) Percentage of terminal triangles. (d) Average smallest angle.

In Figure 4.31(a) we appreciate the increasing proportion of quasi-equilateral triangles in the triangulation throughout the experiment. Similarly, Figure 4.31(b) shows the increasing proportion of the area covered by quasi-equilateral triangles. These two results show that, for every scenario, the ratio quasi-equilateral triangles monotonically increased at each iteration, with a tendency to cover the entire triangulation.

Figure 4.31(c), on the other hand, shows the also increasing proportion of terminal triangles during iterative refinement. As more terminal triangles populate the triangulation, the average extent of propagated refinement is reduced to just one or two triangles as discussed in Section 4.7.2.

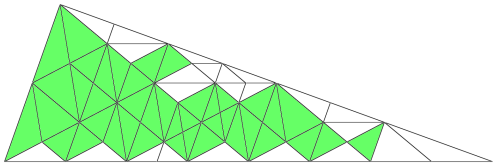
Furthermore, these results show the correlation between quasi-equilateral and terminal triangles and their appearance when triangles from any region are refined. This improves the overall quality of the triangulation while also allowing fast, linear-time refinement using the Lepp-Bisection algorithm.

Figure 4.31(d) shows the evolution of the average smallest angle of the triangles in the

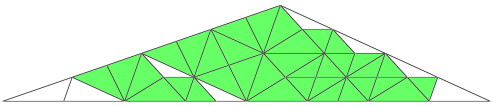


(a) Region I ($\alpha = 55.84$)

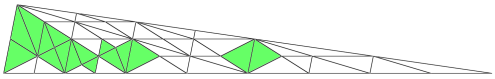
(b) Region II ($\alpha = 36.87$)



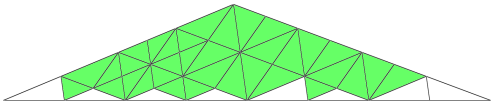
(c) Region III ($\alpha = 19.82$)



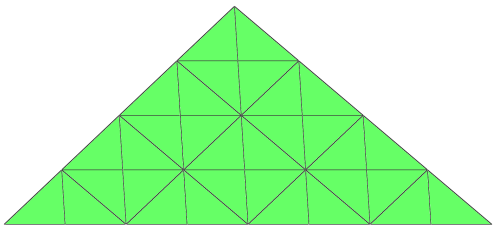
(d) Region IV.a ($\alpha = 18.85$)



(e) Region IV.b ($\alpha = 8.13$)



(f) Region V ($\alpha = 20.22$)



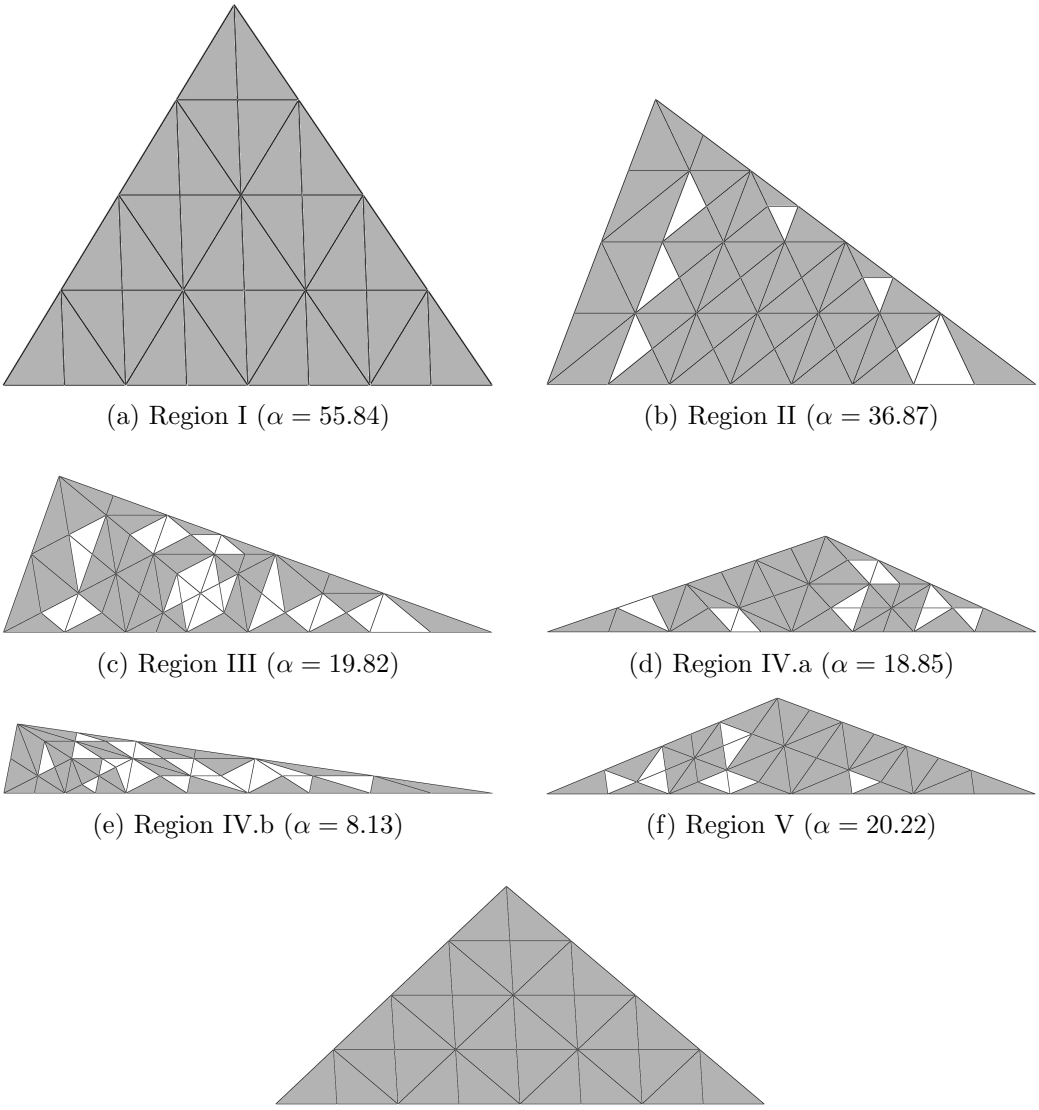
(g) Region VI ($\alpha = 40.11$)

Figure 4.32: Area covered by quasi-equilateral triangles (green) after five iterations of global refinement.

triangulation. As more good-quality triangles appear, the average smallest angle increases, tending to 38°. This also supports the observation that the algorithm would process any triangle in a triangulation faster, since a good-quality triangle is processed in constant time as discussed throughout this chapter.

Finally, in order to visually appreciate the effects of iterative refinement, Figures 4.32 and 4.33 illustrate the output triangulations of a representative triangle from each similarity region obtained after five iterations of global refinement. Figure 4.32 shows the area covered by quasi-equilateral triangles. Figure 4.33 shows the area covered by terminal triangles.

It can be observed that triangles from Regions I and VI are fully covered by quasi-equilateral triangles that also correspond to terminal triangles (Lemma 4.13). Small-angled triangles from Regions III and IV.b present a slower generation rate of good triangles. This



(a) Region I ($\alpha = 55.84$)

(b) Region II ($\alpha = 36.87$)

(c) Region III ($\alpha = 19.82$)

(d) Region IV.a ($\alpha = 18.85$)

(e) Region IV.b ($\alpha = 8.13$)

(f) Region V ($\alpha = 20.22$)

(g) Region VI ($\alpha = 40.11$)

Figure 4.33: Area covered by terminal triangles (grey) after five iterations of global refinement.

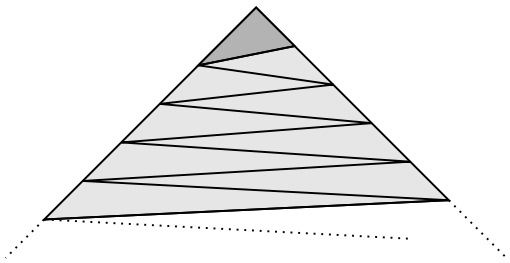


Figure 4.34: Example of propagation affecting an entire triangulation. Shaded triangle (dark) is selected for refinement. Longest-edge propagation also shaded (light).

is closely related to the number of non-similar triangles they can produce since several non-similar sub-triangles generated in the process will not belong to Region I.

4.8.4 Refinement of Single-Lepp Triangulations

So far we have evaluated triangulations covered by relatively small propagating paths. Although the expected size of a Lepp is 3, it is possible to construct pathological input triangulations where the refinement of a target triangle propagates through the entire triangulation; shown in Figure 4.34.

In practice, the average propagation only affects a few number of neighboring triangles. The scenario aforementioned rarely appears in practice, and is only evaluated for theoretical purposes.

For the sake of a more complete empirical analysis, we evaluated triangulations representing this theoretical scenario. Figure 4.35 shows three examples of triangulations covered by a single Lepp. We chose triangulations of different angle qualities: 43°, 18° and 5°. Triangulations of the same quality produced the same behavior of the algorithm during the refinement process regardless of their initial size. This showed that the quality improvement properties of the algorithm remained consistent working with these triangulations.

As with previous experiments, we evaluated the improvement properties of the Lepp-Bisection algorithm measuring the evolution of the triangulation during iterative global refinement. Figure 4.36 shows the evolution of the percentage of terminal triangles, the average number of triangles affected by propagation and the average small angle after each iteration.

The results are similar to those obtained during the previous experiments:

- terminal triangles immediately start to cover the triangulation,
- the average propagations converges to approximately 2 triangles, and
- the average smallest angle tends to 40°.

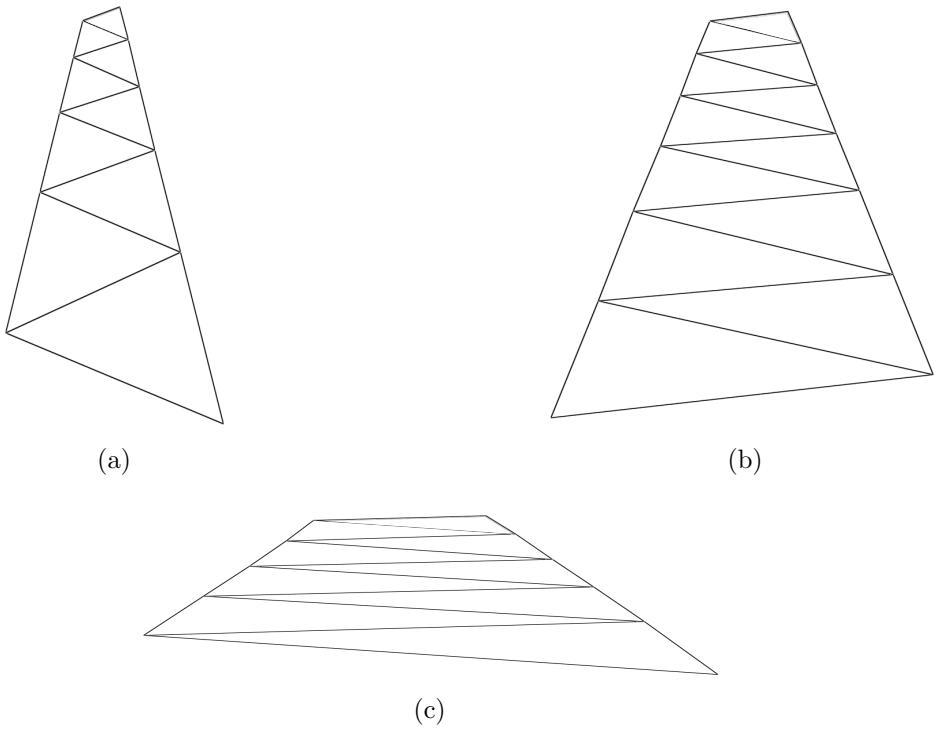


Figure 4.35: Examples of triangulations covered by a single Lepp. (a) `Lepp-Deg43`: 9 triangles with smallest angle of 43°. (b) `Lepp-Deg18`: 12 triangles with smallest angle of 18°. (c) `Lepp-Deg5`: 9 triangles with smallest angle of 5°.

Although not shown in Figure 4.36, the area covered by quasi-equilateral triangles monotonically increased for triangulations `Lepp-Deg18` and `Lepp-Deg5`. Triangulation `Lepp-Deg43` stayed fully covered by good-quality triangles throughout the experiments.

4.9 Conclusions

This chapter presented a new study of the complexity of the longest-edge bisection algorithm for the refinement of triangulations in two dimensions. We studied the computational cost of the algorithm in terms of the size (number of elements) of the refined triangulations. We proved that the size of the triangulations produced by the Lepp-Bisection algorithm is at most a constant factor larger than the size of the initial triangulation. We also presented a complete study of refinement propagation for the Lepp-Bisection algorithm, proving that the algorithm can produce size-optimal triangulations.

Our study was based on the behavior of refinement propagation. We focused the analysis on the non-trivial scenario where non-conforming points affect the shortest edge of a triangle, since this is the worst case scenario for the point selection strategy to maintain a conforming triangulation. Working with bad-quality triangles the growing factor will depend on the geometry of the triangulation independent of the number of triangles. A previous version of these results were published in [5, 6].

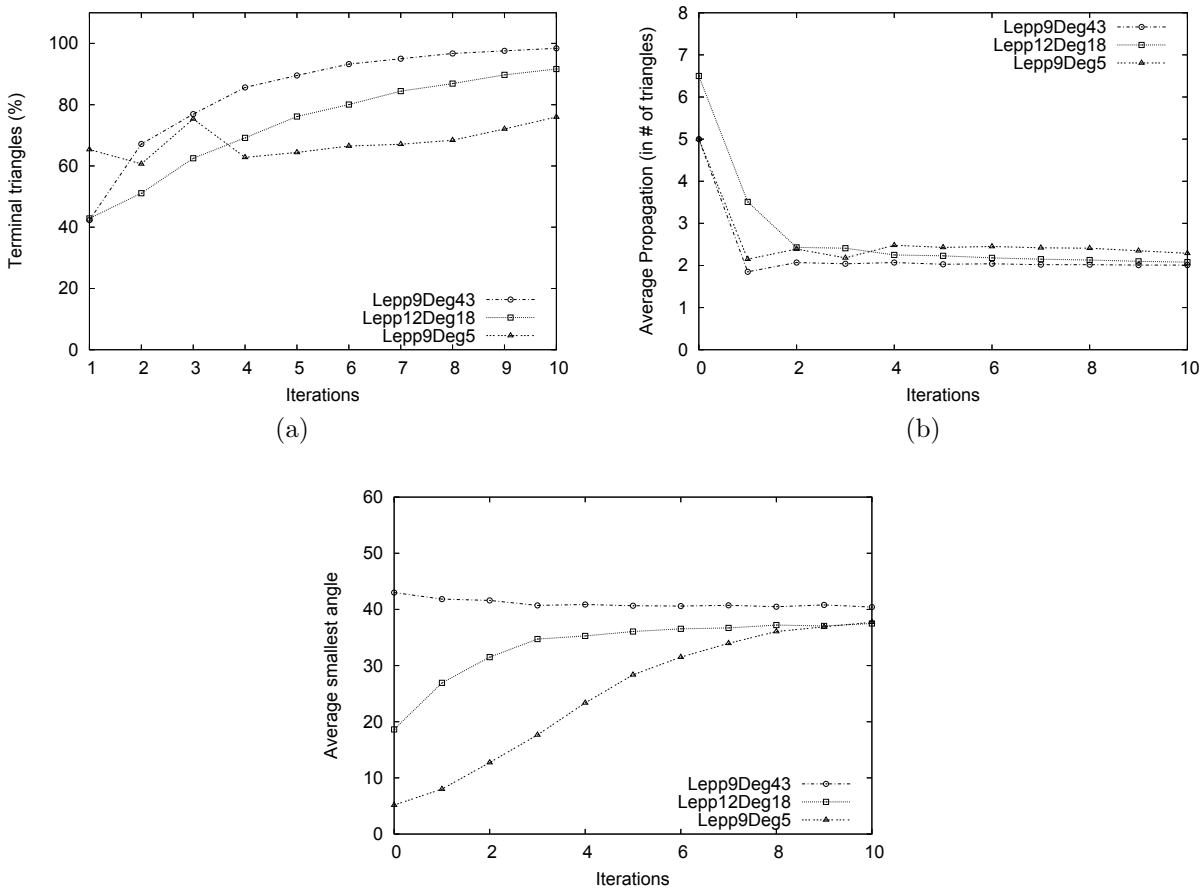


Figure 4.36: Evolution of the single-Lepp triangulations showed in Figure 4.35 after iterative global refinement. (a) Percentage of terminal triangles in the triangulation. (b) Average number of triangles affected by propagation. (c) Average smallest angle in the triangulation.

The properties of the longest-edge bisection of triangles not only offers bounds on the minimum angle and on the number of non-similar triangles generated, but also supports the generation of quasi-equilateral triangles during iterative bisection and the reduction of the cost of processing triangles.

Both the proportion of quasi-equilateral triangles and terminal triangles increase during iterative application of longest-edge bisection. This behavior ensures the construction of irregular and nested triangulations of quality analogous to the input triangulation (in terms on the minimum angle).

We showed that the Lepp-Bisection algorithm improves the global quality of the triangulation, creating good-quality sub-triangles and isolating the triangles with small angles.

We also showed the direct effect of the size of the propagation and the quality of the triangles on the number of points and triangles produced by the longest-edge bisection algorithm. In practice, these two factors quickly converge to a small constant after a few steps, making the algorithm's cost asymptotically optimal.

Our theoretical results present new bounds on the size of the output triangulation produced by a longest-edge bisection algorithm. We prove that the size of the output triangulation will be at most a constant factor larger than the size of the input triangulations.

For any size-optimal input triangulation, the longest-edge refinement algorithm will produce a size-optimal output triangulation: the refined triangulation maintains the shape bounds, and its size remains within a constant factor of the input triangulation.

On the other hand, triangulations which are not size-optimal (such as those covered by small-angled triangles or non-Delaunay triangulations) will benefit from the generation of quasi-equilateral triangles. This effectively isolates existing small-angled triangles and populates the triangulation with triangles of good quality.

In practice, the longest-edge bisection algorithm performs in time linear to the number of triangles selected for refinement, since only a constant number of additional triangles are effectively processed in order to produce a conforming triangulation.

Other refinement operations such as refinement around a vertex, refinement around an edge, or refinement inside a region, present the same behavior as the iterative global refinement studied in this thesis. Our results can be extended to formally address the complexity of these refinement operations.

The Lepp-Bisection algorithm represents an efficient technique for fast and robust refinement suitable for applications where a good-quality input triangulation (or one with a few number of small-angled triangles) needs to be quickly refined in order to be covered by smaller good-quality triangles.

Chapter 5

The Lepp-Delaunay Algorithm and Size-Optimal Refinement

Longest-edge bisection algorithms produce refined triangulations with a number of elements within a constant factor the number of elements in the initial triangulation. The bounds on the minimum angle and on the number of non-similar triangles generated are supported by the properties of the longest-edge bisection of triangles. In practice, longest-edge bisection algorithm quickly produce nicely graded triangulations populated by good quality triangles. The geometric quality of the triangulation will remain the same since small-angled triangles are not eliminated, only isolated.

The Lepp-Delaunay algorithm deals with the automatic generation of triangulations of planar straight-line graphs supported by the mathematical properties of the longest-edge bisection of triangles and the Delaunay triangulation.

In this chapter we extend the analysis of the Lepp-Delaunay algorithm for the quality triangulation of PSLG geometries. We provide stronger proofs on the algorithm's termination and on the generation of asymptotically size-optimal triangulations.

Roughly speaking, the algorithm considers two refinement scenarios we need to analyze: (1) when the algorithm introduces the midpoint of the terminal edge associated with a terminal triangle, and (2) when a terminal triangle has a constrained midsize edge, the algorithm introduces the midpoint of such an edge. The first scenario is the default behavior of the algorithm, while the second scenario is related to the insertion of points near input segments or near the boundary of the geometry.

In Sections 5.3 and 5.4 we analyze the first scenario, stating bounds on how close together two points can be. For various quality bounds, we show in Section 5.5 that points inserted by the algorithm cannot be arbitrarily close to existing points of the triangulation, and consequently the length of new edges cannot be arbitrarily short. In Section 5.6 we analyze the refinement of triangles with constrained edges, and propose an improved point selection strategy for constrained edges. We also study the angles at which adjoining constrained edges

can meet without affecting the performance of the algorithm.

Using these results, in Section 5.7 we adapt the methodology introduced by Ruppert [53] for the analysis of Delaunay refinement algorithms, providing the first formal proofs of the algorithm's termination, good grading, and size-optimality.

5.1 Lepp-Delaunay Algorithm and Ruppert's Refinement Algorithm

Ruppert [53] presented the first provably good Delaunay refinement algorithm with theoretical guarantees of size-optimality and good grading, and with good performance. Ruppert's algorithm is based on the insertion of circumcenter of bad quality triangles (and special treatment of input segments) in order to produce a Delaunay triangulation of the input PSLG with the desired geometric quality. However the implementation of Ruppert's algorithm has some drawbacks:

- The algorithm is sensitive to the order in which bad quality triangles are processed. Although the processing order does not affect the theoretical bounds of the algorithm, an implementation that prioritizes triangles with the smallest angles could reduce the number of triangles generated [9]. Building and maintaining priority queues introduces additional cost and complexity to the implementation.
- Ruppert's algorithm use the concept of "encroachment" to avoid the insertion of points too close to constrained (interior or boundary) edges, and to prevent scenarios where the circumcenter of a bad quality triangle falls outside of the geometry. If a point lies inside the diametral circle of a constrained edge, it encroaches upon the edge and the algorithm inserts the midpoint of the edge instead. In practice, constrained edges are required to be processed first.
- After inserting the circumcenter of a triangle the algorithm requires additional point location strategies to identify the triangles affected by the new point. Managing the numerical error associated with these geometric operations increases the complexity of a robust implementation of the algorithm.
- The proofs of Ruppert's algorithm are limited to geometries where segments meet at angles of more than 90°, since geometries with acute angles could threaten the algorithm's termination. In practice, the algorithm is not guaranteed to terminate when segments meet at angles smaller than 60°.

In contrast, an implementation of the Lepp-Delaunay algorithm has the following advantages over Ruppert's algorithm:

- The insertion of new points is a robust operation. The point selection strategy selects the midpoint of an edge and does not require additional computations nor depend on complex operations (as discussed in Section 3.1.1).
- The algorithm works over the constrained Delaunay triangulation of the geometry. There are no issues associated with a bounding box or processing triangles outside of the actual geometry.
- Our proofs hold for any geometry with input angles of at least 35°. In practice the algorithm works well and terminates working with input angles of 30°.
- The point insertion strategy for constrained edges is simple, and successfully prevents the algorithm from inserting points close to constrained edges.
- The order in which bad quality triangles are processed is irrelevant. This feature support the efficient implementation of the Lepp-Delaunay algorithm in parallel environments.
- Previous studies limited the minimum angle formed by input segments to 60°. In this thesis we prove that the algorithm is guaranteed to terminate for geometries with input angles greater than 35°.

5.2 Preliminaries

As stated in Proposition 3.10, the intrinsic bound on the largest angles of Delaunay terminal triangles defines the region where the Lepp-Delaunay algorithm could insert new points. Furthermore, Corollary 3.12 provides a way to define a bound on the distance between points selected for insertion and existing points of the triangulation.

In this chapter we provide improved bounds on this inter-point distance, so we can guarantee that new points cannot be inserted arbitrarily close to each other, and correspondingly, that new edges cannot be arbitrarily small.

We introduce the concept the concept of insertion radius of, formally defined below.

Definition 5.1. Given a point p and triangulation τ, the *insertion radius* $r_p = d(p, \tau)$ represents the distance from p to the nearest point of τ right before inserting p in the triangulation. After the insertion, r_p becomes the length of the shortest edge adjoining p in the triangulation.

Figure 5.1 shows three possible scenarios for the insertion radius r_p of the midpoint p of a terminal edge. In Figure 5.1(a) the insertion radius is the distance from point p to one endpoint of the longest edge. In Figure 5.1(b) the insertion radius is the distance from point p to the vertex opposite the longest edge. In Figure 5.1(c) we show the case where the point closest to p in the triangulation belongs to a triangle not sharing the terminal edge. The only possible scenario corresponds to the vertex opposite to the longest edge of the midsize-edge neighbor. The insertion radius is the distance from p to this vertex.

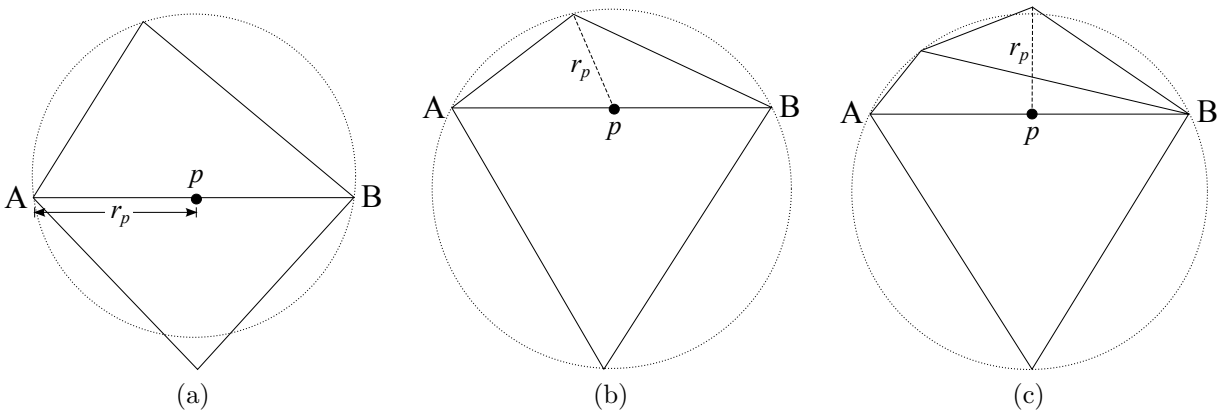


Figure 5.1: The insertion radius of the midpoint p of a terminal edge. (a) The nearest point is one endpoint of the longest edge. (b) The nearest point is the vertex opposite the longest edge. (c) The nearest point is a vertex not belonging to the terminal triangles. The only candidate is the vertex opposite to the longest edge of the midsize-edge neighbor.

□

Throughout our analysis we make use the following geometric property based on the law of sines.

Fact 5.2. *Let $t(ABC)$ be a triangle with edges $|AB| \geq |BC| \geq |CA|$ and angles $\gamma \geq \beta \geq \alpha$. Let r be the circumradius of t. Then:*

- $|AB| = 2r \sin \gamma$
- $|BC| = 2r \sin \beta$
- $|CA| = 2r \sin \alpha$

Figure 5.2 illustrates the relationship between the smallest angle and the shortest edge of a triangle as defined in Fact 5.2. Triangles AOC and BCO are isosceles, so angle $\angle AOB = 180° - 2\sigma$ and $\angle COB = 180° - 2\sigma - 2\alpha$. It follows that $\phi = \angle AOB - \angle COB = 2\alpha$ and shortest edge $|CA| = 2r \sin \alpha$.

□

Next, we establish a relationship between a terminal triangle and the target triangle that originated the propagated refinement. This relationship will be used to analyze how a terminal triangle (and the edges generated after its refinement) compares to the initial bad-quality triangle that produced the refinement.

Definition 5.3. Let t be a target triangle in S_{target} and t_i be a Delaunay terminal triangle in Lepp(t). We say that t is the *parent triangle* of t_i, denoted $\hat{t}_i$, since t is the triangle that originated the propagated refinement.

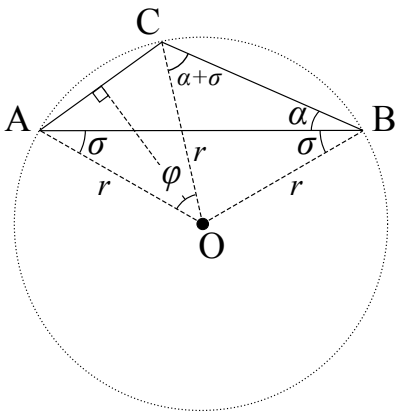


Figure 5.2: Relationship between shortest edge $l = AC$, smallest angle α and circumradius r of a triangle $t(ABC)$.

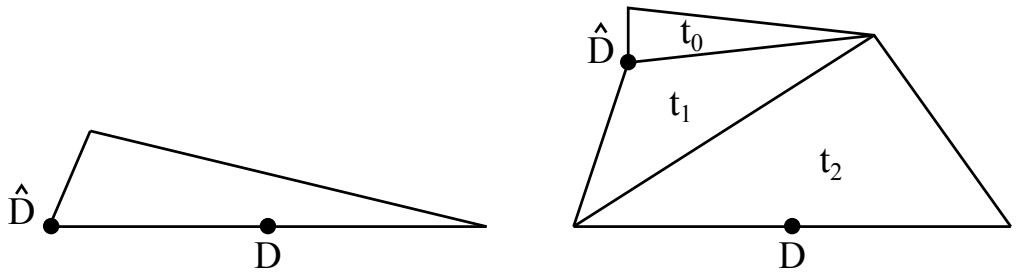


Figure 5.3: Point D and its parent point $\hat{D}$. (a) When the triangle is both a terminal triangle and the current target triangle processed. (b) Triangle t_2 is the terminal triangle in Lepp(t_0), $\hat{t}_2 = t_0$.

We also define the relationship between the point selected by the algorithm and the vertex of the parent triangle "responsible" for the insertion of the new point. This relationship will be used to analyzed how close a new point is to existing points, and how this distance compares to the shortest distance between vertices of the initial bad-quality triangle that produced the refinement.

Definition 5.4. Let point D be the midpoint of the longest edge of a Delaunay terminal triangle t. The *parent point* of D, denoted $\hat{D}$, is the vertex shared by the longest and shortest edges of the target triangle $\hat{t}$ responsible for the selection of D.

Figure 5.3 illustrates the relationship between D and its parent $\hat{D}$. In Figure 5.3(a) the target triangle is at the same time terminal, hence responsible for the insertion of D. In Figure 5.3(b) the terminal triangle (t_2) belongs to the Lepp of a target triangle (t_0).

When the terminal triangle has a constrained midsize edge, the algorithm selects the midpoint of the constrained edge instead of the midpoint of a terminal edge. The parent

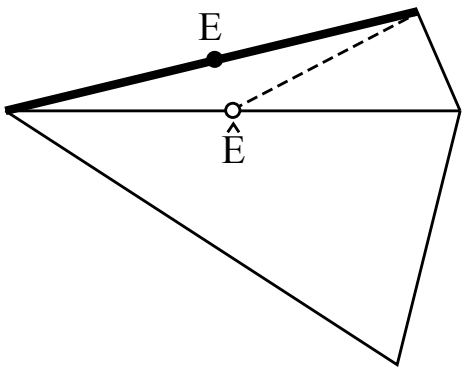


Figure 5.4: Point E is the midpoint of a constrained midsize edge (in bold) of a terminal triangle. The parent point of E, $\hat{E}$, is the midpoint of the terminal edge. In such a case E is inserted instead of $\hat{E}$.

point of a midpoint of a constrained edge is defined below.

Definition 5.5. Let point E be the midpoint of the constrained midsize edge of a Delaunay terminal triangle t. The *constrained parent point* of E, denoted $\hat{E}$, is the midpoint of the longest edge of t.

We make a distinction with constrained edges because in this case it is the midpoint of the longest edge the one responsible for the selection of E. As we will discuss in Section 5.6.1, a constrained midsize edge is bisected because the midpoint of the triangle's longest edge lies too close to E. Figure 5.4 illustrates the relationships between E and its parent $\hat{E}$.

5.2.1 Processing Order for Target Triangles

We define a processing order of target triangles that will let us establish assumptions about the geometry of terminal triangles. The idea of a processing order is incorporated to facilitate the analysis of the algorithm. In practice, a processing order is not necessary nor it affects the outcome of the refinement since the algorithm is order-independent.

We assume that the algorithm processes the target triangles in set S_{target} in a specific order, selecting at each step the triangle in S_{target} with the smallest edge l_{min}. The convenience of such processing order is justified as follows.

Lemma 5.6. *Let triangle $t_{\text{min}} \in S_{\text{target}}$ be the target triangle with smallest edge. Let t_i be a triangle in* $\text{Lepp}(t_{\text{min}})$. *Let $l_{t_{\text{min}}}$ and l_{t_i} be the length of the shortest edge of t_{min} and t_i respectively. Then, $l_{t_{\text{min}}} < l_{t_i}$.*

Proof. We prove the lemma by contradiction. Consider $\alpha_{t_{\text{min}}}$ the smallest angle of triangle t_{min} and α_{t_i} the smallest angle of t_i. Due to the properties of longest-edge propagation, if $l_{t_i} \leq l_{t_{\text{min}}}$ then $\alpha_{t_i} < \alpha_{t_{\text{min}}}$. Therefore, triangle t_i actually belongs to set S_{target} and t_{min} is not the triangle in S_{target} with the smallest edge. □

It must be noted that smallest edge does not necessarily correspond to the smallest angle since t_i could have a smaller angle than $t_{\min}$. If this is the case, t_i is necessarily more obtuse than t.

Other potential orderings could include selecting the triangle with the worst radius-edge ratio, or the triangle with the smallest angle. We refer to these properties throughout the analysis of the Lepp-Delaunay point selection strategy.

Due to the Lepp strategy, the actual order in which the algorithm processes the set of target triangles S_{target} does not affect the final refined triangulation. Other Lepp-based algorithm, such as the Lepp-Bisection algorithm, also share this property (discussed in Section 5.2.1).

5.3 Insertion Radius and Terminal Triangles

In this section we analyze the insertion radius of the points selected for insertion by the Lepp-Delaunay algorithm. We define new bounds on the insertion radius based on the geometry of the terminal triangles.

We propose two different approaches to set the limits on the insertion radius of the midpoint of a terminal edge. Our first bound is expressed in terms of the circumradius of the terminal triangles (Section 5.3.1). The second proposed bound is expressed in terms of the shortest edge of the terminal triangles (Section 5.3.2). Then, we analyze the case of midpoints of constrained midsize edges, defining geometrical bounds on the insertion radius using both the circumradius of the terminal triangles and the smallest edge of such triangles (Section 5.3.3).

5.3.1 Bounds Based on the Circumradius

We start with the geometric analysis of the different scenarios of terminal triangles. We analyze the distance from the midpoint of the longest edge to the vertices of the triangle for each possible geometry. Then, we define the bounds for the insertion radius of the selected point based on the circumradius of the terminal triangles.

Consider triangle $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$, angles γ and α respectively the largest and smallest angles of t, r the circumradius of t, and point D the midpoint of longest edge AB. We analyze the distance from point D to t's vertices for different geometries of triangles:

- t is an *obtuse triangle* with $\gamma = 120°$. This is the scenario for a terminal triangle with the largest possible angle. Then, (i) $d(D, A) = r \sin 120° = \frac{\sqrt{3}}{2} r$; and (ii) $d(D, C)$ will

depend on angle α given by the formula

$$d(D,C) = \frac{r}{2}\sqrt{8\sin^2\alpha + 2(\sqrt{3}\cos\alpha - \sin\alpha)^2 - 3}. \tag{5.1}$$

It must be noted that the least possible value of $d(D,C)$ is $r/2$, which happens when t is an isosceles triangle with $\alpha = 30°$; see Figure 5.5(a). On the other hand, as α decreases $d(D,C)$ increases, tending to $\frac{\sqrt{3}}{2}r$ as α tends to 0°; see Figure 5.5(b).

- t is an *obtuse triangle* with $\gamma < 120°$. Distance to any of the vertices increases as point D approaches to the circumcenter of t. Then, (i) $\frac{\sqrt{3}}{2}r < d(D,A) < r$ since $d(D,A) = r\sin\gamma$; and (ii) $r/2 < d(D,C) < r$ since $d(D,C)$ has the least possible value for isosceles triangles where $d(D,C) = r(1+\cos\gamma)$; see Figure 5.5(c).

- t is a *right triangle* ($\gamma = 90°$). In this scenario midpoint D corresponds to the circumcenter of t. Then, $d(D,A) = d(D,B) = d(D,C) = r$; see Figure 5.5(d).

- t is an *acute triangle* ($60° \leq \gamma < 90°$). Distance to the vertices will depend on how far point D lies from the circumcenter of t. Then, (i) $\frac{\sqrt{3}}{2}r \leq d(D,A) < r$, since $d(D,A) = r\sin\gamma$ has its least possible value when t is equilateral and maximized either when α tends to 0° or γ tends to 90°; and (ii) $r < d(D,C) \leq \frac{3}{2}r$ since point D lies farther away from C than from the circumcenter, maximized for the equilateral triangle; see Figures 5.6(a) and 5.6(b).

The observations above are summarized in the following lemma.

Lemma 5.7. *For any Delaunay terminal triangle $t(ABC)$ with circumradius r, the distances from the midpoint D of its longest edge AB to the triangle's vertices are bounded as follows:*

- $d(D,A) = d(D,B) \geq r\sqrt{3}/2$, *and*
- $d(D,C) \geq \begin{cases} r/2 & \text{if } t \text{ is obtuse} \\ r & \text{if } t \text{ is acute} \end{cases}$

Proof. Consider triangle $t(ABC)$ with longest edge AB, and angles $\gamma = \angle ACB$ and $\alpha = \angle ABC$ respectively the largest and smallest angles of t. Let r be the circumradius of t and point D the midpoint of longest edge AB.

From Fact 5.2 we know that $|AB| = 2r\sin\gamma$, then it follows that $d(D,A) = d(D,B) = r\sin\gamma$, which has the smallest possible value of $r\sqrt{3}/2$ either when $\gamma = 120°$ or $\gamma = 60°$.

The distance to point C will depend on the geometry of the triangle. When t is an obtuse triangle, $d(D,C)$ is lower bounded by $r(1+\cos\gamma)$, the distance from D to the closest point in the circumcircle. In this case $d(D,C)$ has the smallest possible value of $r/2$ when $\gamma = 120°$. On the other hand, when t is an acute triangle vertex C lies farther away from D than from the circumcenter, then $d(D,C)$ has the smallest possible value of approaching r either when $\gamma \doteq 90°$ or $\alpha \doteq 0°$.

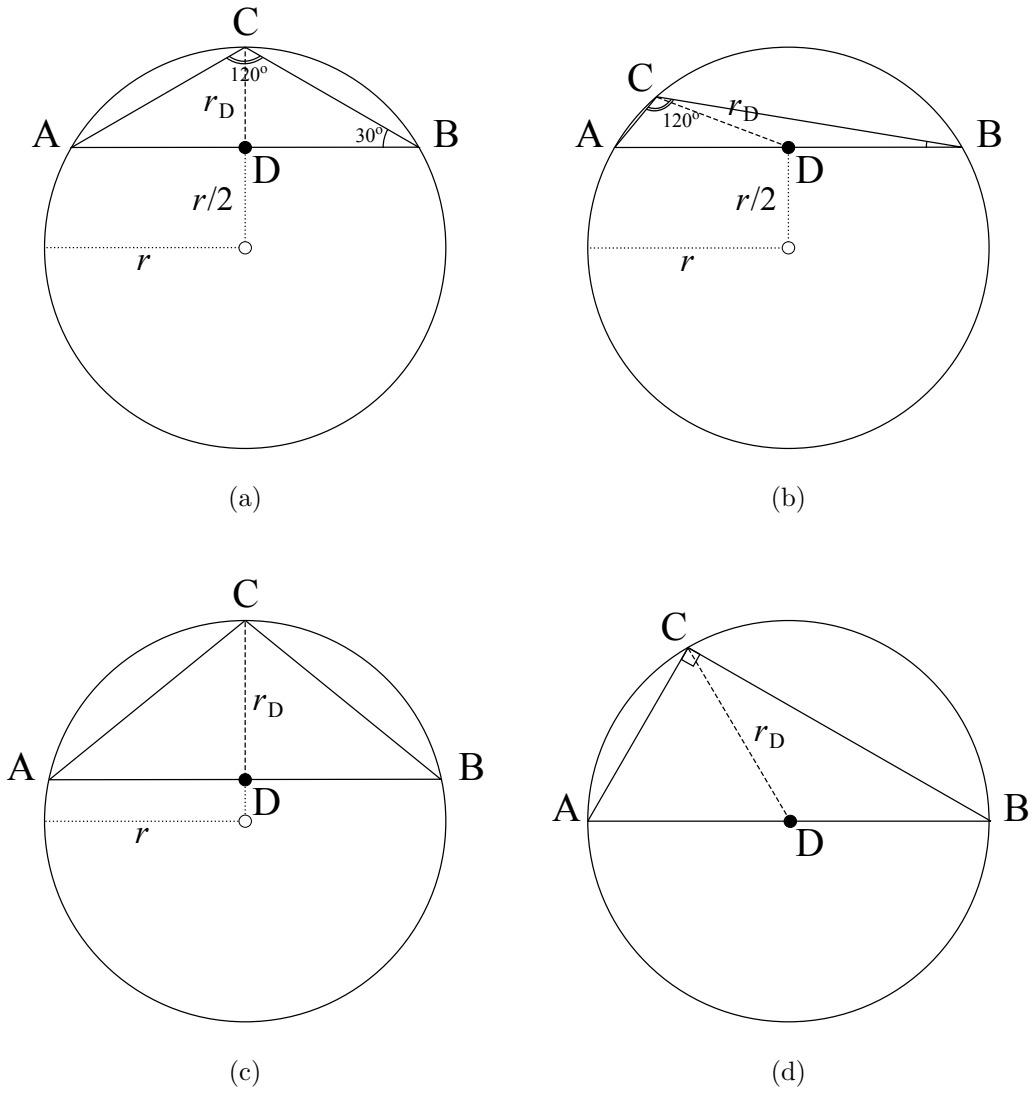
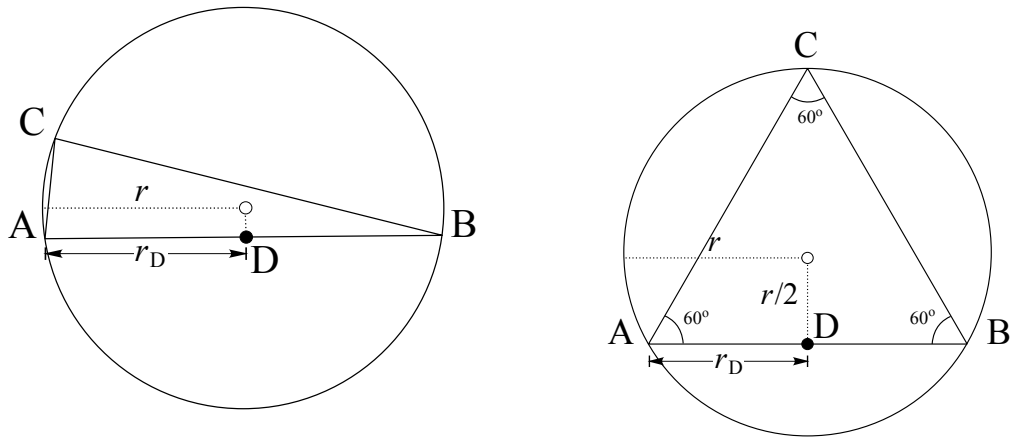


Figure 5.5: Examples of r_D for obtuse Delaunay terminal triangle $t(ABC)$ of circumradius r and D the midpoint of its longest edge. r_D is determined by the vertex of t closest to D. (a) Obtuse triangle with $\gamma = 120°$ and $\alpha = 30°$. (b) Small-angled obtuse triangle with $\gamma = 120°$ and $\alpha = 9°$. (c) Obtuse triangle with $\gamma = 102°$ and $\alpha = 39°$. (d) Right triangle with $\alpha = 30°$.



(a) (b)

Figure 5.6: Examples of r_D for acute Delaunay terminal triangle $t(ABC)$ of circumradius r and D the midpoint of its longest edge. r_D is determined by the vertex of t closest to D. (a) Acute triangle with $\gamma = 82°$ and $\alpha = 15°$. (b) Equilateral triangle.

□

Next, we define a general bound based on Lemma 5.7.

Lemma 5.8. *For any Delaunay terminal triangle $t(ABC)$ with circumradius r, and D the midpoint of its longest edge AB, the insertion radius of D with respect to the vertices of t, $r_{t,D}$, is greater than or equal to $r/2$. Furthermore,*

$$r_{t,D} = \begin{cases} d(D,C) & \text{if } t \text{ is an obtuse triangle} \\ r & \text{if } t \text{ is a right triangle} \\ d(D,A) & \text{if } t \text{ is an acute triangle} \end{cases}$$

Proof. Consider angles γ and α the largest and smallest of t. From Lemma 5.7 we know that $d(D,A) = r\sin\gamma$, $d(D,C)$ can be bounded based on the geometry of t. Then, $r_{t,D} = \min\{d(D,A), d(D,C)\}$ is determined as follows:

- When t is an obtuse triangle $d(D,C)$ can be lower bounded by $r(1+\cos\gamma)$, the shortest distance from D to the circumcircle. Furthermore, for any obtuse triangle it holds that $\sin\gamma > 1 + \cos\gamma$ regardless of the other angles. Then, $r_{t,D} = d(D,C)$.
- When t is a right triangle, D corresponds to the circumcenter of t, and every vertex is at a distance r from D. Then $r_{t,D} = r$.
- When t is an acute triangle, then $d(D,C) \geq r$ (Lemma 5.8). Furthermore, for any acute triangle $\sin\gamma < 1$ regardless of the other angles. Then, $r_{t,D} = d(A,C)$.

Finally, from Lemma 5.7 we know that the triangle's vertices are at least at a distance of $r/2$ from D, so $r_{t,D} \geq r/2$. □

Lemma 5.8 proves that the distance from the midpoint of a terminal edge to the vertices of the terminal triangle can be lower bounded by the circumradius. The following theorem generalizes this result considering any point of the triangulation.

Theorem 5.9. *For any Delaunay terminal triangle t it holds that the insertion radius of point D, r_D, the midpoint of its longest edge, is at least $r/2$, where r is the circumradius of t. Furthermore,*

$$r_D \geq \begin{cases} r(1+\cos\gamma) & \text{if } t \text{ is an obtuse triangle} \\ r & \text{if } t \text{ is a right triangle} \\ r(1-\cos\gamma) & \text{if } t \text{ is an acute triangle} \end{cases}$$

where γ is the largest angle of t.

Proof. Consider terminal triangle $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$, and angles γ and α respectively the largest and smallest angles of t. Let r be the circumradius of t and point D the midpoint of longest edge AB.

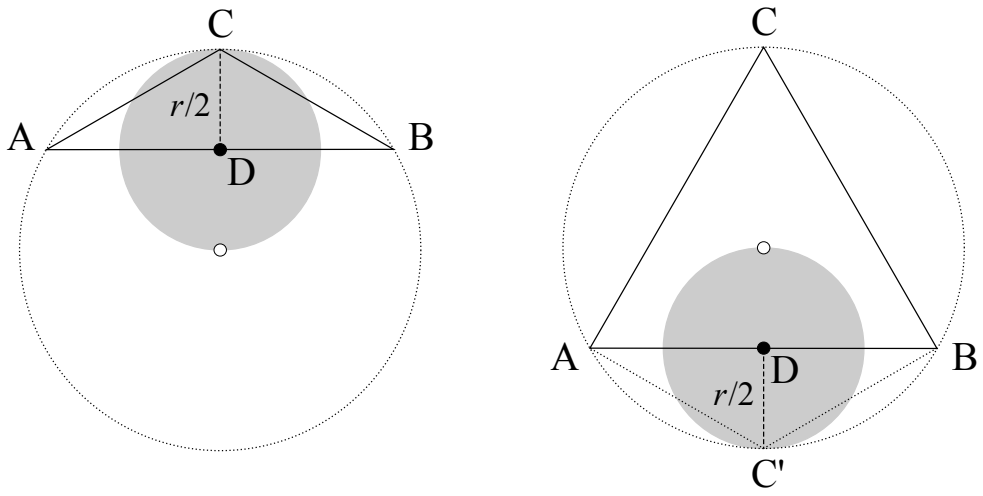
Because of the Delaunay property, t has an empty circumcircle. Then, the distance from the closest point of the triangulation to D can be lower bounded by the distance from D to the circumcircle:

- If t is an obtuse triangle, then $r_D > d_{\min}(D, C)$, where $d_{\min}(D, C) = r(1+\cos\gamma)$ is the shortest distance between D and vertex C (Lemma 5.7).
- If t is a rectangle triangle, then $r_D = r$ since D corresponds the circumcenter of t (Lemma 5.7).
- If t is an acute triangle, r_D is equivalent to the insertion radius of an obtuse terminal triangle sharing the circumcircle of t. Let $t'(AC'B)$ be this triangle, D is the midpoint of its longest edge AB, and $\gamma' = 180° - \gamma$ is its largest angle; see Figure 5.7(b). Then $r_D > d_{\min}(D, C')$, where $d_{\min}(D, C') = r(1+\cos\gamma') = r(1-\cos\gamma)$.

Consider the pessimistic case where D is located as close as possible to the circumcircle; see Figure 5.7. Triangle t corresponds either to the obtuse triangle with $\gamma = 120°$ and $\alpha = 30°$ (Figure 5.7(a)), or to the equilateral triangle (Figure 5.7(b)). In both scenarios $r(1+\cos\gamma)$ has the smallest possible value, $r/2$.

Let δ_D be the circle of radius r_D centered at point D. Then, no other point of the triangulation could lie inside δ_D because δ_D is contained by the empty circumcircle of t.

□



(a) (b)

Figure 5.7: Lower bounds for the insertion radius of point D, midpoint of longest edge AB of a triangle $t(ABC)$. The shaded ball of radius $r/2$ is fully contained in the circumcircle of t. (a) largest angle $\gamma = 120°$ and smallest angle $\alpha = 30°$. (b) $\alpha = \gamma = 60°$.

5.3.2 Bounds Based on the Shortest Edge

The insertion radius of the midpoint D of a terminal edge can be also lower bounded by the length of the shortest edge of the terminal triangle. In this section we define the relationship between the edges created after the introduction of D and the edges in the terminal triangles.

Lemma 5.10. *For any Delaunay terminal triangle $t(ABC)$ with shortest edge CA of length l and D the midpoint of its longest edge AB, the distance from D to the closest vertex of t is greater than or equal to $l/2$.*

Proof. Consider terminal triangle $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$, $l = |CA|$, γ and α respectively the largest and smallest angles of t, and point D the midpoint of terminal edge AB.

The proof is based on the diagram for similarity regions of triangles discussed in Section 4.2; see Figure 5.8. Candidate Delaunay terminal triangle t has its vertex C above arc C_5, corresponding to triangles with $\gamma \geq 120°$. We consider triangles such that $d(D, C) \leq |CA|$ or $d(D, A) \leq |CA|$. Consider $c_1 > 0$ the ratio between l and the distance from D to the closest vertex of t. The possible values of c_1 are discussed below.

- $d(D, C) \leq |CA|$: Corresponds to triangles with vertex C lying to the right of line OM of Figure 5.8. Constant c_1 decreases as C moves away from the intersection of arc C_3 and line PD. At the same time, constant c_1 decreases as C approaches line OM, acquiring the lowest value $c_1 = 1$ when C actually lies over OM. In this case $d(D, C) = |CA|$ regardless of the triangles geometry. On the other hand, c_1 increases as vertex C approaches PD, acquiring the highest values for isosceles triangles where C lies over PD. Furthermore:

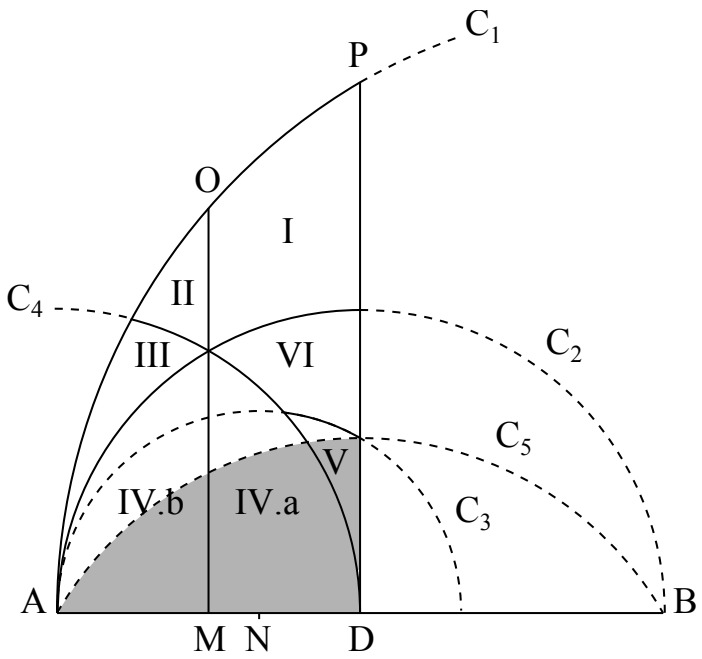


Figure 5.8: Diagram for triangle $t(ABC)$. Vertex C lies in one of the regions defining a triangle t with $|AB| \geq |BC| \geq |CA|$. The shaded region corresponds to triangles with $\gamma > 120°$.

  - If t is an acute triangle (region above arc C_2), then c_1 is maximized when t corresponds to the isosceles triangle with $\gamma \doteq 90°$. Therefore $c_1 < \sqrt{2}$.
  - If t is obtuse (region below arc C_2), then c_1 is maximized when t corresponds to the isosceles triangle with $\gamma = 120°$, and $d(D, C) = |CA|/2$. Therefore $c_1 \leq 2$.

- $d(D, A) \leq |CA|$: Corresponds to triangles with vertex C lying above arc C_4 of Figure 5.8. Constant decreases as C approaches arc C_4, acquiring the lowest value $c_1 = 1$ when C actually lies over C_4. In this case $d(D, A) = |CA|$ regardless of the triangle geometry. On the other hand, c_1 increases as vertex C moves away from arc C_4:

  - If t is an acute triangle, then the equilateral triangle maximizes the value of c_1, and $d(D, A) = |CA|/2$. Therefore $c_1 \leq 2$.
  - If t is an obtuse triangle, then the isosceles triangle with $\gamma \doteq 90°$ maximizes the value of c_1, and $d(D, A) = |CA|/\sqrt{2})$. Therefore, $c_1 \leq \sqrt{2}$.

Since $c_1 \leq 2$, it follows that the distance from D to the closest vertex of t is at least $l/2$.

□

Lemma 5.10 proofs that after introducing the midpoint of a terminal edge the length of the edges adjoining the new point to the vertices of the terminal triangle can be lower bounded using the length of the shortest edge. The following theorem generalizes this result to any edge in the triangulation adjoining the new point.

Theorem 5.11. *For any Delaunay terminal triangle t with shortest edge of length l, it holds that the insertion radius of the midpoint of its longest edge is at least $l/2$.*

Proof. The proof follows the same logic of the proof for Theorem 5.9. Consider the terminal triangle $t(ABC)$ with circumradius r and point D the midpoint of the longest edge AB. Consider the pessimistic case where D is located as close as possible to vertex C ($\gamma = 120°$ and $\alpha = 30°$ are the largest and smallest angles respectively). Then, $l = 2r \sin 30° = r$, $d(D, C) = 1 + \cos 120° = r/2$, and $r_D = l/2$. Due to the Delaunay property, any other point of the triangulation must lie outside the circumcircle of t. Then $r_D > l/2$.

□

5.3.3 Bounds for Constrained Midsize Edges

When the algorithm selects the midpoint of the constrained midsize edge of a terminal triangle, the insertion radius of the new point will be defined by the length of the midsize edge. Among the vertices of the terminal triangle, the endpoints of the midsize edge become the closest points to the point selected.

The following lemma defines the bounds on the insertion radius using the circumradius of the terminal triangle and using the smallest edge of t.

Lemma 5.12. *For any Delaunay terminal triangle $t(ABC)$ with shortest edge CA of length l, and E the midpoint of its constrained midsize edge BC, the distance from of E to the closest vertex of t is greater than or equal to $l/2$, or at least $r/2$, where r is the circumradius of t. Furthermore, this distance is $|BC|/2 = r \sin \beta$, where β is the angle opposite edge BC.*

Proof. Consider triangle $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$; angles γ, β and α respectively the largest, midsize and smallest angles of t; and r the circumradius of t. Let point D be the midpoint of longest edge AB and point E be the midpoint of constrained edge BC. Since E is the midpoint of edge BC, the vertex of t closest to E is always an endpoint of BC, that is $|BC|/2$. By Fact 5.2 $|BC| = 2r \sin \beta$, and it follows that $d(E, B) = r \sin \beta$.

We observe that $d(E, B)$ is minimized for the obtuse triangle with $\gamma \doteq 120°$ and $\alpha \doteq 30°$ (Figure 5.9). The insertion of point D would produce a triangle with smallest angle of 30°. Assume that the algorithm rejects D and selects point E instead. In this scenario it holds that $|BC| = |CA| = r$, therefore $d(E, B) = |CA|/2 = r/2$. It follows that the distance from E to any vertex of t is lower bounded by $|CA|/2$ (in terms of the smallest edge), or by $r/2$ (in terms of the circumradius).

Finally, distance $d(E, A)$ decreases as the smallest angle of the terminal triangle approaches 0°. In this boundary case $d(E, A) \doteq \frac{\sqrt{3}}{2} r$, since edges AB and BC tend to have the same length. On the other hand, $d(E, A)$ is maximized for the equilateral triangle, where $d(E, A) \doteq \frac{3}{2} r$.

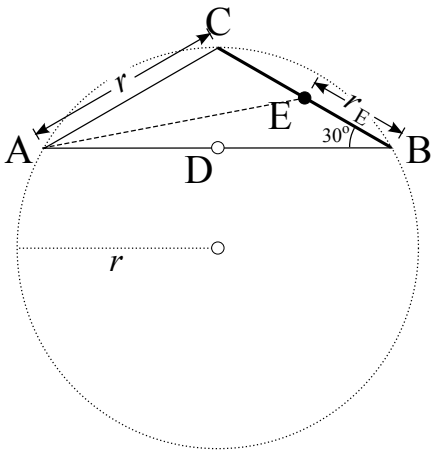


Figure 5.9: Delaunay terminal triangle $t(ABC)$ with angles $\gamma = 120°$ and $\alpha = 30°$. Point E is the midpoint of its constrained midsize edge.

□

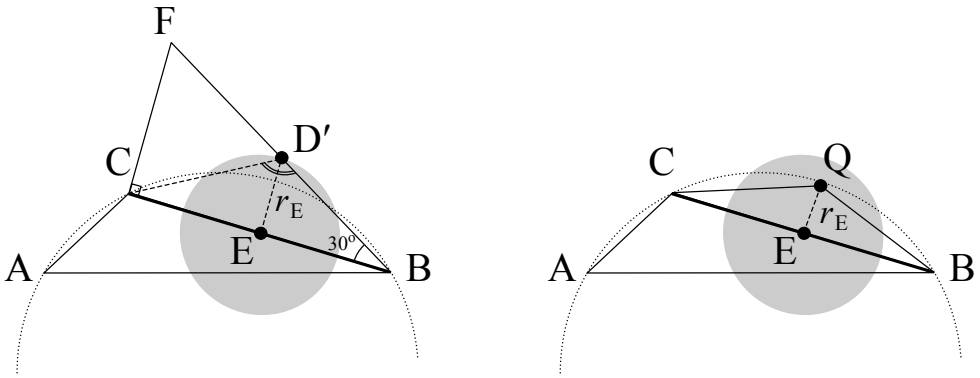
The relationship between the constrained midsize edge and the insertion radius of its midpoint is presented in the following lemma.

Lemma 5.13. *Let $t(ABC)$ be a Delaunay terminal triangle with constrained midsize edge BC. Let E be the midpoint of this edge. Then, the insertion radius of E, r_E, is greater than $d(E,B)/\sqrt{3} = |BC|/2\sqrt{3}$.*

Proof. Consider that no point from the input lies too close to constrained edge BC. If at some point before the insertion of E the midsize-edge neighbor of t was refined, say t', the algorithm inserted the midpoint of the longest edge of t', say D'. As observed in Figure 5.10(a), due to the requirement of 30° to insert the midpoint of longest edge instead of the midpoint of the constrained midsize edge, the smallest triangle that the algorithm could produce is the obtuse triangle with largest angle of 120° and smallest angle of 30°. After a few calculations, distance $d(E,D')$ is equal to $\frac{|BC|}{2\sqrt{3}}$ which in turn is equal to $\frac{d(E,B)}{\sqrt{3}}$, defining a lower bound on the insertion radius of E. □

It must be noted that, if an input point Q lies within distance $\frac{d(E,B)}{\sqrt{3}}$ from point E (Figure 5.10(b)), or point Q is the midpoint of a constrained edge $e \neq BC$ previously inserted by the algorithm (and Q lies within distance $\frac{d(E,B)}{\sqrt{3}}$ from E), then Q is the point of the triangulation closest to E. In both scenarios $r_E = d(E,Q)$, as stated by the following lemma.

Lemma 5.14. *Let $t(ABC)$ be a Delaunay terminal triangle and E be the midpoint of its constrained midsize edge BC. If there is a point Q within distance $d(E,B)/\sqrt{3}$ from E, then Q is either an input point or Q is the midpoint of another constrained edge. The insertion radius of point E is $d(E,Q)$.*



(a) (b)

Figure 5.10: Lower bounds for the insertion radius of point E, midpoint of constrained midsize edge BC of a triangle $t(ABC)$. The shaded ball has a radius of $\frac{d(E,B)}{\sqrt{3}}$. (a) After insertion of point D' triangle CBD' largest angle $\angle BD'C = 120°$ and smallest angle $\angle CBD' = 30°$. For the general case $r_E > \frac{d(E,B)}{\sqrt{3}}$. (b) Input point Q is the closest point to the midpoint of constrained edge BC, therefore $r_E = d(E, Q)$.

□

The following theorem synthesizes the bounds on the insertion radius presented in Lemma 5.12 and Lemma 5.13. We define a bound on the insertion radius of the midpoint of a constrained midsize edge of a terminal triangle in terms of the circumradius of the triangle and in terms of the smallest edge of the triangle.

Theorem 5.15. *For any Delaunay terminal triangle t the insertion radius of the midpoint of its constrained midsize edge is at least $r/2\sqrt{3}$, where r is the circumradius of t; or at least $l/2\sqrt{3}$, where l is the length of the shortest edge of t.*

Proof. The bound is based on the worst-case scenario where t is the obtuse triangle with largest angle $\gamma = 120°$ and smallest angle $\alpha = 30°$, as illustrated in Figure 5.11.

Consider triangle $t(ABC)$ with constrained midsize edge BC, shortest edge CA and point E the midpoint of BC. From Lemma 5.13 we know that $r_E > |BC|/2\sqrt{3}$, which applies to the general case of refinement of terminal triangles. From Lemma 5.12 we know that the $|BC|$ is lower bounded either by $r/2$ or by $|CA|/2$. It follows that $r_E > \frac{r}{2\sqrt{3}}$, or that $r_E > \frac{l}{2\sqrt{3}}$. □

Theorem 5.15 holds when there are no input points or points over a constrained edge lying within the bound stated above, otherwise the insertion radius is defined by the distance to this point (Lemma 5.14).

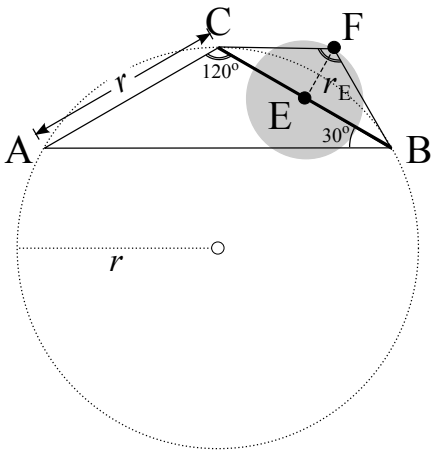


Figure 5.11: Lower bounds for the insertion radius of point E, midpoint of constrained midsize edge BC of a terminal triangle $t(ABC)$ with largest angle $\gamma = 120°$ and smallest angle $\alpha = 30°$. Midsize edge neighbor triangle, CBF, is similar to t. The shaded ball has a radius of $\frac{r}{2\sqrt{3}}$.

5.4 Insertion Radius and Parent Points

In this section we present a study on the relationship between the insertion radius of the point selected and its parent point, independent of the quality parameter and the minimum angle allowed.

When no constrained edges are involved in the refinement, the Lepp-Delaunay algorithm selects the midpoint of the terminal edge in the longest-edge propagating path of the target triangle being processed. We analyze two scenarios for terminal triangles: when the triangle is the terminal triangle in the propagating path of a target triangle (Section 5.4.1), and when the triangle is both terminal and the target triangle currently being processed by the algorithm (Section 5.4.2). For the latter, we analyze the case of obtuse and acute triangles, identifying the bounds on the insertion radius for each scenario.

Finally, we study the case of terminal triangles with constrained edges and the relationship between the insertion radius of the midpoint of a constrained midsize edge and the insertion radius of the midpoint of the terminal edge (Section 5.4.3).

5.4.1 Terminal Triangles in a Lepp

Lemma 5.16. *Let t be a Delaunay terminal triangle. Let D be the midpoint of the longest edge. Let $t' \neq t$ be the target triangle in S_{target} currently being processed by the Lepp-Delaunay algorithm. Then $r_D \geq k_1 r_{\hat{D}}$, for a constant k_1.*

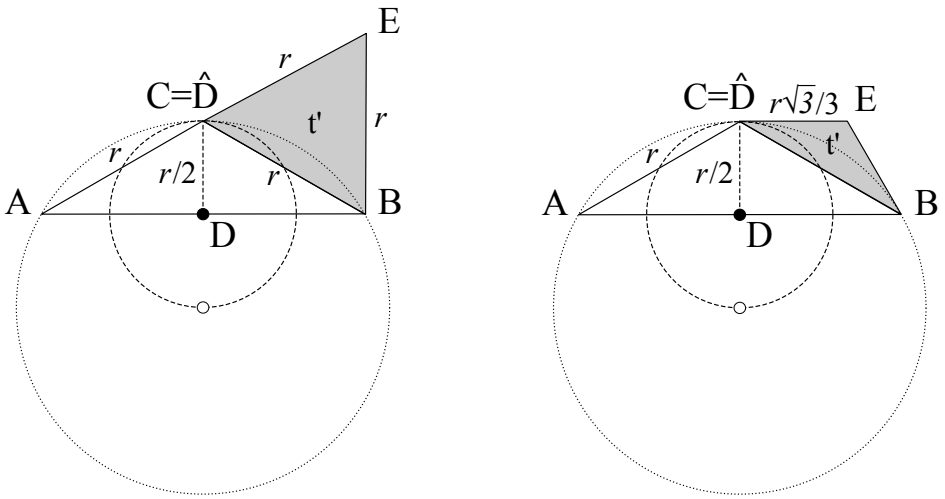


Figure 5.12: Terminal triangle $t(ABC)$ with smallest edge l_t and its parent $t'(CBE)$ with smallest edge $l_{t'}$. t is the terminal triangle in Lepp(t') and has smallest angle $\alpha_t = 30°$. (a) When $l_{t'} = l_t = r$ and $\alpha_{t'} = 60°$. (b) When $l_{t'} = r\sqrt{3}/3$ and $\alpha_{t'} = 30°$.

Proof. Consider terminal triangle $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$, point D the midpoint of AB and circumradius r. Consider parent triangle $t'(A'B'C')$ with $|A'B'| \geq |B'C'| \geq |C'A'|$, so that t is the terminal triangle in Lepp(t').

Since $\hat{D} = A'$, then $r_{\hat{D}} \leq l_{t'}$, the length of the smallest edge of t'. Due to the processing order, t' is the triangle in S_{target} with the smallest edge, and $l_{t'} < l_t$ (Lemma 5.6). By transitivity $r_{\hat{D}} < l_t$ (i).

As a conservative bound, we could consider edge CA as an upper bound for r_D, so that r_D could not be smaller than the insertion radius of a target terminal triangle (Section 5.4.2), and therefore $k_1 > 0.5$.

Furthermore, from Theorem 5.9 we know that the obtuse triangle with the least possible insertion radius $r_D = r/2$ corresponds to triangle t with largest angle $\gamma_t = 120°$. Let $\alpha_t = 30°$ be the smallest angle of t, which maximizes the length of the smallest edge of t, that is, $l_t = |CA| = 2r \sin 30° = r$ (ii). Then, from (i) and (ii) it holds that $r_{\hat{D}} \leq r$.

In fact, this is only true when t' is the equilateral triangle (Figure 5.12(a)), which would correspond to an angle tolerance parameter $\theta = 60°$. If we consider a quality parameter $\theta = 30°$, then edge $l_{t'}$ reaches its maximum length when $\alpha_{t'} \doteq 30°$; as seen in Figure 5.12(b).

After some geometric calculations $r_{\hat{D}} < \frac{r\sqrt{3}}{3} \approx 0.58r$. Since $r_D \geq r/2$, then $k_1 > \sin 120° \approx 0.87$.

□

5.4.2 Target Terminal Triangles

When the current triangle selected for refinement is also a Delaunay terminal triangle the insertion radius of the midpoint D introduced by the algorithm will depend on various factors. The following lemmas describe the scenarios for r_D when the target triangle is an obtuse triangle (Lemma 5.19) or an acute triangle (Lemma 5.20).

We start the analysis describing the boundary scenarios of obtuse triangles in the following propositions.

Proposition 5.17. *Let t be a Delaunay terminal triangle with largest angle $\gamma = 120°$. Let D be the midpoint of the longest edge. Let t' be the current target triangle being processed by the Lepp-Delaunay algorithm. If $t = t'$ then $r_D \geq k_2 r_{\hat{D}}$, for a constant k_2 that depends on the smallest angle of t.*

Proof. Consider the terminal triangle $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$ and point D the midpoint of AB. Since $\gamma = 120°$, smallest angle $\alpha \leq 30°$. Triangle t itself is the target triangle responsible for the insertion of D, so $\hat{D} = A$ and $r_{\hat{D}} \leq |CA| \leq r$, where r is the circumradius of t. Furthermore, CA is the smallest edge of the triangles in S_{target}, and smallest angle α is smaller than θ, for an angle refinement parameter θ.

Then $r_D = r(1 + \cos\gamma) = r/2$ (Theorem 5.9), and constant k_2 is determined by the formula

$$k_2 = \frac{r/2}{2r\sin\alpha} = \frac{1}{4\sin\alpha}. \tag{5.2}$$

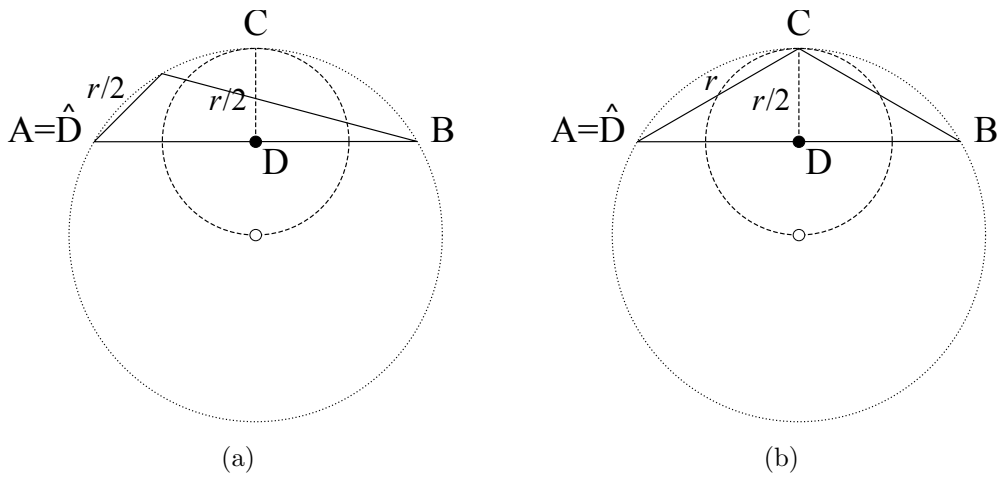
Consider the following scenarios:

- $k_2 \geq 1$ when $\geq |CA| = r/2$ and $r_D \geq r_{\hat{D}}$. This is true for any triangle with $\alpha \leq \arcsin(0.5) \approx 14.48°$; see Figure 5.13(a).
- Otherwise, $k_2 < 1$ and $|CA| > r/2$. Furthermore, $k_2 \geq 0.5$ since edge AC is maximized for the isosceles triangle with $\alpha = 30°$; see Figure 5.13(b).

□

Proposition 5.18. *Let t be a Delaunay terminal triangle with largest angle $\gamma = 90°$. Let D be the midpoint of the longest edge. Let t' be the current target triangle being processed by the Lepp-Delaunay algorithm. If $t = t'$ then $r_D \geq k_3 r_{\hat{D}}$, for a constant k_3 that depends on the smallest angle of t.*

Proof. Again, consider the terminal triangle $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$ and point D the midpoint of AB. Since $\gamma = 90°$, then smallest angle $\alpha \leq 45°$. Triangle t is the target triangle responsible for the insertion of D, so $\hat{D} = A$ and $r_{\hat{D}} \leq |CA| \leq \sqrt{2}r \approx 1.41r$.



(a) (b)

Figure 5.13: Terminal triangle $t(ABC)$ with $\gamma = 120°$. A is the parent point of D. (a) $\alpha \approx 14.48°$: $DC = CA$. (b) $\alpha = 30°$: $DC = CA/2$.

In this configuration point D is the circumcenter of t and $r_D = r$ regardless of α (Theorem 5.9). Constant k_3 is given by the formula

$$k_3 = \frac{1}{2\sin\alpha}. \tag{5.3}$$

We consider the following scenarios:

- $k_3 \geq 1$, when $r_D \geq r_{\hat{D}}$ and $|CA| \leq 2\sin\alpha \leq r$. This is true for any triangle with $\alpha \leq \arcsin 0.5 = 30°$; see Figure 5.14(a).
- Otherwise, $\alpha > 30°$, $r < |CA| \leq \sqrt{2}r$, and $k_3 < 1$. Furthermore, $k_3 \geq 1/\sqrt{2} \approx 0.7$ since edge AC is maximized for the isosceles triangle with $\alpha = 45°$; see Figure 5.14(b).

Finally, for any angle tolerance parameter $\theta < 30°$, then $k_3 > 1$. Furthermore, CA is the smallest edge of the triangles in S_{target}, and smallest angle α is smaller than θ.

□

Lemma 5.19. *Let t be an obtuse Delaunay terminal triangle. Let D be the midpoint of the longest edge. Let t' be the current target triangle in S_{target} being processed by the Lepp-Delaunay algorithm. If $t = t'$ then $r_D \geq k_4 r_{\hat{D}}$, for a constant k_4 that depends on the smallest and largest angles of t.*

Proof. Consider $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$ and point D the midpoint of AB. Let γ and α respectively be the largest and the smallest angles of t, and r the circumradius. Largest angle γ is bounded by $90° < \gamma \leq 120$. Furthermore:

- $r_D = r(1 + cos\gamma)$ is determined by the largest angle (Theorem 5.9).

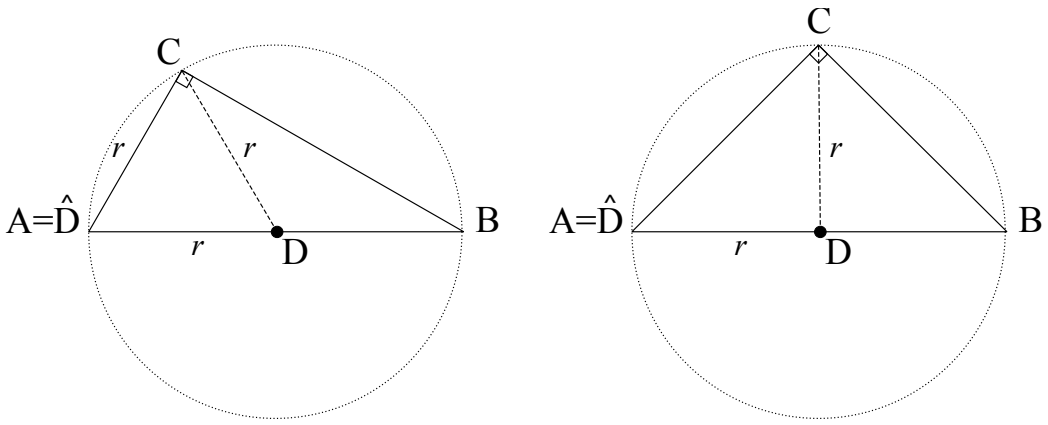


Figure 5.14: Terminal triangle $t(ABC)$ with $\gamma = 90°$. A is the parent point of D. (a) $\alpha = 30°$: $DC = CA = AD$. (b) $\alpha = 45°$: $DC = CA/\sqrt{2}$.

- Since $\hat{D} = A$, then $r_{\hat{D}} = 2r \sin \alpha$ is determined by the smallest angle of the triangle (Fact 5.2).

Then, the value of k_4 is given by the formula

$$k_4 = \frac{1 + \cos \gamma}{2 \sin \alpha}, \tag{5.4}$$

which depends on the geometry of t. Consider the following scenarios:

- For a fixed angle γ, consider $\alpha_{\max}$ the angle α satisfying Equation 5.4. Then, any obtuse triangle with $\alpha \leq \alpha_{\max}$ will have $k_4 \geq 1$.
- On the other hand, for a fixed angle α, and $\gamma_{\max}$ the angle γ satisfying Equation 5.4, any obtuse triangle with $\gamma \leq \gamma_{\max}$ will have $k_4 \geq 1$.

Finally, the worst scenario is defined in Proposition 5.17, with the most obtuse triangle $\gamma = 120$. In this scenario angle $\alpha_{\max} = 14.48$ represents values of $k_4 \geq 1$. It must be noted that the less obtuse triangle t is, the higher angle $\alpha_{\max}$. An obtuse triangle with $\gamma \doteq 90°$ will have constant $k_4 \geq 1$ for any smallest angle $\alpha \doteq 30°$ (Proposition 5.18). □

Lemma 5.20. *Let t be an acute Delaunay terminal triangle. Let D be the midpoint of the longest edge. Let t' be the current target triangle in S_{target} being processed by the Lepp-Delaunay algorithm. If $t = t'$, then $r_D \geq k_5 r_{\hat{D}}$, for a constant k_5 that depends on the smallest and largest angles of t.*

Proof. Consider $t(ABC)$ with $|AB| \geq |BC| \geq |CA|$ and point D the midpoint of AB. Let γ and α be respectively the largest and the smallest angles of t, and r the circumradius. Furthermore:

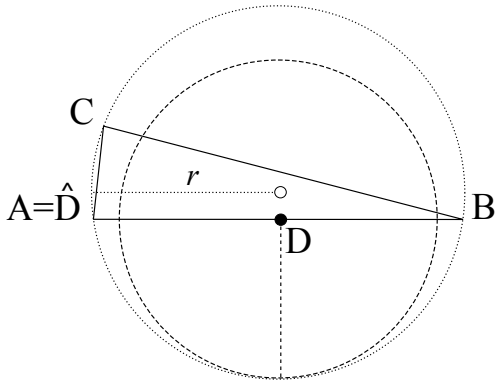


Figure 5.15: Insertion radius for terminal triangle $t(ABC)$. t is acute and A is the parent point of D. $\gamma = 14.48°$ and $|CA| = r/2$.

- $r_D = r(1 - cos\gamma)$ is determined by the largest angle (Theorem 5.9).
- $\hat{D} = A$ and $r_{\hat{D}} = 2r \sin\alpha$ is determined by the smallest angle (Fact 5.2).

The value of k_5 is given by the formula

$$k_5 = \frac{1 - \cos\gamma}{2\sin\alpha}, \tag{5.5}$$

which depends on the geometry of t.

Since t is the target triangle in S_{target} currently being processed, then $\alpha < \theta$, where θ is the angle tolerance parameter. This implies that angle γ is bounded by $\gamma_{\min} = \frac{180-\theta}{2}$, and $r_D > r(1 - \cos\gamma_{\min})$.

For any angle tolerance parameter $\theta < 30°$, then $k_5 > \frac{1-\cos 75°}{2\sin 30°} \approx 0.74$. Furthermore, for any $\theta < 23.47°$ it holds that $k_5 > 1$.

In the scenarios in which t is the sole terminal triangle (i.e. it does not have a longest-edge neighbor), or the neighbor terminal triangle is acute, vertex A is the point of the triangulation that lies the closest to D and $r_D \leq d(D, A)$, where $d(D, A) = r\sin\gamma$. For any angle tolerance parameter $\theta < 28.95°$, it holds that $r\sin\gamma_{\min} > 2r\sin\alpha$, and constant $k_5 > 1$.

□

5.4.3 Constrained Midsize Edges

When the terminal triangle has its midsize edge constrained, the algorithm could select the midpoint of the constrained edge instead of the midpoint of the terminal edge. In these

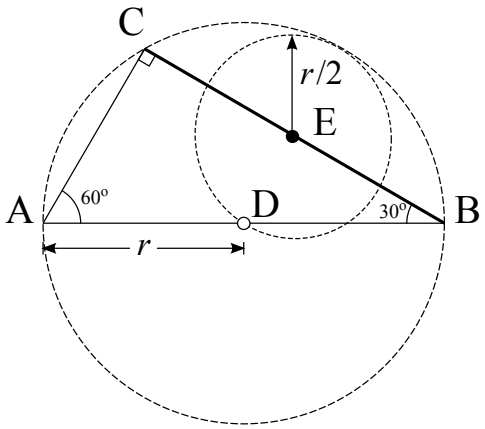


Figure 5.16: Insertion radius for terminal triangle $t(ABC)$ with constrained midsize edge. t is the right triangle with smallest angle $\alpha = 30°$. Point D, the midpoint of longest edge AB, is the parent point of E, the midpoint of midsize edge. Circles of radius r and $r/2$ represent the insertion radius of D and E respectively.

scenarios, the midpoint of the longest edge is discarded and becomes the parent point of the selected point. The following lemma establishes the relationship between the insertion radius of the midpoint of a constrained edge and the insertion radius of its parent.

Lemma 5.21. *Let t be a Delaunay terminal triangle with constrained midsize edge. Let E be the midpoint of this edge. Then $r_D > r_{\hat{D}}/2$.*

Proof. Consider the terminal triangle $t(ABC)$ with largest angle γ, midsize angle β and smallest angle α. Let edge BC be the constrained midsize edge, and edge AB be the longest (terminal) edge. Let point E be the midpoint of edge BC and point D be the midpoint of edge AB. Then, $\hat{E} = D$.

By Theorem 5.9, $r_{\hat{E}}$ is at least $r(1+\cos\gamma)$ if t is an obtuse triangle, or at least $r(1-\cos\gamma)$ if t is acute. $r_{\hat{E}}$ is maximized when t is a right triangle. By Lemma 5.13, r_E is at least $|BC|/2\sqrt{3} = r\sin\beta/\sqrt{3}$.

The algorithm selects point E only if the longest-edge bisection of t (i.e. the insertion of point $\hat{E}$) produces a triangle with an angle smaller than 30°. Because of this angle verification, the boundary case arises when t is the right triangle and $\alpha = 30°$. Figure 5.16 illustrates this scenario.

Then, $r_{\hat{E}} = r$ and $r_E = r\sin 60°/\sqrt{3} = r/2$, so the insertion radius of $\hat{E}$ is twice the insertion radius of E. The same bound certainly holds for any terminal triangle with $\alpha < 30°$. Furthermore, for any terminal triangle such that its longest-edge bisection produces a triangle with an angle smaller than 30° it holds that $\frac{r_{\hat{E}}}{r_E} < 2$. □

5.5 Processing Terminal Edges

The basic point selection strategy used by the Lepp-Delaunay algorithm selects the midpoint of the terminal edge for insertion. This point is later inserted into the triangulation, performing the necessary edge-flip operations to maintain the constrained Delaunay property.

If no edge-flip operation takes place, the Delaunay-insertion of the selected point corresponds to the longest-edge bisection of the terminal triangles. As discussed in Section 4.7.2, the longest-edge bisection operation tends to create quasi-equilateral triangles, while also increasing the area covered by good-quality triangles (Lemma 4.15).

On the other hand, if edge-flip operations are performed, this will result in the elimination of small-angled and too-obtuse triangles, while maximizing the smallest angles of the affected triangles due to the Delaunay property.

As discussed in Section 5.4, the insertion of a terminal-edge midpoint could create an edge smaller than the shortest edge, l, of the terminal triangle. A naive analysis could assume that small edges are associated with triangles of poor quality, and therefore an algorithm creating smaller edges than those currently existing in the triangulation could threaten its termination.

In what follows we show that the insertion of the longest-edge midpoint of a triangle improves the quality of the triangles affected. We analyze various scenarios for the angle tolerance parameter θ and define the geometrical properties of the triangles obtained after refinement.

5.5.1 An Initial Bound on the Smallest Angles

Working with angle tolerance parameter $\theta = 17.17°$, new edges created after the longest-edge bisection of a terminal triangle t will not be smaller than triangle's shortest edge l_t. For some specific scenarios, an edge created during the edge-flip operation affecting one of the descendants of t could create an new edge slightly smaller than l_t. In what follows we show that this new edge will be associated with good-quality triangles.

Consider the obtuse terminal triangle $t(ABC)$ with largest angle $\gamma = 120°$ and smallest angle $\alpha = (60° - \arcsin(\sqrt{3}/4))/2 \approx 17.17°$ so that edge $DC = CA$; as shown in Figure 5.17.

Consider triangle $t_1(ADC)$, the left descendant of t. Since t is obtuse, by longest-edge bisection property it holds that $\alpha_1 \geq 2\alpha$ (Theorem 2.7), where α_1 is the smallest angle of t_1. After some geometric calculations,

$$\alpha_1 = \arccos\left(\frac{\frac{1}{2}\sin 120°}{2\sin 17.17°}\right) \approx 42.83°.$$

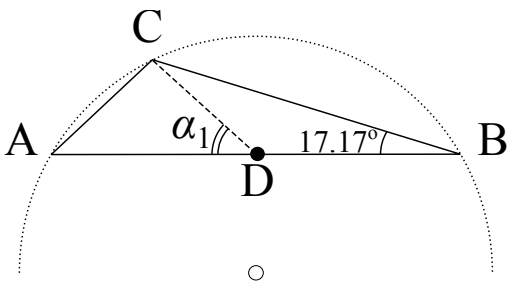


Figure 5.17: Terminal triangle $t(ABC)$ with $\gamma = 120°$, $\alpha = 17.17°$ and $DC = CA$. Triangle $t_1(ADC)$ has smallest angle $\alpha_1 = 42.83°$. Triangle $t_2(BCD)$ is too-obtuse.

Triangle t_1 will not belong to set S_{target} and could be refined only if it becomes the terminal triangle in the Lepp of a target triangle $\hat{t}$. Next, we prove that even if t_1 is processed by the algorithm, no new edge will be smaller than the shortest edge of parent triangle $\hat{t}$.

Consider that triangle t_1 became terminal and point D_1, the longest-edge midpoint of t_1, was introduced creating edges AD_1, D_1D and D_1C ($AD_1 = D_1D$ and $D_1C < AD_1$), as shown in Figure 5.18(a). Although these edges are smaller than the shortest edge, CA, due to the longest-edge propagation property they cannot be smaller than the shortest edge of target triangle $\hat{t}$. Let $t^*(ACE)$ be the triangle preceding t in Lepp($\hat{t}$). Region R defines the geometrical region for vertex E such that edges EA and CE are not smaller than D_1C. Such a triangle has angles greater than α_1, and $\hat{t}$ will necessarily have an edge smaller than D_1C. Moreover, since we consider $\theta = 17.17°$, then $\hat{t}$ can only exist when t^* has an edge smaller than D_1C.

If we consider that triangle t_1 was eliminated right after the introduction of D by an edge-flip operation, then triangle t^* had vertex E lying inside the circumcircle of t_1 (the shaded area in Figure 5.18(b)). Furthermore, triangle t^* will have a shortest edge smaller than $\approx 0.54|DC|$ and a smallest angle $\alpha_{t^*} < \alpha_1/2 \approx 21.42°$. An edge-flip operation affecting these triangles will replace edge CA for a longer edge DE, while also improving the angle distribution of the affected triangles.

This same analysis is valid for any triangle t with smallest angle $\alpha < 17.17°$, and the Lepp-Delaunay algorithm does not create edges smaller than those in the current triangulation when processing t_1.

Next we analyze triangle $t_2(BCD)$, the right descendant of t. Since angle α is smaller than 22.24°, then t_2 is a too-obtuse triangle with largest angle $\gamma_2 > 120°$ and will necessarily be removed by an edge-flip operation (Section 3.3.5). Figure 5.19 illustrates this scenario.

The Lepp-Delaunay could then create an edge smaller than CA if the midsize-edge neighbor triangle CBF has vertex F lying on the region between the circumcircle of t and the circle δ_D of radius CA centered at D; see Figure 5.20(a). Since we assume that vertex C is not adjacent to any edge smaller than CA, point F cannot lie inside circle δ_C of radius CA centered at C. Figure 5.20(a) illustrates the region defining candidate longest-edge neighbor meeting these requirements.

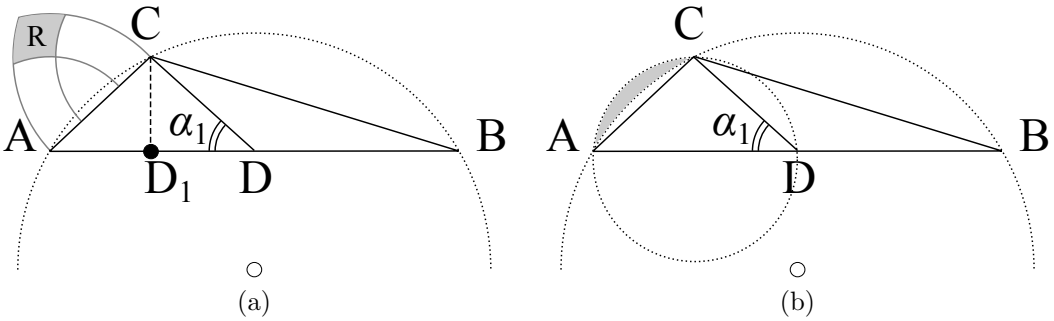


Figure 5.18: (a) Region R defines the triangle preceding t in the Lepp such that none of its edges are smaller than D_1C. (b) Shaded region for shortest-edge neighbor that triggers the edge-flip operation which removes triangle t_1.

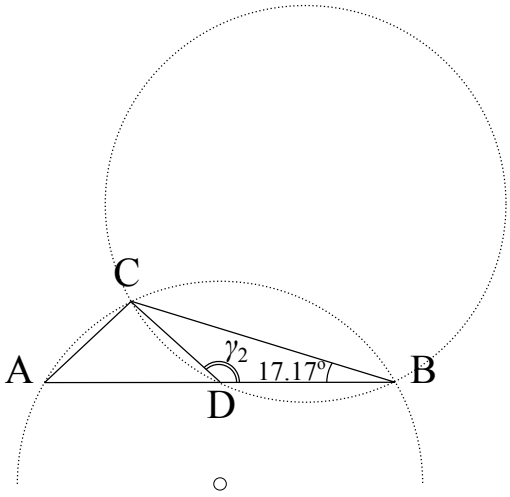


Figure 5.19: Obtuse triangle $t_2(BCD)$ with largest angle $\gamma_2 = 137.17°$. t_2 is eliminated by edge-flip operations whenever the algorithm inserts a point inside its circumcircle.

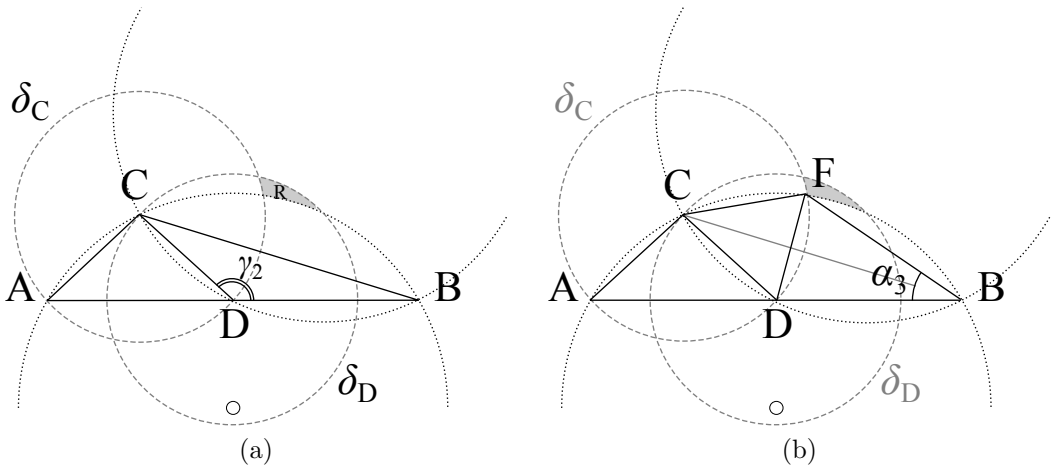


Figure 5.20: (a) Region R defines triangle t_3, the longest edge neighbor of triangle $t_2(BCD)$, such that no edge of t_3 is smaller than DC. (b) Triangle $t_3(CBF)$ has vertex F lying as close as possible to D, and is removed by an edge-flip operation that produces two quasi-equilateral triangles. Angle $\alpha_3 = 34.34°$.

If such a triangle exists, then the algorithm creates the new edge DF of length at least $0.86|CA|$, producing two quasi-equilateral triangles with angles greater than 2α (Figure 5.20(b)).

These same observations can be extended to the case when triangle t is the current target triangle in S_{target} being processed.

The following lemma summarizes these results.

Lemma 5.22. *Let $\theta = 17.17°$ be the angle tolerance parameter such that target triangles in S_{target} have angles smaller than θ. Then, the Lepp-Delaunay algorithm produces an output triangulation with edges longer than l_{min}, the smallest edge of the input triangulation. Moreover, if a new small edge is created, it will be associated with triangles of improved quality.*

5.5.2 A Stricter Bound on the Smallest Angles

We also analyze a stricter bound on parameter θ such that it is guaranteed that new edges are longer than any existing edge in the input triangulation. Working with parameter $\theta = 14.48°$, for every configuration of triangles the edges created after the insertion of the terminal edge midpoint will be always longer than the terminal triangle's shortest edge, thus longer than l_{min}, the smallest edge of the target triangulation.

Consider the obtuse terminal triangle $t(ABC)$ with largest angle $\gamma = 120°$ and smallest angle $\alpha = \arcsin(1/4) \approx 14.48°$ such that edge $CA = r/2$, the minimum insertion radius of midpoint D for a triangle of circumradius r (also discussed in Section 5.4.2). In this scenario new edge $DC > CA$ and any edge adjoining D after edge-flip operations will be necessarily greater than CA as well; see Figure 5.21.

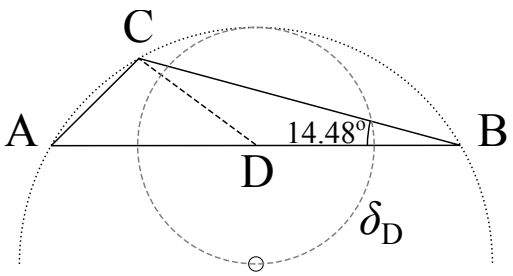


Figure 5.21: Terminal triangle $t(ABC)$ with $\gamma = 120°$, $\alpha = 14.48°$ and $CA = r_D$. Circle δ_D of radius CA centered at point D is empty of any point of the triangulation.

Lemma 5.23. *Let $\theta = 14.48°$ be the angle tolerance parameter such that target triangles in S_{target} have angles smaller than θ. Then, the Lepp-Delaunay algorithm produces an output triangulation with no edges smaller than $l_{\min}$, the smallest edge of the triangles in S_{target}.*

Proof. Consider circle δ_D of radius CA centered at point D, as shown in Figure 5.21. δ_D is fully contained in the circumcircle of t, and by Delaunay property no point of the triangulation lies inside δ_D. Either by terminal edge refinement or by edge-flip, the algorithm will not create edges adjoining point D that could be smaller than CA. □

5.5.3 A Relaxed Angle Bound on the Angle Tolerance Parameter

Finally, we discuss and analyze the algorithm when the angle tolerance parameter θ is relaxed 25.66°. In this scenario new edges created after the longest-edge bisection of a terminal triangle t could be smaller than triangle's shortest edge l_t. Similarly to the scenario discussed in Section 5.5.1 edges created during edge-flip operations affecting the descendants of t could create new small edges. We show that these new edges will be associated with good-quality triangles.

Although the angle parameters proposed in lemmas 5.22 and 5.23 ensure the termination of the algorithm with good-quality refined triangles, such low angle bounds could limit the improvement properties of the Lepp-Delaunay algorithm. Previous empirical studies showed that, in practice, the Lepp-Delaunay algorithm terminates for values of $\theta = 30°$ [37, 40].

Working with an angle bound $\theta = 25.66°$, it is possible to guarantee that the algorithm could only produce slightly smaller edges that are associated with good-quality triangles, without affecting the performance of the algorithm nor its termination (in practice, the algorithm terminates with angle bounds of up to 30°).

Consider the obtuse terminal triangle $t(ABC)$ with largest angle $\gamma = 120°$ and smallest angle $\alpha = \arcsin(\frac{\sin 120°}{2}) \approx 25.66°$ such that edge $CA = AD = DB$. Moreover, $|CA| = 2r \sin 25.66° = r\sqrt{3}/2 \approx 0.87r$, and $|DC| \approx 0.51r$ (from Equation 5.1); as shown in Figure 5.22(a).

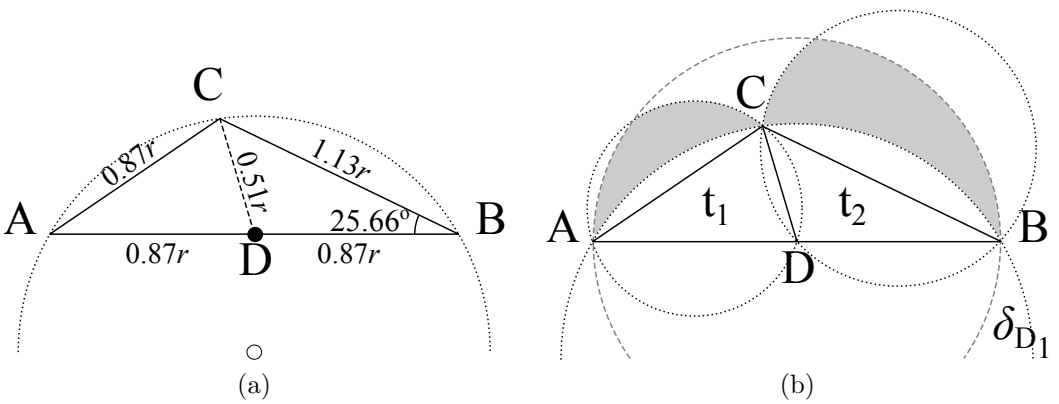


Figure 5.22: Obtuse terminal triangle $t(ABC)$ with $\gamma = 120°$ and $\alpha = 25.66°$. (a) Relationship between edges and circumradius r. (b) Shaded areas define the regions of potential points that could produce edges smaller than CA through edge-flip operations. δ_{D_1} is the circle of radius $|CA|$ centered at D. The dotted circles are the circumcircles of triangles t_1 and t_2.

Under these circumstances the algorithm creates one mandatory edge which is smaller than edge CA: edge DC ($|DC| > 0.59|CA|$). Additional edges smaller than CA could be created through edge-flip operations if a point lies (1) within a distance $d(A, C)$ from point D and (2) inside one of the circumcircles of the newly created triangles. As illustrated in Figure 5.22(b), this region is defined by the intersection of circle δ_{D_1} (of radius $d(A, C)$ and centered at point D) and the circumcircles of triangles $t_1(ADC)$ and $t_2(BCD)$. Note that any edge-flip operation performed over triangles outside δ_{D_1} can only produce edges longer than CA.

In what follows, we analyze how local triangles are affected by the creation of new slightly smaller edges. We discuss each scenario in which the algorithm creates edges smaller than those affected by refinement, and show that the new edges can only be associated with good-quality triangles.

Characterization of Edges Smaller Than DC

Consider δ_{D_2} the circle of radius $|DC|$ centered at point D. The small crescent-shape region between δ_{D_2} and the circumcircle of t, region R_1, defines the area where edge-flip operations could create edges smaller than DC; see Figure 5.23. Consider point E the point lying in region R_1 so that edge EC is present in the triangulation. This edge will be associated with the small-angled neighbor triangle $t'(CBE)$, with smallest angle $\alpha_{t'}$ smaller than $\approx 8°$.

Figure 5.24 illustrates the improvement of triangles in this scenario. It can be observed that edge EC is significantly smaller than any new edge that the algorithm could create after processing triangle $t(ABC)$.

Compared to Ruppert's refinement algorithm [53], the insertion of point D produces better triangles (in terms of the internal angles) than inserting the circumcenter of t'.

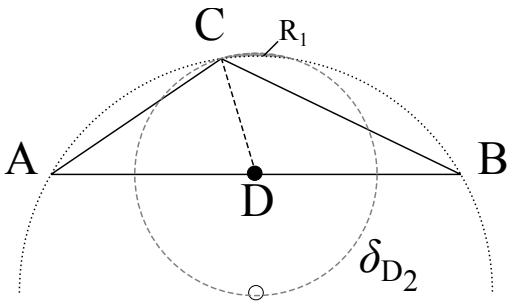


Figure 5.23: Points lying in (shaded) region R_1 define the candidate triangles which produce an edge smaller than DC after one edge-flip operation. δ_{D_2} is the circle of radius $|DC|$ centered at D.

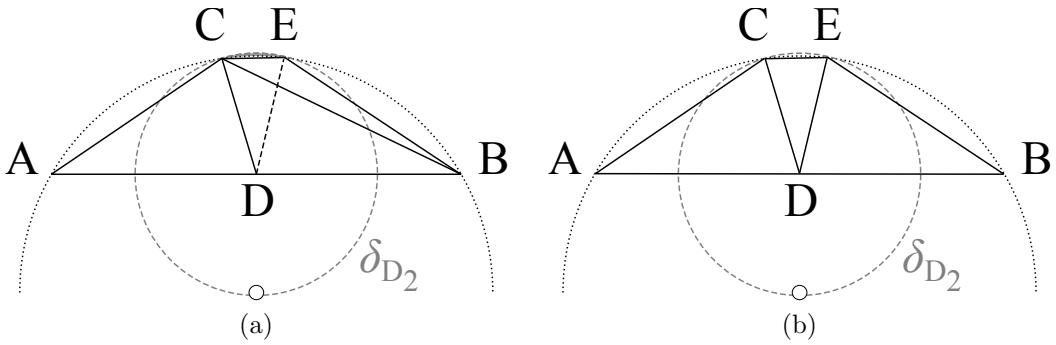


Figure 5.24: Example of the quality improvement obtained after refinement using the Lepp-Delaunay algorithm. (a) Initial triangulation. Triangle CBE is the midsize-edge neighbor of t. (b) Refined triangulation. Triangles have smallest angles greater than θ.

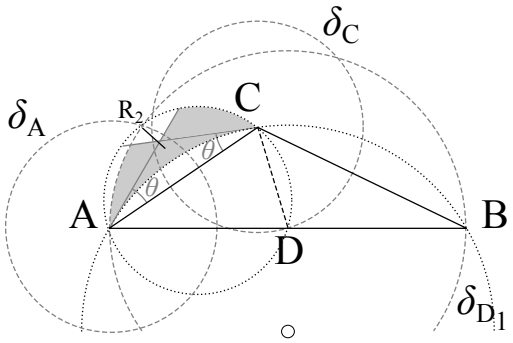


Figure 5.25: Points lying in (shaded) region R_2 define the candidate parent triangle $\hat{t}$ which produces an edge smaller than CA after one edge-flip operation. δ_{D_1} is the circle of radius $|CA|$ centered at D. δ_A and δ_C are the circles of radius $|DC|$ centered at A and C respectively.

The following proposition summarizes this observation.

Proposition 5.24. *If the algorithm creates a new edge adjoining D which is smaller than DC, then there exists an edge adjoining C in the triangulation which is smaller than any edge the algorithm could create.*

Characterization of Edges Smaller Than CA

Terminal triangle t will be longest-edge bisected by the Lepp-Delaunay algorithm in one of the following two scenarios: (1) t is the terminal triangle in the Lepp of the current target triangle being processed, or (2) t is itself the current target triangle being processed.

Processing terminal triangles: For the first scenario, due to the processing order of triangles, if t is not the current target triangle being processed, then parent triangle $\hat{t}$ is the triangle with the smallest edge $l_{\min}$ among the target triangles in S_{target}. The smallest angle of $\hat{t}$, $\alpha_{\hat{t}}$, is necessarily smaller than α (otherwise $\hat{t}$ could not belong to S_{target}). Although $l_{\min}$ is smaller than edge CA, in some scenarios it could be longer than edge DC. The rest of this analysis considers the worst-case scenario in which t is the longest-edge neighbor of $\hat{t}$.

On the one hand, if $\hat{t}$ is the shortest-edge neighbor of t, then $\hat{t}$ will have shortest edge $l_{\min}$ smaller than any edge the algorithm could create after processing t.

Consider δ_A and δ_C the circles of radius $|DC|$ centered respectively at point A and at point C. Region R_2 (shaded area in Figure 5.25) defines the region where the third vertex of $\hat{t}$ could lie. Since this region is fully contained within circles δ_A and δ_C, $l_{\min}$ will necessarily be smaller than edge DC. Therefore, the algorithm will not create any edge smaller than $l_{\min}$ through edge-flip operations, nor will it produce triangles with angles smaller than $\alpha_{\hat{t}}$.

On the other hand, if $\hat{t}$ is the midsize-edge neighbor of t, then $\hat{t}$ could have its shortest edge, $l_{\min}$, longer than edge DC, the shortest edge created by the algorithm after the longest-

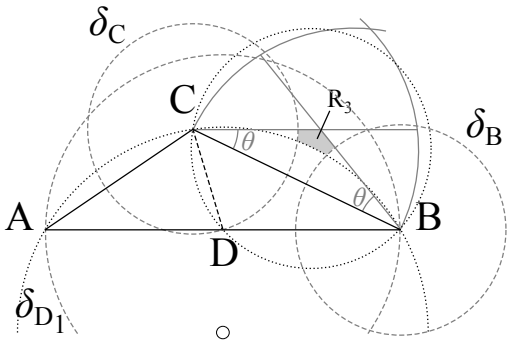


Figure 5.26: Points lying in (shaded) region R_3 define the candidate parent triangle $\hat{t}$ which produces an edge smaller than CA after one edge-flip operation. δ_{D_1} is the circle of radius $|CA|$ centered at D. δ_B and δ_C are the circles of radius $|DC|$ centered at B and C respectively.

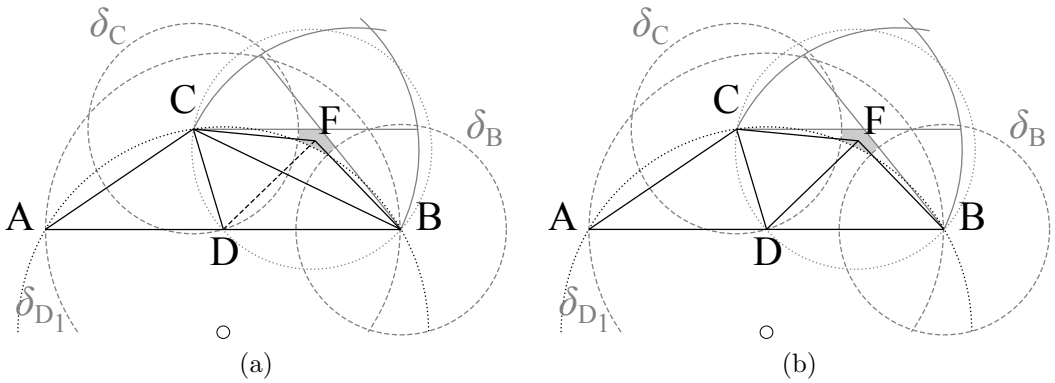


Figure 5.27: Example of the quality improvement after refinement using the Lepp-Delaunay algorithm. (a) Initial triangulation. Triangle CBF is the midsize-edge neighbor of t. (b) Refined triangulation. Triangles are quasi-equilateral and have smallest angles greater than θ.

edge bisection of t. Although the algorithm could create an edge slightly smaller than $l_{\min}$, the triangles associated with this edge will be of good quality.

Consider δ_B and δ_C the circles of radius $|DC|$ centered respectively at point B and at point C. Region R_3 (shaded area in Figure 5.26) defines the region of the third vertex of triangle $\hat{t}$ so that $l_{\min} > DC$, in which case the algorithm could introduce an edge smaller than $l_{\min}$. Otherwise, $l_{\min}$ is smaller than DC and the algorithm does not produce new small edge.

If a new smallest edge is in fact created, the triangles associated with this edge will be quasi-equilateral. Figure 5.27 illustrates the improvement of triangles for this scenario. Triangle $\hat{t}(CBF)$ has smallest angle $\alpha_{\hat{t}}$ between 17.17° and 25.66°. After the edge-flip operation edge FD will be longer than $l_{\min}$ and every triangle produced will have angles above 30°. Let $l_{\min} = FC$. Edge FC attains maximum length $(0.63r)$ when $\alpha_{\hat{t}} = 25.66°$ and $\hat{t}$ is isosceles, therefore the algorithm would introduce edge DC with length 0.82 of that of $l_{\min}$.

The following proposition summarizes this observation.

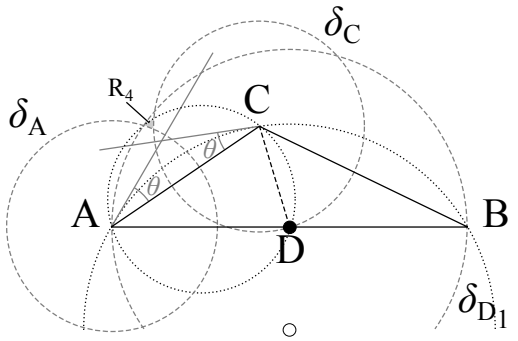


Figure 5.28: Points lying in (shaded) region R_4 define the shortest-edge neighbor triangle which produces an edge smaller than CA after one edge-flip operation. δ_{D_1} is the circle of radius $|CA|$ centered at D. δ_A and δ_C are the circles of radius $|DC|$ centered at A and C respectively.

Proposition 5.25. *If t is the terminal triangle in the Lepp of the current target triangle $\hat{t}$ being processed, then $|DC| > 0.82|l_{\min}|$, where $l_{\min}$ is the shortest edge of $\hat{t}$.*

□

Processing target triangles: For the second scenario, in which t is the current target triangle being processed ($l_{\min} = CA$), the shortest-edge and midsize-edge neighbors of t must have angles greater than θ (otherwise they would belong to S_{target} and should be processed before t). These neighbor triangles must have at least one edge smaller than $l_{\min}$ in order to force the algorithm to create a new small edge through an edge-flip operation.

We analyze the two cases where the algorithm produces edges smaller than $l_{\min}$, and establish bounds on the length of these new edges. We also study the geometry of the triangles before and after refinement. The two cases are defined by the geometry of the shortest-edge neighbor, and that of the midsize-edge neighbor:

1. Consider the shortest-edge neighbor triangle $t_s(ACG)$. Also consider δ_A and δ_C the circles of radius $|DC|$ centered respectively at point A and at point C; as shown in Figure 5.28. Region R_4 defines the region where point G of triangle t_s has edges longer than DC and the edge-flip operation creates edge DG which is smaller than CA. It must be noted that if such a triangle exists, it must have two edges smaller than CA, and $l_{\min}$ is not the smallest edge in the triangulation.

   Figure 5.29 illustrates an example of the improvement of triangles. Since t_s lies inside the circumcircle of t_1 (left descendant of t), it holds that $\angle AGC < 180° - \angle ADC$, so the smallest edge of t_s attains its maximum length when t_s is isosceles. Let edge CG be this edge. Then $|CG| < \frac{|CA|}{2\sin 53.58°} \approx 0.54r$. In this case, the ratio between edge DC, the new smallest edge created by the algorithm, and edge CG, the smallest edge of the known triangulation, is at least 0.95. After the edge-flip operation, edge DG will not be smaller than CG and the triangles produced will all have angles greater than 25.66°.

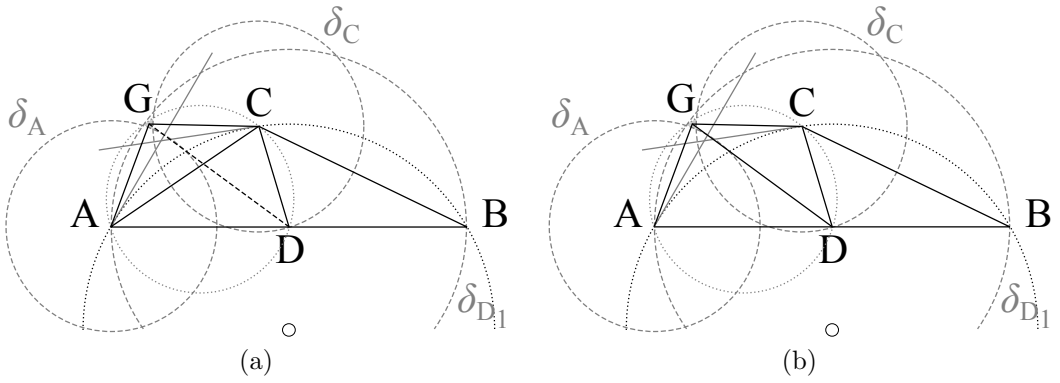


Figure 5.29: Example of the quality improvement after refinement using the Lepp-Delaunay algorithm. (a) Initial triangulation. Triangle ACG is the shortest-edge neighbor of t. (b) Refined triangulation. Triangles are quasi-equilateral and have smallest angles greater than θ.

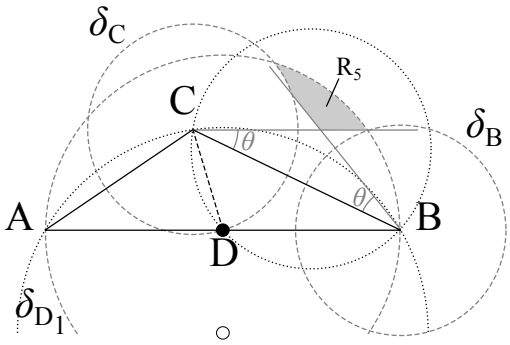


Figure 5.30: Points lying in (shaded) region R_5 define the shortest-edge neighbor triangle which produces an edge smaller than CA after one edge-flip operation. δ_{D_1} is the circle of radius $|CA|$ centered at D. δ_B and δ_C are the circles of radius $|DC|$ centered at B and C respectively.

2. Consider midsize-edge neighbor triangle $t_m(BHC)$. Again, consider δ_B and δ_C the circles of radius $|DC|$ centered respectively at point B and at point C; as shown in Figure 5.30. Then region R_5 defines the area in the triangulation where point H of triangle t_m has edges longer than DC so that the edge-flip operation creates the edge DH which is smaller than CA. Similar to the shortest-edge neighbor, if such a triangle exists it must have at least one edge smaller than CA, so $l_{\min}$ is not the smallest edge in the triangulation.

   Figures 5.31 and Figure 5.32 illustrate two examples of the improvement of triangles for this configuration. Since t_m lies both inside the circumcircle of t_2 (right descendant of t) and δ_{D_1}, the smallest edge of t_m attains its maximum length when t_m is isosceles (Figures 5.31). Let edge H_1C be this edge. Then $|H_1C| \approx 0.72r$, in which case, the ratio between edge DC and edge H_1C, the smallest edge of the known triangulation, is at least 0.7. Edge DH_1 (created after the edge-flip operation) will be longer than H_1C, and every triangle produced will be regarded as quasi-equilateral, with angles greater than 30°.

   Another example is shown in Figure 5.32. The configuration of triangle t_m produces

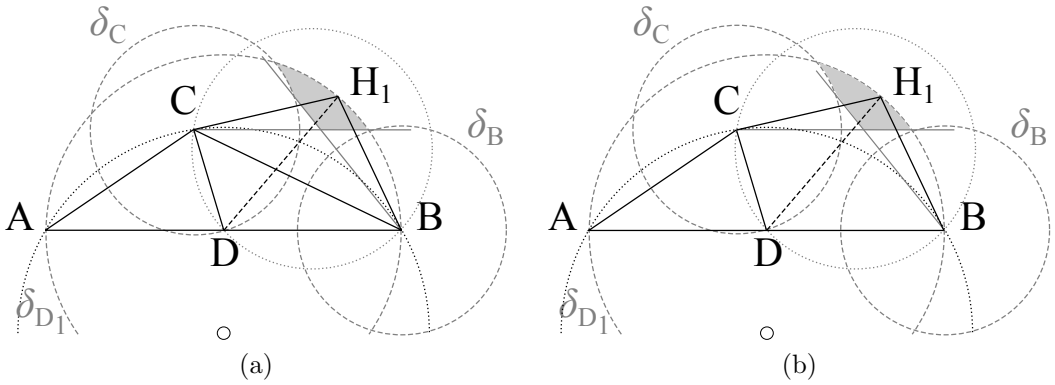


Figure 5.31: Example of the quality improvement after refinement using the Lepp-Delaunay algorithm. (a) Initial triangulation. Triangle BH_1C is the midsize-edge neighbor of t such that the length of shortest edge is maximized. (b) Refined triangulation. Triangles are quasi-equilateral and have smallest angles greater than θ.

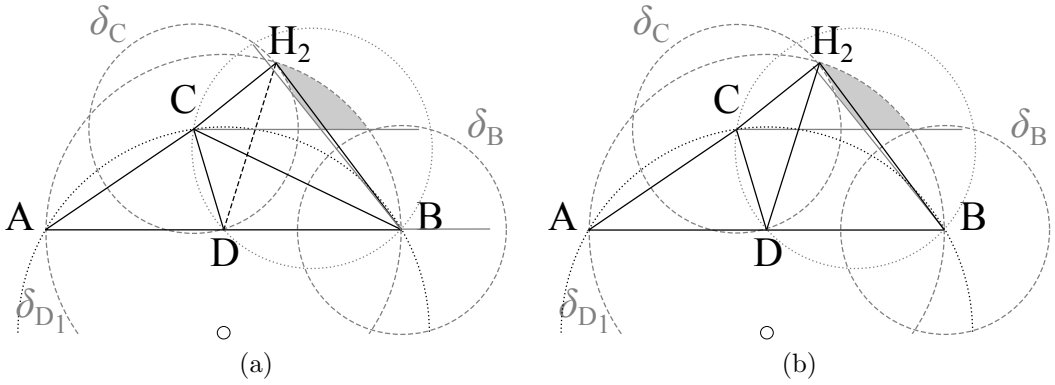


Figure 5.32: Example of the quality improvement after refinement using the Lepp-Delaunay algorithm. (a) Initial triangulation. Triangle BH_2C is the midsize-edge neighbor of t such that after the edge-flip operation triangle DH_2C has the smallest angle. (b) Refined triangulation. Triangles are quasi-equilateral and have smallest angles greater than θ.

the smallest angle $\angle CH_2D$, but even in this scenario every triangle produced after refinement will be considered quasi-equilateral, with angles greater than 30°.

The following proposition summarizes these observations and defines the bounds on edge DC (in terms of smallest edge $l_{\min}$).

Proposition 5.26. *If a terminal triangle t is the current target triangle being processed, then* $0.59|l_{\min}| < |DC| < 0.70|l_{\min}|$, *if* CA *is the smallest edge among the edges of t and its direct neighbors. If a new edge smaller than* CA *is created, then* $0.7|l_{\min}| < |DC| < 0.82|l_{\min}|$, *and* $l_{\min}$ *corresponds to this new small edge.*

Characterization of Edge DC

Until now, we have discussed the scenarios in which edge-flip operations creates edges smaller than CA. When no edge-flip operation is required to maintain the Delaunay triangulation, edge DC remains as the single new edge smaller than CA (since $|DC| > 0.59|CA|$). If triangle t is both terminal triangle and the triangle in S_{target} with the smallest edge ($l_{\min} = CA$), then the algorithm will produce an edge smaller than $l_{\min}$.

In such scenario triangle t_1, the left descendant of t, is a quasi-equilateral triangle; and t_2, the right descendant of t, becomes the target triangle with the smallest edge ($l_{\min} = DC$). It must be noted that although t_2 still has a smallest angle of 25.66°, it is less obtuse than t and has a better radius-edge ratio.

Triangle t_2 will be eliminated by an edge-flip operation without generating smaller edges than DC. In the worst-case scenario, after eliminating triangle t_2, the algorithm could produce a triangle similar to t (with same radius-edge ratio). This triangle is not terminal and will be eliminated after an edge-flip operation, and the geometry will not be reproduced. Figure 5.33 illustrates this scenario.

Triangle $t(ABC)$ is both terminal triangle and the triangle in S_{target} with the smallest edge, edge CA. Neighboring triangles will either have angles above threshold θ or edges longer than CA. A triangle BIC candidate to insert the midpoint of its longest-edge would be, in the worst-case, similar to t. After its longest-edge bisection, point J would lie on the circumcircle of $t_2(BCD)$ (Figure 5.33(a)). The corresponding edge-flip operation produces triangle $t_3(DJC)$, which is similar to t and has shortest edge CA, making this triangle the next target triangle to be processed (Figure 5.33(b)). Since longest-edge neighbor triangle BJD does not share terminal edge with t_3, the latter will be removed through edge-flip operations. Eventually, the algorithm introduces a point K inside the circumcircle of t_3 – in the worst case this point corresponds to the midpoint of edge BJ (Figure 5.33(c)). Finally, the last edge-flip operation eliminates t_3 and produces quasi-equilateral triangles, all with edges longer than $l_{\min}$ (Figure 5.33(d)).

The following proposition summarizes this observation.

Proposition 5.27. *If a terminal triangle t is the current target triangle being processed, then $|DC| > 0.59|l_{\min}|$, where $l_{\min} = CA$ is the shortest edge of t in a configuration where triangle t_2 is not removed immediately after processing t. Triangle t_2 is later removed by the algorithm producing edges that are longer than DC.*

Finally, this scenario is avoided for any obtuse triangle with $\gamma < 97.7°$.

Acute Triangles

When the terminal triangle t is acute, new edge DC, as well as edges AD and BD, will be longer than smallest edge CA, implying that the algorithm could not create edges smaller

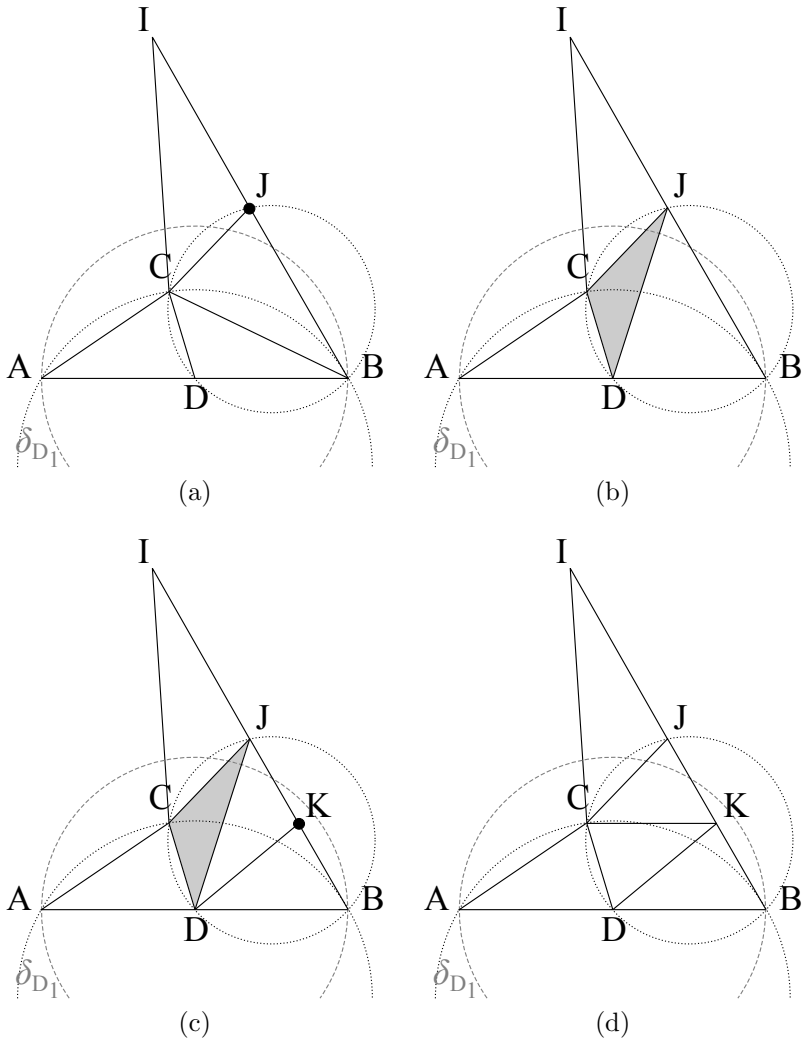


Figure 5.33: Example of the quality improvement after refinement using the Lepp-Delaunay algorithm. (a) Initial triangulation. Triangle BH_2C is the midsize-edge neighbor of t such that after the edge-flip operation triangle DH_2C has the smallest angle. (b) - (d) Refinement process. Triangles in final triangulation are quasi-equilateral and have smallest angles greater than θ.

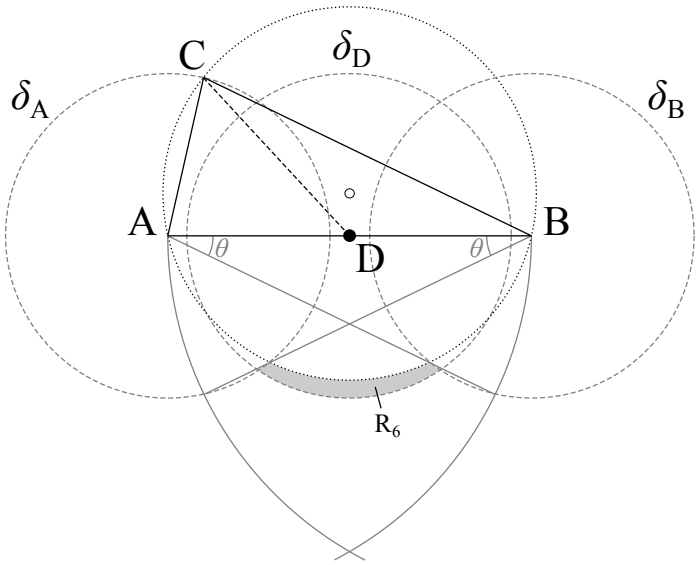


Figure 5.34: Processing of acute terminal triangle $t(ABC)$. Points lying in (shaded) region R_6 define the longest-edge neighbor triangle which produces an edge smaller than CA after the insertion of point D. δ_A, δ_B and δ_D are the circles of radius $|CA|$ centered at A, B and D respectively. The dotted circle corresponds to the circumcircle of triangle t.

than those already existing in the triangulation. We focus our analysis on the worst-case scenario, in which t is the current target triangle being processed ($l_{\min} = CA$, where $l_{\min}$ is the smallest edge among the triangles in S_{target}) and the algorithm creates an edge slightly smaller than $l_{\min}$. We show that this new edge will be associated with triangles of better geometric quality.

Consider terminal triangle $t(ABC)$ with smallest angle $\alpha = 25.66°$ and largest angle $\gamma = \frac{180°-\alpha}{2} \approx 77.17°$, such that t is the acute triangle with the smallest γ; see Figure 5.34. Consider δ_A, δ_B and δ_D the circles of radius $|CA|$ centered respectively at points A, B and D. Region R_6 defines the area for vertex L of longest-edge neighbor $t'(BAL)$ so that $DL < DC$.

The smallest possible edge DL is produced when point L lies the closest to D outside of the circumcircle of t, then $|DL| > 1 - \cos\gamma \approx 0.78r$. Since $|CA| = 2\sin\alpha \approx 0.86r$, it follows that $|DL| > 0.9|CA|$. Figure 5.35 illustrates this scenario. Triangle t' has largest angle $\gamma' > 180° - \gamma \approx 102.83°$ and its two descendants are quasi-equilateral triangles and will not be further divided because they cannot become terminal triangles nor removed through edge-flip operations (otherwise t is not the target triangle with smallest edge).

Regardless of the geometry of t', after splitting t its left descendant $t_1(DCA)$ is a quasi-equilateral triangle, while its right descendant $t_2(BCD)$ is a too-obtuse triangle and will be later removed through an edge-flip operation. Furthermore, the point triggering the edge-flip operation that eliminates t_2 lies outside of δ_D and does not create edges smaller than $l_{\min}$.

The following proposition summarizes this observation.

Proposition 5.28. *If acute terminal triangle t with $\alpha = 25.66°$ is the current target triangle being processed, the algorithm could create a new edge which is at least $0.9|l_{\min}|$, where $l_{\min}$*

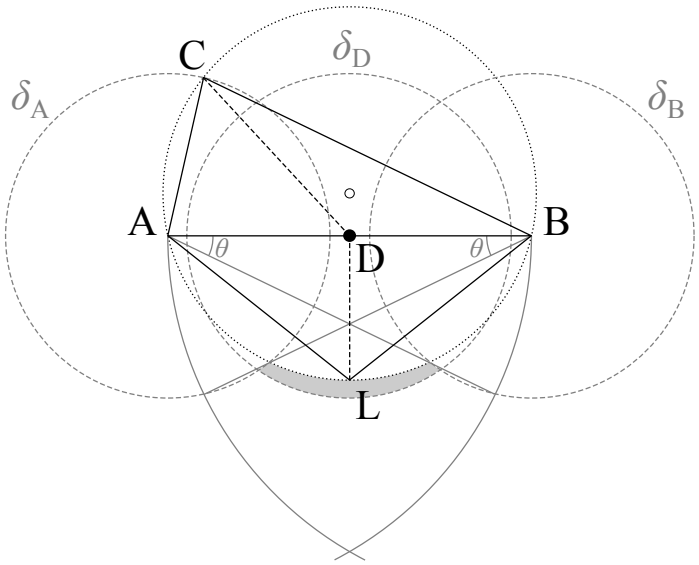


Figure 5.35: Example of the refinement of terminal triangles $t(ABC)$ and $t'(BAL)$. Triangle t is acute with smallest angle 25.66°. Triangle t' is obtuse with smallest angle 38.59°. Descendants of t' will be of good quality, with angles greater than θ.

is the shortest edge of t and the smallest among the target triangles. The triangles associated with this new edge will be of good quality and no other edge smaller than $l_{\min}$ *will be produced.*

Finally, this scenario is avoided for any acute triangle with $\alpha < 23.47°$.

Remarks

Based on Proposition 5.24, 5.25, 5.26, 5.27 and 5.28, the following lemma summarizes the algorithm's performance working with an input angle tolerance parameter of 25.66°.

Lemma 5.29. *Let* $\theta = 25.66°$ *be the angle tolerance parameter such that target triangles in* S_{target} *have angles smaller than* θ*. Then, the Lepp-Delaunay algorithm produces an output triangulation with edges longer than* $l_{\min}$*, the smallest edge of the input triangulation. Moreover, if a new small edge is created, it will be longer than* $l_{\min}/\sqrt{3}$ *and will be associated with triangles of improved quality.*

An angle bound $\theta = 25.66°$ ensures that the two edges created after splitting the longest-edge of the terminal triangle are at least as long as the shortest edge. As discussed throughout the section, in some isolated scenarios a new edge smaller that $l_{\min}$ could be created (usually, when a target terminal triangle has a largest angle close to 120°). In this case the new edge still has a bounded length ($l_{\min}/\sqrt{3}$) and is associated with good quality triangles. As discussed throughout this section, this edge cannot be involved in the generation of new small edges.

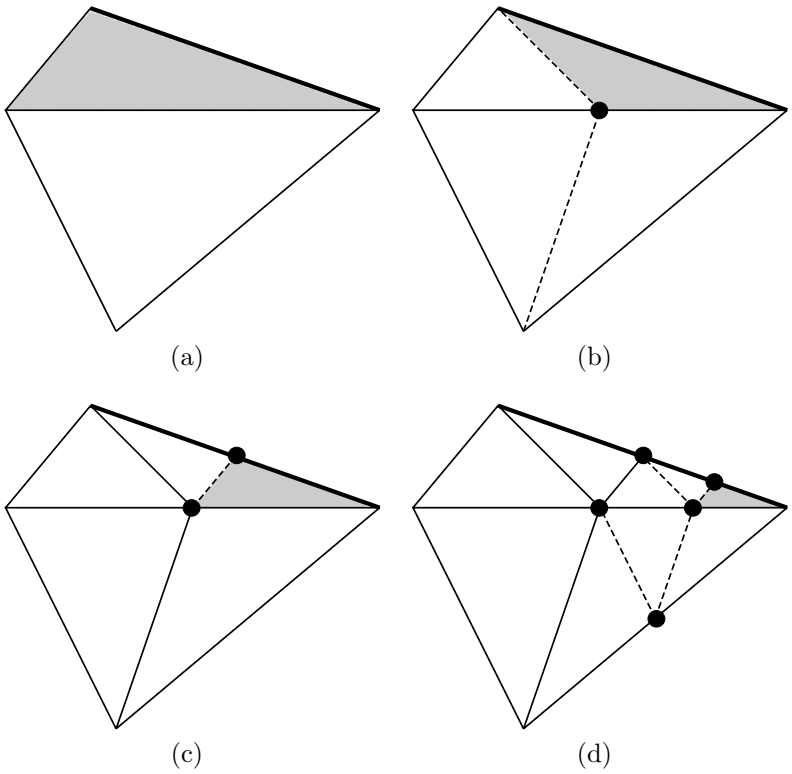


Figure 5.36: If the terminal triangle has its midsize edge constrained (bold), the iterative longest-edge bisection reproduces its geometry, creating an infinite processing cycle. Bad quality triangles are shaded.

5.6 Processing Constrained Edges

In this section we discuss the issues related to the refinement of triangles with constrained edges. The Lepp-Delaunay algorithm must ensure that the point inserted at the midpoint of a terminal edge does not lie arbitrarily close to a constrained edge. Doing so could force the algorithm to produce arbitrarily small triangles between the inserted point and the constrained edge.

This issue is addressed by adding a verification of the point selection strategy of the algorithm that prevents midpoints of terminal edges from getting too close to a constrained edge. As described by the Point Selection Strategy in Algorithm 3.4, if the longest-bisection of a terminal triangle with constrained midsize edge produces a triangle with an angle smaller than 30°, then the algorithm selects the midpoint of the constrained edge, performing the bisection by the midsize edge of the terminal triangle.

This verification was initially introduced to avoid undesirable infinite processing cycles, which arise when the iterative longest-edge bisection of a bad terminal triangle reproduces the geometry of the triangle. Figure 5.36 illustrates this scenario.

On the other hand, a bisection by the constrained edge quickly eliminates the bad quality triangle without producing new similar triangle nor threatening the algorithm's termination,

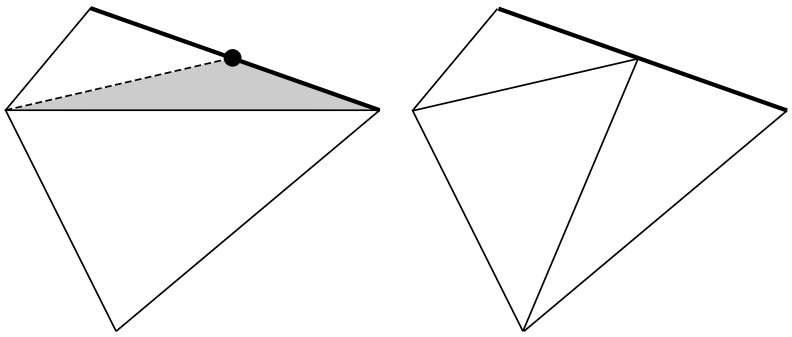


Figure 5.37: The insertion of the midpoint of the constrained edge of a terminal triangle does not reproduce the geometry and eliminates target terminal triangles in a few steps. Bad quality triangles are shaded.

as illustrated in Figure 5.37.

The verification of angles in the point selection strategy implicitly defines a bound on the distance between new points and constrained edges. The 30° angle threshold restrain the algorithm from selecting points that lie too close to a constrained edge, as discussed in Section 5.3.3. This approach eliminates the scenario of infinite processing cycles and also improves the distribution of points along boundary and interior constrained edges.

Any bad quality terminal triangle that has its midsize edge constrained, and smallest angle below 30°, is eliminated by a bisection by the midsize edge. If the new triangles produced after this operation are considered of bad quality, they are eliminated either by edge-flip operations or by the insertion of new points as discussed in Section 5.5. In practice, the point selection strategy terminates and produces well graded triangulations for angle parameters of up to 30°.

It is worth noting that a terminal triangle with constrained longest edge is always longest-edge bisected, regardless of its quality. Moreover, if the terminal triangle has a largest angle of more than 120°, it necessarily has a constrained longest edge along the boundary of the input PSLG, and could only be produced by the initial constrained triangulation.

In Section 5.6.1 we show that the point selection strategy can be improved in order to work with angle parameters greater than 30°. Our proposed strategy adds flexibility to the original verification of constrained edges, increasing the scenarios where the longest-edge bisection is allowed without affecting the bounds on the insertion radius.

We then show that giving priority to target triangles with constrained edges helps to reduce the number of points inserted by the algorithm (Section 5.6.2). Finally, in Section 5.6.3, we establish the minimum angle at which two adjoining constrained edges could meet so that the bounds on the insertion radius are respected and the algorithm's termination is not threatened.

5.6.1 Improved Point Selection Strategy for Constrained Edges

A limitation of the 30° threshold of the Point Selection Strategy in Algorithm 3.5 is related to the termination of the algorithm for parameter $\theta > 30°$. Consider terminal triangle t_c with constrained midsize edge and smallest angle α such that $30° < \alpha < \theta$. If the terminal edge midpoint is selected for insertion, the algorithm will fail to terminate because of the infinite processing cycle produced by an iterative longest-edge bisection of terminal triangle. This behavior is observed in Figure 5.36.

We propose an improved point selection strategy that deals with this limitation, allowing the algorithm to work with quality parameters above 30°. Additionally, our strategy provides a relaxes the verification of terminal triangles with angles smaller than 30°. The new point selection strategy is described in Algorithm 5.1.

Algorithm 5.1 Improved Point Selection Strategy (midpoint)

Input: Terminal triangles t_i, t_j, terminal edge e_{term}
Output: Point P

Select point P, midpoint of e_{term}
if e_{term} is not constrained **then**
  if midsize edge of t_i is constrained **then**
    Let e_{cons} be the constrained edge of t_i
    if t_i is a bad quality triangle **or** the distance from midpoint of e_{term} to midpoint of e_{cons} is less than length$(e_{\text{cons}})/2\sqrt{3}$ **then**
      P is the midpoint of constrained edge e_{cons}
    end if
  else
    Perform the same verification for triangle t_j
  end if
end if

For some obtuse terminal triangles with constrained midsize edge and angles smaller than 30°, selecting the midpoint of the terminal edge could produce a better angle distribution than splitting the constrained edge. Our improved strategy incorporates a flexible approach that takes advantage of this scenarios, allowing longest-edge bisection of terminal triangles with angles below 30°. This modification improves the algorithm's convergence since the longest-edge bisection operation tends to produce sub-triangles with better angle distribution than bisections by the midsize edge. In practice, a faster convergence reduces the points inserted by the algorithm.

Consider a Delaunay terminal triangle $t(ABC)$ with longest edge AB, constrained midsize edge BC, largest angle $\gamma = 120°$, and smallest angle $\alpha = 25.66°$. Let $t(AC'B)$ be its longest-edge neighbor, the equilateral triangle, as illustrated in Figure 5.38(a).

Since $\alpha < 30°$, the original point selection strategy will perform the bisection by the midsize edge of t: the algorithm inserts point E, midpoint of edge BC, producing two triangles with angles smaller than α since $\angle EC'B \approx 17.65°$ and $\angle CAE \approx 23.85°$ (Figure

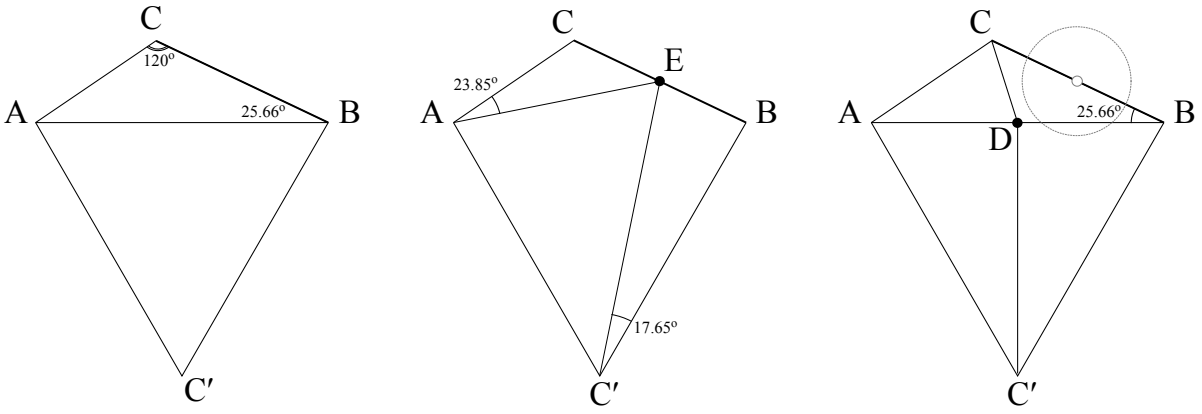


Figure 5.38: An example of the advantage of selecting the midpoint of a terminal edge for terminal triangles with angles smaller than 30°. (a) Terminal triangles $t(ABC)$ and $t'(AC'B)$. Triangle t has a constrained midsize edge BC and smallest angle $\alpha = 25.66°$. (b) Triangles obtained after the insertion of the midpoint of the constrained edge (original strategy). (c) Triangles obtained after the insertion of the midpoint of the terminal edge (proposed strategy). Grey circle centered at midpoint of $|BC|$ has radius $|BC|/2\sqrt{3}$.

5.38(b)). If we consider an angle parameter $\theta = 25.66°$, these two sub-triangles will be regarded as bad quality triangles and will be removed after further processing.

For the same scenario, the proposed point selection strategy allows the longest-edge bisection of t since $d(E, D) > |BC|/2\sqrt{3}$. In fact, $d(E, D) = r \sin 25.66° \approx 0.43r$ and $|BC|/2\sqrt{3} = r \sin 34.34°/\sqrt{3} \approx 0.32r$. The insertion of point D does not produce triangles with angles smaller than α, as illustrated in Figure 5.38(c).

The advantage over the original point selection strategy is observed for angle parameters θ between 21.21° and 30°. Figure 5.39 illustrates the boundary case of Delaunay terminal triangle $t(ABC)$ with largest angle $\gamma = 120°$ and smallest angle $\alpha = 21.21°$. Neither the original nor the improved point insertion strategies produce angles smaller than 21.21°. For terminal triangles with angles smaller than 21.21° both strategies are equivalent.

□

According to the improved point selection strategy, the midpoint of the constrained midsize edge is selected when the distance from the midpoint of the longest edge to the midpoint of the midsize edge is smaller than a $\frac{1}{2\sqrt{3}}$ of the midsize edge. This is equivalent to the equation $\sin\alpha < \frac{\sin\beta}{2\sqrt{3}}$, where α is the smallest angle of the triangle and β the midsize angle. The bounds on the insertion radius defined in Section 5.3.3 hold for the proposed insertion strategy. Moreover, the relationship between the insertion radius of the selected point and the insertion radius of its parent is the same as discussed in Section 5.4.3.

□

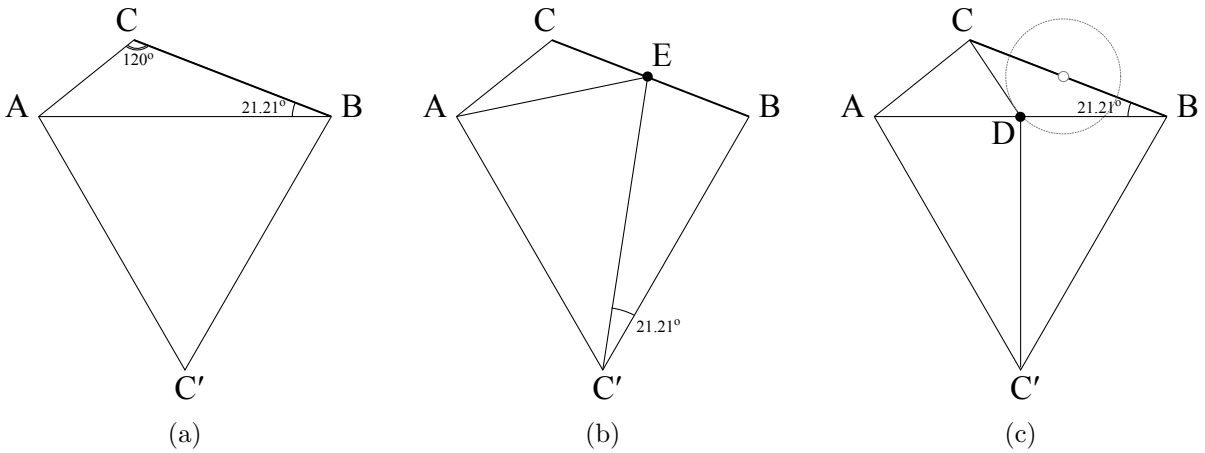


Figure 5.39: Example of the boundary case where the original strategy does not produce triangles with smaller angles. (a) Terminal triangles $t(ABC)$ and $t'(AC'B)$. Triangle t has a constrained midsize edge BC and smallest angle $\alpha = 21.21°$. (b) Refinement with the original strategy. (c) Refinement with the improved strategy. Grey circle centered at midpoint of $|BC|$ has radius $|BC|/2\sqrt{3}$.

Finally, Figure 5.40 and Figure 5.41 show an empirical comparison between the original and the improved selection strategies. Both figures show the algorithm termination producing well-graded triangulations with internal angles greater than 30°. For the same triangulations we show that, unlike the improved strategy, the original point selection strategy is more likely to produce infinite processing cycles around constrained edges for values of θ above 30°.

5.6.2 Giving Priority to Target Triangles with Constrained Edges

In this section we proposed a simple modification of the algorithm that reduces the number of points inserted by the algorithm near constrained edges. Given the set of target triangles, S_{target}, we propose to first process the subset of target triangles with constrained edges (in any given order), followed by the subset of target triangles with no constrained edges (also in any given order).

Before refining a terminal triangle with constrained midsize edge, the point selection strategy in Algorithm 5.1 verifies that the midpoint of longest edge is not too close to the midpoint of its constrained midsize edges. Since this verification is only performed for terminal triangles, working with an initial triangulation with small-angled triangles near constrained edges, the algorithm could insert a points too close to the constrained edge if a neighboring triangle is refined before the triangle with the constrained edge; see Figure 5.42.

For example, if the propagating path of a bad quality triangle t with constrained midsize edge has a sequence of small-angled triangles between the constrained edge and the terminal triangles in Lepp(t), then the algorithm could insert the midpoint of a terminal triangle close to the constrained edge, forcing the creation of a new small feature associated with new small-angled triangles. Figure 5.43 illustrates this scenario. If t_1 is processed before t_0, the

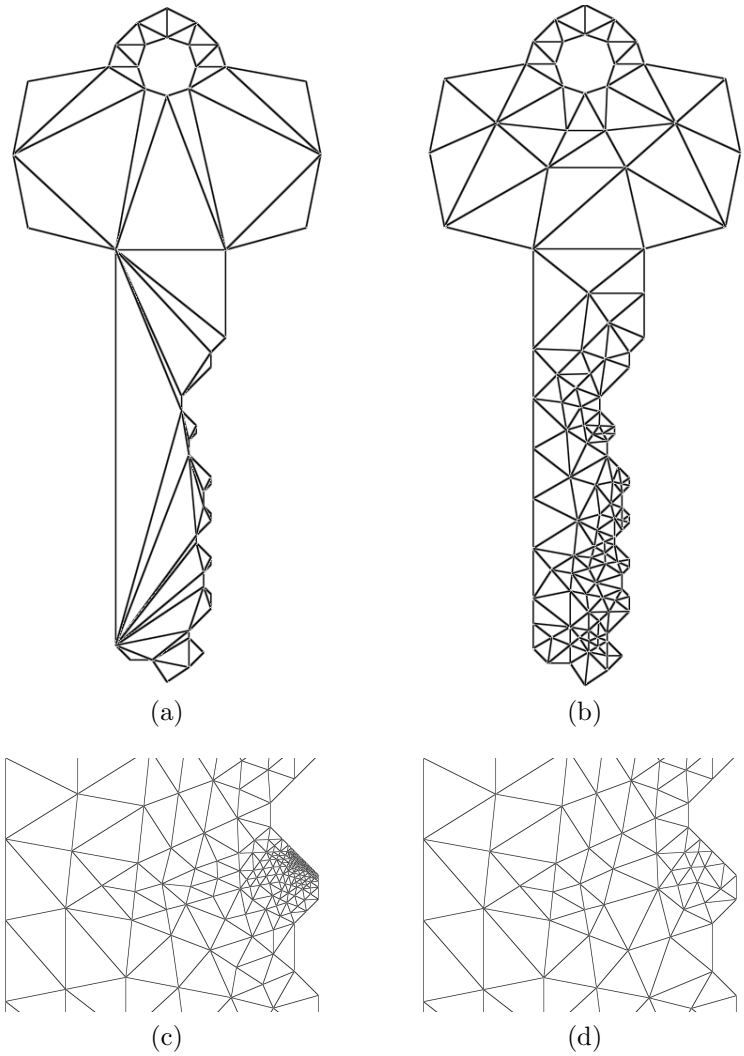


Figure 5.40: Comparison between the original and the improved point selection strategies over constrained edges. (a) Constrained Delaunay triangulations of a key shaped geometry. (b) Refined triangulation produced by both strategies with no internal angles smaller than $30°$. (c) For $\theta = 34°$, using the original strategy an infinite processing cycle is produced around some constrained edges after a few iterations. (d) Same geometric features of the refined triangulation produced by the improved strategy.

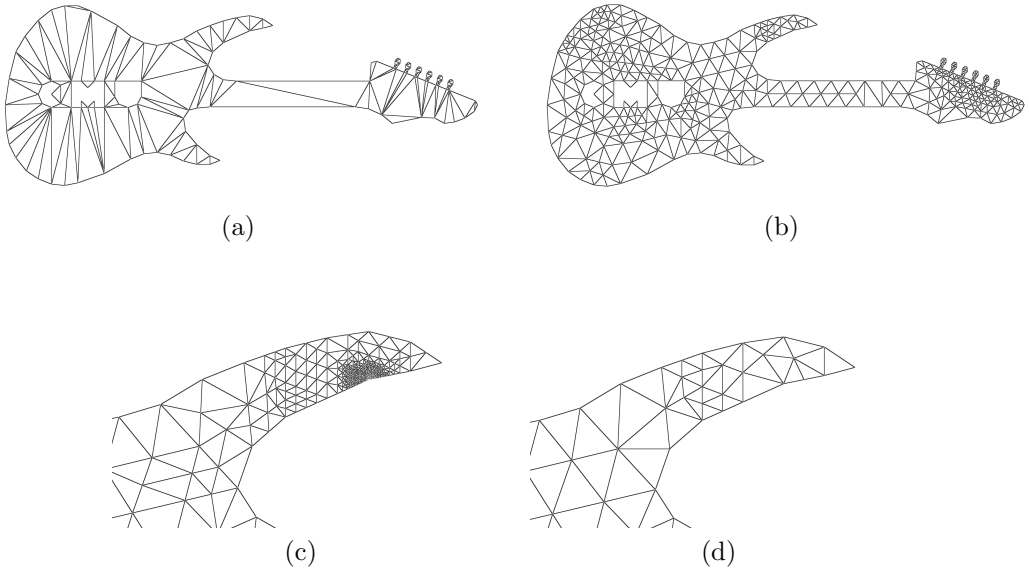


Figure 5.41: Comparison between the original and the improved point selection strategies over constrained edges. (a) Constrained Delaunay triangulations of a guitar shaped geometry. (b) Refined triangulation produced by the improved point selection strategy with no internal angles smaller than 32°. The original strategy fails to terminate. (c) The original strategy produces an infinite processing cycle around constrained edges after few iterations. (d) Same geometric feature of the refined triangulation produced by the improved strategy.

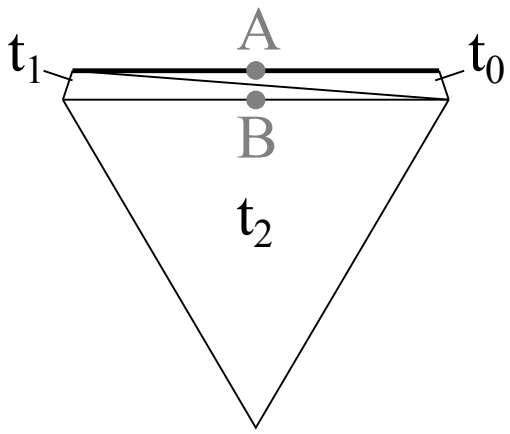
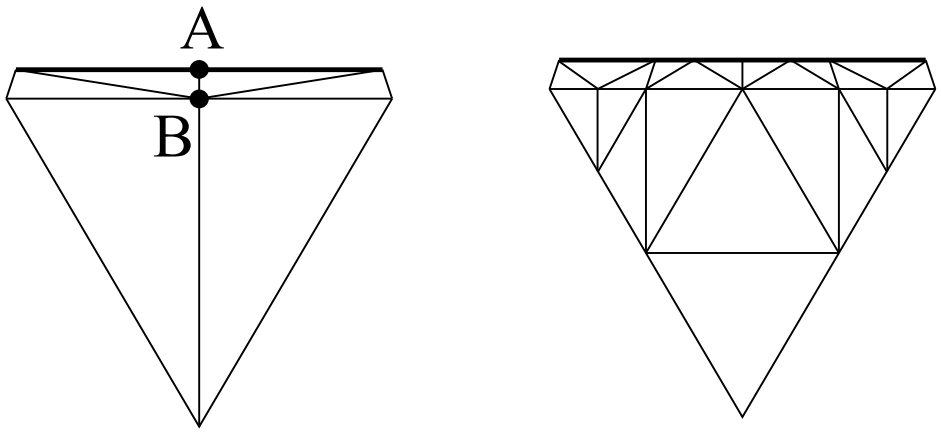


Figure 5.42: Example of an initial CDT with the midpoint of longest edge of terminal triangles t_1 and t_2 close to a constrained edge (bold). Point A is the midpoint of the constrained edge. Point B is the midpoint of the terminal edge.



(a) (b)

Figure 5.43: (a) The insertion of point B forces the insertion of point A. Length of edge AB is bounded by the geometry. (b) Refined triangulation with no angles smaller than 20°.

algorithm inserts point B, eventually forcing the insertion of point A. This scenario creates a new small geometric feature, edge AB, which will be later adapted by the algorithm (Figure 5.43(a)). Although point B lies close to A, they are not arbitrarily close to each other. Moreover, the length of edge AB is bounded by the shortest edge of triangle t_0 as it was established in Section 5.4. Figure 5.43(b) shows the refined triangulation produced by the algorithm for an angle parameter $\theta = 20°$.

Algorithm 5.2 describes a version of the Lepp-Delaunay algorithm modified to first process the target triangles with constrained edges.

For the input triangulation shown in Figure 5.42, the algorithm first inserts point A since t_0 is a bad quality triangle with a constrained midsize edge (Figure 5.44(a)). Point B is never inserted and the algorithm will only adapt to the small edges that initially belonged to triangles t_0 and t_1. Figure 5.44(b) shows the refined triangulation for the same angle bound of 20°.

5.6.3 Angles Between Adjoining Constrained Edges

In this section we discuss the minimum angle at which two adjoining segments of the input geometry could meet (we also call these angles *constrained angles*). We study the minimum angle ϕ that maintains our bounds on the insertion radius and therefore ensures the termination of the algorithm. The logic behind a bound on ϕ is related to the scenario in which two adjoining input segments produce constrained edges that have their midpoints *encroaching upon* each other.

The concept of encroachment has been previously employed to limit how close an inserted point can lie to a constrained edge. Ruppert's methodology, for example, applies a limit equal to half the length of the constrained edge [53]. We adapted this limit to match the verification

Algorithm 5.2 Lepp-Delaunay Algorithm (with priority)

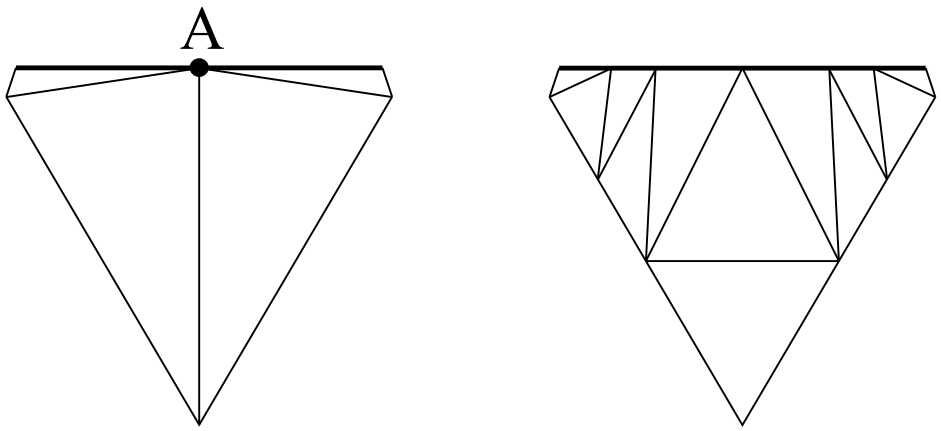
Input: PSLG polygon $\mathcal{P}$, quality parameter ϵ
Output: Constrained Delaunay Triangulation τ of quality defined by ϵ

Construct τ the CDT of $\mathcal{P}$.
Find $S_{\text{target}} \in \tau$, the set of bad quality triangles defined by ϵ
Let $S_{\text{cons}} \subseteq S_{\text{target}}$ be the set of target triangles with constrained edges
for each u in S_{cons} **do**
  if u has longest edge or midsize edge constrained **then**
    Perform constrained Delaunay insertion of midpoint of the constrained edge
  else
    while u remains in τ **do**
      Find Lepp(u), terminal triangles u_i, u_j and terminal edge e_{term} {*triangle u_j can be null for boundary e_{term}*}
      Select point P using Point Selection Strategy
      Perform constrained Delaunay insertion of P into τ
    end while
  end if
  Update S_{target} {*since additional target triangles could have been eliminated*}
end for
for each v in S_{target} **do**
  if v has longest edge or midsize edge constrained **then**
    Perform constrained Delaunay insertion of midpoint of the constrained edge
  else
    while v remains in τ **do**
      Find Lepp(v), terminal triangles v_i, v_j and terminal edge e_{term} {*triangle v_j can be null for boundary e_{term}*}
      Select point P using Point Selection Strategy
      Perform constrained Delaunay insertion of P into τ
    end while
  end if
  Update S_{target} {*since additional target triangles could have been eliminated*}
end for

used by the point selection strategy when selecting the midpoint of constrained edge.

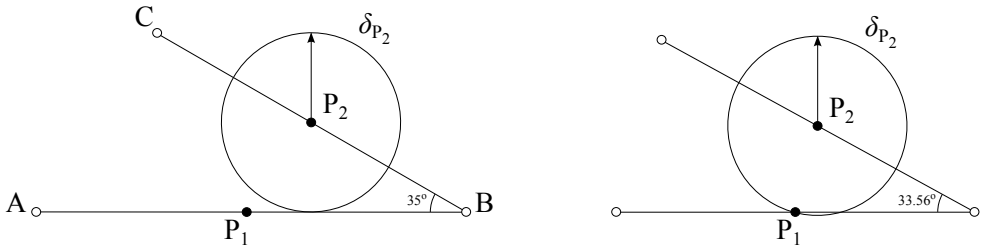
Consider two adjoining constrained edges c_1 and c_2 with midpoints P_1 and P_2. We say that P_1 *encroaches upon* constrained edge c_2 if $d(P_1, P_2) < \text{length}(c_2)/2\sqrt{3}$, that is, P_1 lies within the circle δ_{P_2} of radius $\text{length}(c_2)/2\sqrt{3}$ centered at P_2.

In order to maintain the bounds on the insertion radius, edges c_1 and c_2 should meet at an angle that ensures that P_1 does not encroach upon c_2 nor P_2 upon c_1. Consider $\text{length}(c_1) > \text{length}(c_2)$, so the condition is met when c_1 is tangential to δ_{P_2}. This implies that the Lepp-Delaunay algorithm allows input angles $\phi = 35°$ (Figure 5.45(a)). Furthermore, the closest points P_1 and P_2 happens when $\text{length}(c_1) = \text{length}(c_2)$, and thus $\phi \approx 33.56°$ (Figure 5.45(b)). This represents a tighter bound on the minimum angle at which input segments can meet.



(a) (b)

Figure 5.44: (a) The insertion of point A eliminates the bad quality triangles without creating a new small geometric feature. (b) Refined triangulation with no angles smaller than 20°.



(a) (b)

Figure 5.45: Minimum angle allowed for adjoining constrained edges. Points P_1 and P_2 are the midpoints of edges AB and BC. Circle δ_{P_2} has radius $|BC|/2\sqrt{3}$. (a) $|AB| > |BC|$ and edge AB is tangential to δ_{P_2}. (b) $|AB| = |BC|$ and $d(P_1, P_2) = |BC|/2\sqrt{3}$.

When $\theta > \phi$ the algorithm is not able to improve the angle formed by the constrained edges. In fact, this angle will produce an infinite processing cycle since the algorithm will perform iterative longest-edge bisections trying to eliminate the triangle containing the constrained angle. We say that the algorithm terminates when no two segments meet at an angle less than $\max\{33.5°, \theta\}$.

□

Small angles inherit in the input geometry cannot be improved through Delaunay refinement, and triangulating the domain without creating new small angles is not always possible [59]. Complex strategies have been proposed for the treatment of meshes with small angles. For example, Ruppert [53] uses concentric circular shells to avoid infinite processing cycles produced by edge bisections.

In practice, the Lepp-Bisection algorithm can be used to refine the area inside a small constrained angle. The size and quality of the triangles generated would still be bounded (as discussed in Chapter 4). A new modification, for example, could halt the refinement of the triangle with small constrained angle when the longest-edge bisection produces a new edge with length smaller than a given parameter (e.g. the current smallest edge in the triangulation).

5.7 Termination and Good Grading

A geometric argument of the algorithm's termination is based on the facts that the algorithm improves the triangles as they are processed, and that the algorithm constrains how close together two points can lie and how short an edge can be. Points inserted by the algorithm cannot introduce edges arbitrarily short, so the algorithm cannot run indefinitely creating ever smaller triangles.

The following theorem provides the first proof of the algorithm's termination.

Theorem 5.30. *Given a PSLG $\mathcal{P}$ where no two segments meet an angle less than* $33.5°$ *and an angle parameter $\theta < 25.66°$, the Lepp-Delaunay algorithm terminates producing a constrained Delaunay triangulation τ of $\mathcal{P}$ with internal angles greater than θ and no edge smaller than $l_{\min}/\sqrt{3}$, where $l_{\min}$ is the shortest distance between two points of $\mathcal{P}$.*

Proof. As discussed in Section 5.5, the improvement properties of the algorithm assure that the midpoint selection strategy does not introduce points too close to existing points of the triangulation, nor does it create arbitrarily small edges. For a angle parameter $\theta = 25.66°$, it was demonstrated that in most cases the algorithm produces edges that are longer than $l_{\min}$, the smallest edge in the initial constrained Delaunay triangulation. Moreover, in the scenario in which the algorithm creates an edge smaller than $l_{\min}$, then: (1) the length of the new smallest edge is at least $l_{\min}$; (2) only quality triangles will be associated with the new

smallest edge; and (3) every edge introduced by the algorithm will be longer than the new smallest edge (Lemma 5.29 in Section 5.5.3).

To ensure the termination of the algorithm, we rely on the improved non-cycle assumption in order to avoid any infinite processing cycle, as discussed in Section 3.3.5.

Furthermore, if we relax the angle parameter to $\theta = 22.24°$, we can assure that no infinite processing cycles will be produced during refinement (Theorem 3.14). This angle parameter maintains the same improvement properties that ensure the algorithms termination without having to deal with the verifications of the non-cycle assumption, guaranteeing a triangulation with no angles smaller than θ and no edges smaller than $0.66l_{\min}$.

□

In the rest of this section we develop standard formal proofs for the algorithm's termination, size-optimality and good grading, which together formalize the proof Theorem 5.30. The proofs are based on the methodology introduced by Ruppert [53] for Delaunay-based mesh refinement algorithms, later revisited by Shewchuk [60, 62] and by Cheng *et al.* [9].

5.7.1 Local Feature Size

The demonstration is based on the fact that the spacing of points at any location in the output triangulation is within a constant factor of the sparsest possible spacing. After inserting a point, the nearest point in the triangulation lies at a distance at least a constant fraction of its local feature size. Therefore, the density of the output triangulation will be bounded by the geometry of the input triangulation.

The spacing of points is represented by the local feature size function, which has the property to be proportional to the sparsest possible spacing.

Definition 5.31. **[53]** Given a PSLG $\mathcal{P}$, the *local feature size* at any point p, $\text{lfs}(p)$, correspond to the radius of the smallest disk centered at p that intersects two non-incident vertices or segments of $\mathcal{P}$.

Figure 5.46 illustrates the local feature size function for different points.

Function $\text{lfs}(\cdot)$ is 1-Lipschitz, it is continuous and its gradient has magnitude smaller than 1 [62]. The following lemma is key to bound the grading of elements in the triangulation.

Lemma 5.32. [53] *Given a PSLG $\mathcal{P}$, and any two points P and Q in the plane,*

$$\text{lfs}(Q) \leq \text{lfs}(P) + d(P, Q)$$

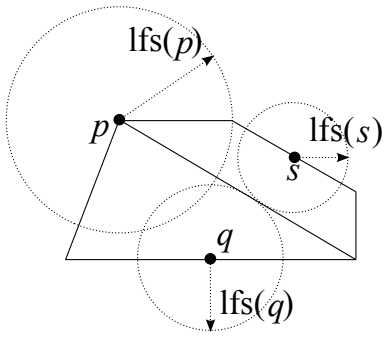


Figure 5.46: Examples of local feature size for points p, q and s.

Proof. By definition, the disk of radius $r = \text{lfs}(P)$ centered at P must intersect two non-incident features in $\mathcal{P}$. The disk of radius $r' = r + d(P, Q)$ centered at Q contains the prior disk, thus intersects the same two features. Therefore, $\text{lfs}(Q) \leq r'$, and we have

$$\text{lfs}(Q) \leq r' = r + d(P, Q) = \text{lfs}(P) + d(P, Q).$$

□

5.7.2 Proof of Termination

The key idea behind the termination of the algorithm is that the descendant of a point has an insertion radius lower bounded by its own insertion radius, that is, the insertion radius cannot be arbitrarily small. This scenario was initially explored in Sections 5.3 and 5.4 where we respectively set the bounds for the insertion radius of a point, and the relationship between the insertion radius of a point and that of its parent.

Following Ruppert's methodology, we formally formulate this relationship using the local feature size function.

Lemma 5.33. *Consider an angle parameter $\theta < 25.66°$, point Q the midpoint of a terminal edge, and point P the parent of Q. Then $r_Q \geq \lambda r_P$, where $\lambda > \frac{1}{\sqrt{3}}$ is a constant determined by θ.*

Proof. Since Q is the midpoint of a terminal edge, the ratio between the insertion radius of Q and that of its parent is maximized for the terminal triangle with largest angle $\gamma = 120°$. For a terminal triangle with smallest angle $\alpha = 25.66°$ the ratio between the insertion radius of Q and P is defined by $\frac{1+\cos 120°}{2 \sin 25.66°} = \frac{1}{\sqrt{3}}$, the ratio between the smallest edge of a target terminal triangle and the distance from the midpoint of its longest edge to the circumcircle. □

Lemma 5.33 establishes a theoretical limit on the rate in which the insertion radius evolves as new points are inserted. As discussed in Section 5.5, a configuration of triangles maximizing the decreasing rate of insertion radius (by half) can only happen for an angle

parameter $\theta = 30°$: the algorithm selects a bad quality terminal triangle where $\gamma = 120°$ and the smallest angle $\alpha = 30°$.

As discussed in Section 5.5.2, for any triangle with smallest angle $\alpha < 14.48°$ it is guaranteed that the insertion radius of any point inserted by the algorithm is greater than the insertion radius of its parent (Lemma 5.23). If we guarantee that the insertion radius cannot decrease during refinement process, we can prove that the algorithm terminates.

To facilitate the analysis, we set the angle parameter to 14.48° to guarantee than the insertion radius never decreases. We later generalize the proofs to show that the angle parameter can be relaxed to 25.66° if we allow smaller details in the output triangulation.

The following lemma, adapted from [60], formalizes this idea.

Lemma 5.34. *Consider $l_{\min}$ the shortest distance between two points of the input PSLG $\mathcal{P}$, and no two segments of $\mathcal{P}$ meet an angle less than 33.5°. Given an angle tolerance parameter $\theta < 14.48°$, the Lepp-Delaunay algorithm terminates and the output triangulation has no edge shorter than $l_{\min}$.*

Proof. The proof follows from Lemma 5.23 in Section 5.5. □

Although an angle parameter θ between 14.48° and 25.66° could produce a slight decrease on the insertion radius in some scenarios, we proved in Sections 5.5.1 and 5.5.3 that this could only happen once. Therefore, there is not a pessimistic scenario where the algorithm continuously decreases the insertion radius of new points. The following lemma presents a relaxed proof for the algorithm's termination.

Lemma 5.35. *Consider $l_{\min}$ the shortest distance between two points of the input PSLG. Given an angle tolerance parameter $\theta < 25.66°$, the Lepp-Delaunay algorithm terminates and the output constrained Delaunay triangulation has no edge shorter than $\lambda l_{\min}$.*

Proof. The proof follows from Lemma 5.33 and the worst-case scenario discussed in Section 5.5.3 (Proposition 5.27), when terminal triangle has a largest angle 120°. For example, an angle parameter of 25.66° ensures triangulations with edges of at least $l_{\min}/\sqrt{3}$. □

In practice, the algorithm is observed to terminate for angle parameters of up to 30°, in which case the Lepp-Delaunay algorithm terminates producing a CDT with edges longer than $l_{\min}/2$. The same methodology used in Section 5.5.3 can be extended to proof this case. Termination is possible for angle parameters above 30°, but not guaranteed.

5.7.3 Proof of Good Grading and Size Optimality

The grading refers to the variation of triangle sizes over short distances, a well-graded triangulation should offer a smooth transition from small triangles to big triangles without

producing small angles.

A key piece of this proof is to guarantee that the length of an edge of the output triangulation will be proportional to the local feature size of its endpoints. We define the relationship between the local feature size and the insertion radius of a point Q, $D_Q = \text{lfs}(Q)/r_Q$.

This relationship represents the density of points near Q weighted by the local feature size after point Q is inserted. A small value of D_Q represents an insertion radius proportionally bigger than its local feature size. For this reason, smaller values of D_Q are desirable.

In order to establish a bound on the rate in which D_Q could increase during refinement, we use the following lemma to establish the relationship between D_Q and $D_{\hat{Q}}$, where $\hat{Q}$ is the parent point of Q.

Lemma 5.36. *Consider Q the point selected for insertion by the Lepp-Delaunay algorithm, and $P = \hat{Q}$ its parent. It holds that $D_Q \leq 1 + D_P/\lambda$.*

Proof. By Lemma 5.32 we know that $\text{lfs}(Q) \leq \text{lfs}(P) + d(P, Q)$. By Lemma 5.33, we know that $r_Q \geq \lambda r_P$. And by definition, $r_Q \geq d(P, Q)$. Then

$$\begin{aligned} \text{lfs}(Q) &\leq \text{lfs}(P) + r_Q \\ &= D_P r_P + r_Q \\ &\leq \frac{D_P}{\lambda} r_Q + r_Q \\ &\leq D_Q \leq 1 + D_P/\lambda. \end{aligned}$$

□

The following lemma adapts Ruppert's main result to our analysis of the Lepp-Delaunay algorithm, establishing the relationship between insertion radius and local feature size for any point in the triangulation.

Lemma 5.37. *Suppose an angle parameter θ such that any triangle with angles smaller than θ is selected for refinement, then there exists a constant $\epsilon \geq 1$ such that for any point Q inserted by the algorithm, $D_Q \leq \epsilon$. Therefore, the insertion radius of a point is proportional to its local feature size.*

As discussed by Shewchuk [58,60], the proof of the lemma follows from Lemmas 5.33 and 5.36. For the case of the Lepp-Delaunay algorithm, the lemma is satisfied when the angle parameter θ is smaller than 25.66°.

In practice the algorithm produces well graded triangulations for angle parameters greater than 25.66°. Rivara and Palma [49] performed and extensive study on the implementation

issues and the practical behavior of the Lepp-Delaunay algorithm with angle parameters of up to 28°.

The following theorem establishes the bound on the distance between points of the output triangulation.

Theorem 5.38. *For any point Q of the output triangulation produced by the Lepp-Delaunay algorithm, the distance to its nearest neighbor N is greater than $\frac{\text{lfs}(Q)}{\epsilon+1}$.*

Proof. By Lemma 5.37, $\frac{\text{lfs}(Q)}{r_Q} \leq \epsilon$ for any point Q fo the triangulation. Then,

- if Q was inserted after N, the distance between the two points is greater than $r_Q \geq \frac{\text{lfs}(Q)}{\epsilon}$;
- if N was inserted after Q,
$$d(N,Q) \geq r_N \geq \frac{\text{lfs}(N)}{\epsilon}.$$

By Lemma 5.32, $\text{lfs}(N) + d(N,Q) \geq \text{lfs}(Q)$, and

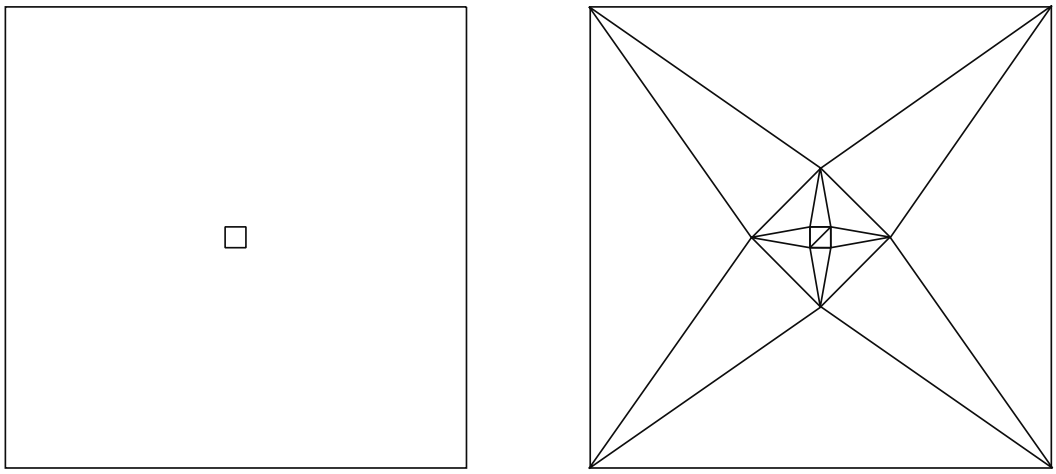
$$d(N,Q) \geq \frac{\text{lfs}(N)}{\epsilon}$$
$$d(N,Q) \geq \frac{\text{lfs}(Q) + d(N,Q)}{\epsilon}.$$

It follows that $d(N,Q) \geq \frac{\text{lfs}(Q)}{\epsilon+1}$. □

The proof of size-optimality provided by Ruppert [53] is based on the results of Theorem 5.38. Additionally, Mitchell [28] provided new bounds on the number of elements, called *cardinality*, of a triangulation with bounded minimum angle. The following theorem summarizes the results of Mitchell.

Theorem 5.39. [28] *Consider $\text{lfs}_\tau(Q)$ the local feature size at point Q with respect to a triangulation τ of a PSLG. Also consider $\text{lfs}(Q)$ the local feature size at point Q with respect to the input PSLG. Given an angle parameter θ such that τ has angles greater than θ, suppose that there exists a constant $K_1 \geq 1$ such that for every point Q, $K_1\text{lfs}_\tau(Q) \geq \text{lfs}(Q)$. Then, the number of elements of τ is less than K_2 times the number of elements of any triangulation of the same PSLG with smallest angle θ, where $K_2 = O(K_1^2/\theta)$.*

Theorem 5.38, along with Lemmas 5.23 and 5.34, can be used to show that meshes generated by the Lepp-Delaunay algorithm satisfy the precondition of Theorem 5.39. Furthermore, Lemma 5.29 and Theorem 5.30 can also be used to satisfy Mitchell's precondition assuming that new small edges do not affect the performance algorithm.



(a) (b)

Figure 5.47: Input geometry for the first grading test. (a) Larger square is 22 times bigger than the small one. (a) An optimal triangulation of the input with angles greater than 20°.

This result proves that the Lepp-Delaunay algorithm produces size-optimal triangulations, with cardinality within a constant factor of the best possible mesh satisfying the parameter θ. A complete proof of Mitchell's theorem can be found in [28].

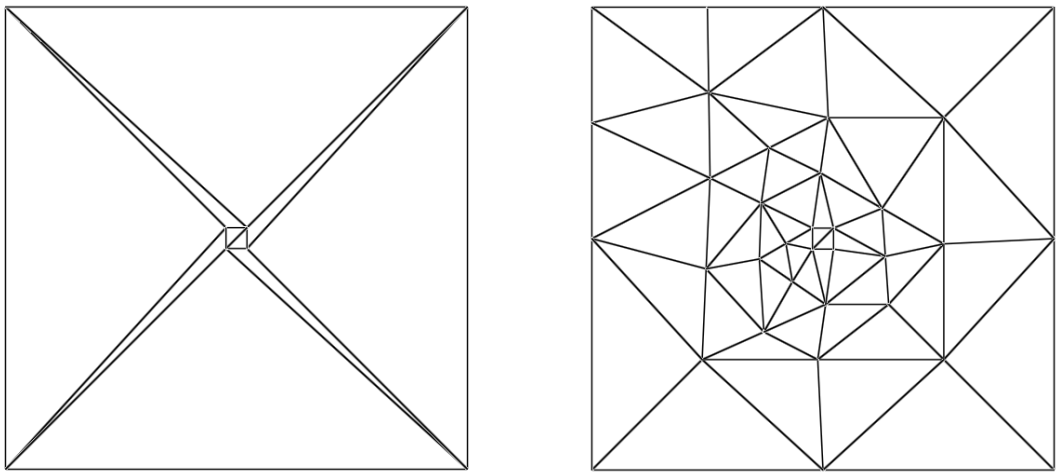
5.7.4 A Grading Test Case

We perform an empirical evaluation of the Lepp-Delaunay algorithm's grading capabilities following the benchmark proposed by Ruppert [53]. The goal is to evaluate how the algorithm adapts to small features of the input while using as few triangles as possible. This benchmark indirectly evaluates the constant factor associated with the cardinality of the triangulation compared to an optimal triangulation of the same input.

The class of inputs used in this evaluation is defined by two squares, a small square of side 1 centered inside a larger square of side d. Figure 5.47(a) illustrates an instance of this class where $d = 22$. A manually generated "optimal" triangulation for an angle parameter $\theta = 20°$ is illustrated in Figure 5.47(b). We attached an isosceles triangle with smallest angle 20° opposite each edge of the small square. The star-shaped construct was then closed with four more triangles, creating an internal square with a 45° rotation. These steps were repeated the large square of side d was created.

Figure 5.48(a) illustrates the initial constrained Delaunay triangulation of the input, containing initial angles as small as 2.6°. The final refined triangulation for the same angle bound is illustrated in Figure 5.48(b). The triangulation produced by the algorithm has about 3 times as many points as the optimal triangulation.

In Figure 5.49(a) we show the triangulation produced by the algorithm for an input with $d = 148$ and an angle parameter of $\theta = 20°$. Compared to the optimal triangulation, the



(a) (b)

Figure 5.48: Performance of the Lepp-Delaunay algorithm for $d = 22$. (a) Initial constrained Delaunay triangulation. (b) Refined triangulation with angles greater than 20°.

algorithm produces about 3.3 times as many points. Further experimentation with different values of d showed, on average, a factor of 3.4 times the size of the optimal triangulation (for the same grading test Ruppert's algorithm produced a factor of 4 [53]).

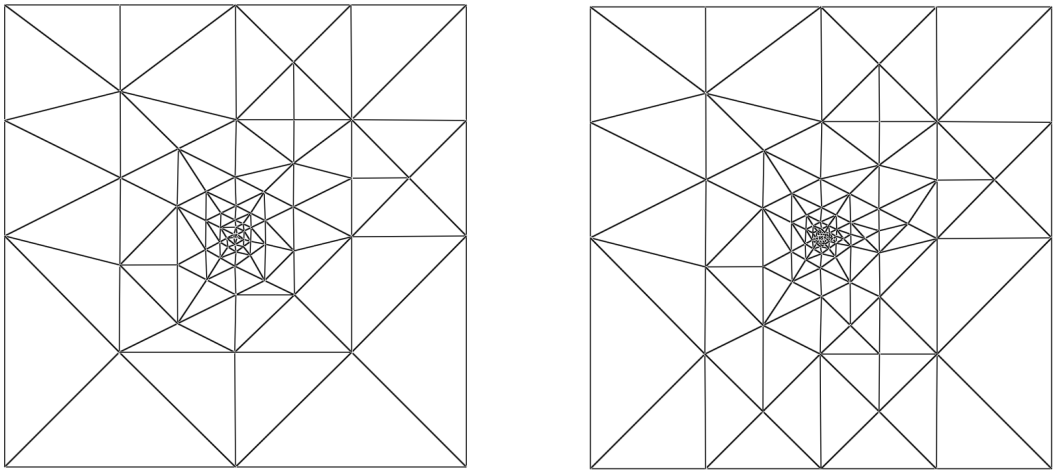
In Figure 5.49(b) we forced the algorithm to improve the triangulation so that no internal angles are smaller than 30°. It is observed that the algorithm maintains the good grading while adapting the triangulation to the small features at the center of the square. These tests provide empirical evidence that the Lepp-Delaunay algorithm performs well in practice and according to the theory.

5.8 Conclusions

The Lepp-Delaunay algorithm is a fast and easy to implement technique for quality mesh generation. In this chapter we presented a detailed geometrical analysis of the Lepp-Delaunay algorithm and the terminal-edge midpoint strategy. By using a case analysis methodology covering all the possible scenarios, we show that for every triangle there exists a small bounded area where the algorithm inserts new points. These results translate into a bound on the distance among points and the length of the edges produced by the algorithm.

The geometrical improvement properties guarantee that the algorithm performs well and terminates for angle parameters of up to 25.66°, in practice the algorithms exhibits good grading for angle parameters of up to 30°.

Using Ruppert's methodology for the analysis of Delaunay refinement, we proved that the Lepp-Delaunay algorithm is guaranteed to terminate and produce well-graded, size-optimal triangulations for an angles of up to 25.66°.



(a) (b)

Figure 5.49: Performance of the Lepp-Delaunay algorithm for $d = 148$. (a) Refined triangulation with angles greater than 20°. (b) Refined triangulation with angles greater than 30°.

The propagated refinement helps to improve the local grading of the triangulation. The Delaunay-insertion of points contribute to reduce the extent of the propagating paths as more quasi-equilateral triangles populates the triangulation as it was discussed in Chapter 4. The behavior of the propagation remained the same as with the Lepp-Bisection algorithm: the propagation tends to affect only a small number of triangles as more terminal triangles appear.

Using the proper data structures, an implementation of the Lepp-Delaunay algorithm consistently takes $O(n \log n + N)$ time in practice, where n is the number of points in the initial PSLG and $N \geq n$ is the number of points in the output triangulation. The term $n \log n$ covers the cost associated with the construction of the initial constrained Delaunay triangulation. The term N covers the cost of refinement, since points are inserted in constant time.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis focused on the analysis of two refinement algorithms based on the longest-edge strategy, Lepp-Bisection and Lepp-Delaunay algorithms, for 2-dimensional triangulation refinement.

The fundamental operations, longest-edge bisection and longest-edge propagation, introduce many favorable properties. The simplicity of these operations facilitates a fast refinement of triangulations through local operations. These algorithms produce provably good meshes and are robust and easy to implement, especially compared to Delaunay refinement algorithms with circumcenter strategies.

We performed an exhaustive geometrical analysis of the longest-edge bisection of triangles, the evolution of propagated refinement, and the insertion of new points. Our main contribution are the theoretical guarantees regarding the quality and size of the refined triangulations.

Chapter 4 provided new results on the complexity of the Lepp-Bisection algorithm. We showed that the longest-edge bisection strategy inserts a constant number of new points in order to produce conforming triangles. We also showed that the number of triangles in a propagating path quickly convergences to two as the triangles are iteratively refined. Consequently a triangle is refined in constant time, inserting a constant number of points. The Lepp-Bisection algorithm is suitable for local and adaptive mesh refinement, with an asymptotically optimal cost. Our empirical study of the algorithm also showed that triangles are refined in constant time. Moreover, the experiments showed that the algorithm improves both the quality of the triangles (tending to produce quasi-equilateral triangles) and the grading of the triangulation.

Chapter 5 focused on the Lepp-Delaunay algorithm. We performed an analysis of the point selection strategy (midpoint of a terminal edge) over Delaunay triangulations, providing

a complete geometrical characterization of the new edges created by the algorithm. We showed that the insertion radius is bounded by the geometry of the terminal triangles: the insertion radius is at least half the length of the smallest edge of a triangle, or at least half its circumradius. We also showed that the insertion radius of new points is at most a factor of two smaller than the insertion radius of its predecessors, and showed that for an angle parameter of 25.66° new edges are at least $l_{\min}/\sqrt{3}$, where $l_{\min}$ is the smallest feature in the input geometry. We also proposed an improved point selection strategy for constrained edges that reduces the number of points inserted near constrained edges, and allows the algorithm to process input geometries with constrained angles as small as 35°. Finally, our study showed that the Lepp-Delaunay algorithm produces size-optimal triangulations with internal angles between 25.66° and 128.68°.

6.2 Further Research

An immediate theoretical direction of this research should explore the longest-edge algorithms for the refinement of 3-dimensional triangulations. Intuitively, the theoretical results of this thesis could be generalized to the 3-dimensional scenario. Empirical experimentation with Lepp-Bisection and Lepp-Delaunay suggests that the properties hold in these scenarios [48, 50, 51].

Following the methodology used during the analysis of the Lepp-Bisection algorithm, an initial study of the longest-edge bisection of tetrahedra would provide the foundation for the rest of the analysis. We consider the following issues:

- How to define a class of "quasi-equilateral" tetrahedron
- What are the geometrical properties of the longest-edge bisection of tetrahedra
- How to define a classification of tetrahedra based on the behavior of iterative bisections
- How the iterative bisection might converge to quasi-equilateral tetrahedra

Once the cost of processing an element (in terms of number of new points) is established, the study should focus on 3-dimensional propagating paths and the generation of "terminal" tetrahedra.

An analysis of the Lepp-Delaunay algorithm for 3-dimensional mesh refinement should consider the study of Delaunay terminal tetrahedra and, similar to the 2-dimensional scenario, how the longest-edge propagation implicitly defines bounds on the position where a new point is inserted. A study of the insertion radius would define a frame of reference to apply the standard methodology used in the analysis of Delaunay refinement algorithms.

A different direction of future research should further explore the robustness of the longest-edge refinement algorithms, expanding upon the current connection to dyadic data. An

evaluation of the performance of these robust implementations must be performed to validate the theoretical results.

In a practical direction, there is room for new point selection strategies for the Lepp-Delaunay algorithm. This strategies should aim to reduce the number of points inserted by the algorithm. For example, Üngör [70] proposed a selection strategy based on off-centers (instead of the classical circumcenter), or Rivara and Calderon [44], who proposed a selection strategy based on centroids of terminal triangles (instead of midpoints).

We also propose to study additional strategies for the refinement of meshes with small angles using the Lepp-Delaunay algorithm. Small angles inherit in the input geometry cannot be improved through Delaunay refinement, and triangulating the domain without creating new small angles is not always possible [59]. Cheng *et al.* [9] proposed the use of these two rather complex modifications to Ruppert's algorithm: *concentric circular shells* for segment splitting (initially proposed in [53]), and accept triangles with constrained angles when they become isosceles (initially proposed in [26]). Future studies of the Lepp-Delaunay algorithm should extend the guarantees presented in this thesis for such scenario.

Further studies of the advantages of the longest-edge propagation in the performance of the Lepp-Delaunay algorithm are of both theoretical and practical interest. To this end, we propose to measure the construction time of a quality triangulation using the Lepp-Delaunay algorithm, and compare it to the construction time of the classical Delaunay algorithm using the points of quality triangulation. The goal of this experiment is to evaluate if the propagation strategy naturally produces optimal insertion sequences during the construction of a Delaunay triangulation.

Finally, further empirical evaluation of the Lepp-Delaunay algorithm should compare the performance of the algorithm to other well-known refinement techniques such as Ruppert's algorithm.

Bibliography

[1] Andrew Adler. On the Bisection Method for Triangles. *Mathematics of Computation*, 40:571–571, 1983.

[2] Ángel Plaza, Sergio Falcón, and José P. Suárez. On the non-degeneracy property of the longest-edge trisection of triangles. *Applied Mathematics and Computation*, 216(3):862 – 869, 2010.

[3] Brenda S. Baker, Eric Grosse, and Conor S. Rafferty. Nonobtuse triangulation of polygons. *Discrete & Computational Geometry*, 3:147–168, 1988.

[4] Timothy Baker. Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation. *Engineering with Computers*, 5:161–175, 1989. 10.1007/BF02274210.

[5] Carlos Bedregal and Maria-Cecilia Rivara. A study on size-optimal longest edge refinement algorithms. In Xiangmin Jiao and Jean-Christophe Weill, editors, *Proc. of the 21st Int. Meshing Roundtable*, pages 121–136. Springer Berlin Heidelberg, 2013.

[6] Carlos Bedregal and Maria-Cecilia Rivara. Longest-edge algorithms for size-optimal refinement of triangulations. *Computer-Aided Design*, 46:246–251, 2014.

[7] Marshall Bern, David Eppstein, and John Gilbert. Provably good mesh generation. *Journal of Computer and System Sciences*, 48(3):384–409, June 1994.

[8] Houman Borouchaki and Paul Louis George. Aspects of 2-D Delaunay mesh generation. *Int. Journal for Numerical Methods in Engineering*, 40(11):1957–1975, 1997.

[9] Siu-Wing Cheng, Tamal K. Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. Chapman and Hall / CRC computer and information science series. CRC Press, 2013.

[10] L. P Chew. Building voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH, USA, 1990.

[11] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Adaptive set intersections, unions, and differences. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 743–752, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[12] RexA. Dwyer. A faster divide-and-conquer algorithm for constructing Delaunay triangulations. *Algorithmica*, 2(1-4):137–151, 1987.

[13] Vladmir Estivill-Castro and Derick Wood. A survey of adaptive sorting algorithms. *ACM Computing Surveys*, 24(4):441–476, 1992.

[14] C. O. Frederick, Y. C. Wong, and F. W. Edge. Two-dimensional automatic mesh generation for structural analysis. *Int. Journal for Numerical Methods in Engineering*, 2(1):133–144, 1970.

[15] Leonidas J. Guibas, Donald E. Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and voronoi diagrams. *Algorithmica*, 7(1-6):381–413, 1992.

[16] Claudio Gutierrez, Flavio Gutierrez, and Maria-Cecilia Rivara. Complexity of the bisection method. *Theoretical Computer Science*, 382(2):131–138, 2007.

[17] Antti Hannukainen, Sergey Korotov, and Michal Krek. On global and local mesh refinements by a generalized conforming bisection algorithm. *Journal of Computational and Applied Mathematics*, 235(2):419 – 436, 2010. Special Issue on Advanced Computational Algorithms.

[18] Reiner Horst. On generalized bisection of n-simplices. *Mathematics of Computation*, 66:691–698, 1997.

[19] Mark T. Jones and Paul E. Plassman. Computational results for parallel unstructured mesh computations. Technical report, University of Tennessee, Knoxville, TN, USA, 1994.

[20] Mark T Jones and Paul E Plassmann. Adaptive refinement of unstructured finite-element meshes. *Finite Elements in Analysis and Design*, 25(1-2):41–60, 1997.

[21] Baker Kearfott. A Proof of Convergence and an Error Bound for the Method of Bisection in R^n. *Mathematics of Computation*, 32, 1978.

[22] D.T. Lee and B.J. Schachter. Two algorithms for constructing a Delaunay triangulation. *Int. Journal of Computer and Information Sciences*, 9(3):219–242, 1980.

[23] Alberto Márquez, Auxiliadora Moreno-González, Ángel Plaza, and José P. Suárez. The seven-triangle longest-side partition of triangles and mesh quality improvement. *Finite Elements in Analysis and Design*, 44(12–13):748 – 758, 2008.

[24] Elefterios A. Melissaratos and Diane L. Souvaine. Coping with inconsistencies: a new approach to produce quality triangulations of polygonal domains with holes. In *Proc. of the 8th Annual Symposium on Computational Geometry*, SCG '92, pages 202–211, New York, NY, USA, 1992. ACM.

[25] Gary L. Miller. A time efficient Delaunay refinement algorithm. In *Proc. of the 15th annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 400–409, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[26] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. When and why Ruppert's algorithm works. In *Proc. of the 12st Int. Meshing Roundtable*, pages 91–102, Santa Fe, NM, Sept 2003. Sandia.

[27] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, and Noel Walkington. A Delaunay based numerical method for three dimensions: Generation, formulation, and partition. In *Proc. of the 27th Annual ACM Symposium on Theory of Computing*, STOC '95, pages 683–692, New York, NY, USA, 1995. ACM.

[28] Scott A. Mitchell. Cardinality bounds for triangulations with bounded minimum angle. In *Proc. of the 6th Canadian Conference on Computational Geometry*, pages 326–331. University of Saskatchewan, 1994.

[29] S. N. Muthukrishnan, P. S. Shiakolas, R. V. Nambiar, and K. L. Lawrence. Simple algorithm for the adaptive refinement of three dimensional problems with tetrahedral meshes. *AIAA Journal*, 33(5):928–932, 1995.

[30] R. V. Nambiar, R. S. Valera, K. L. Lawrence, Robert B. Morgan, and David Amil. An algorithm for adaptive refinement of triangular element meshes. *Int. Journal for Numerical Methods in Engineering*, 36(3):499–509, 1993.

[31] L. Paul Chew. Constrained Delaunay triangulations. *Algorithmica*, 4(1-4):97–108, 1989.

[32] A. Plaza and G.F. Carey. Local refinement of simplicial grids based on the skeleton. *Applied Numerical Mathematics*, 32(2):195 – 218, 2000.

[33] Ángel Plaza, Sergio Falcón, José P. Suárez, and Pilar Abad. A local refinement algorithm for the longest-edge trisection of triangle meshes. *Mathematics and Computers in Simulation*, 82(12):2971 – 2981, 2012.

[34] Ángel Plaza and Maria-Cecilia Rivara. Mesh refinement based on the 8-tetrahedra longest-edge partition. In *Proc. of the 12th Int. Meshing Roundtable*, pages 67–78, 2003.

[35] Ángel Plaza, José P. Suárez, Miguel A. Padrón, Sergio Falcón, and Daniel Amieiro. Mesh quality improvement and other properties in the four-triangles longest-edge partition. *Computer Aided Geometric Design*, 21(4):353–369, April 2004.

[36] M. Rivara and P. Inostroza. A discussion on mixed (longest side midpoint insertion) Delaunay techniques for the triangulation refinement problem. In *Proc. of the 4th Int. Meshing Roundtable*, pages 335–346, 1995.

[37] M.-C. Rivara, N. Hitschfeld, and B. Simpson. Terminal-edges Delaunay (small-angle based) algorithm for the quality triangulation problem. *Computer-Aided Design*, 33(3):263 – 277, 2001.

[38] Maria-Cecilia Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *Int. Journal for Numerical Methods in Engineering*, 20(4):745–756, 1984.

[39] Maria-Cecilia Rivara. Design and data structure of fully adaptive, multigrid, finite-element software. *ACM Trans. on Mathematical Software*, 10(3):242–264, 1984.

[40] Maria-Cecilia Rivara. New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations. *Int. Journal for Numerical Methods in Engineering*, 40(18):3313–3324, 1997.

[41] Maria-Cecilia Rivara. A study on Delaunay terminal edge method. In *Proc. of the 15th Int. Meshing Roundtable*, pages 543–562, 2006.

[42] Maria-Cecilia Rivara. Lepp-Bisection algorithms, applications and mathematical properties. *Applied Numerical Mathematics*, 59(9):2218–2235, 2009.

[43] Maria-Cecilia Rivara and Carlo Calderon. Lepp terminal centroid method for quality triangulation: a study on a new algorithm. In *GMP'08: Proc. of the 5th Int. Conference on Advances in Geometric Modeling and Processing*, pages 215–230, Berlin, Heidelberg, 2008. Springer-Verlag.

[44] Maria-Cecilia Rivara and Carlo Calderon. Lepp terminal centroid method for quality triangulation. *Computer-Aided Design*, 42(1):58–66, 2010.

[45] Maria-Cecilia Rivara and Nancy Hitschfeld. LEPP - Delaunay algorithm: a robust tool for producing size-optimal quality triangulations. In *Proc. of the 8th Int. Meshing Roundtable*, pages 205–220, 1999.

[46] Maria-Cecilia Rivara and Nancy Hitschfeld. LEPP - Delaunay algorithm: a robust tool for producing size-optimal quality triangulations. In *Proc. of the 8th Int. Meshing Roundtable*, pages 205–220, 1999.

[47] María-Cecilia Rivara and Gabriel Iribarren. The 4-triangles longest-side partition of triangles and linear refinement algorithms. *Mathematics of Computation*, 65(216):1485–1502, 1996.

[48] Maria-Cecilia Rivara and Cristian Levin. A 3-d refinement algorithm suitable for adaptive and multi-grid techniques. *Communications in Applied Numerical Methods*, 8(5):281–290, 1992.

[49] Maria-Cecilia Rivara and M Palma. New LEPP algorithms for quality polygon and volume triangulation: implementation issues and practical behavior. *Trends in Unstructured Mesh Generation*, AMD 220:1–8, 1997.

[50] Maria-Cecilia Rivara, Pedro Rodriguez, Rafael Montenegro, and Gaston Jorquera. Multithread parallelization of Lepp-Bisection algorithms. *Applied Numerical Mathematics*, 62(4):473–488, 2012.

[51] Pedro A. Rodríguez and Maria-Cecilia Rivara. Multithread Lepp-Bisection algorithm for tetrahedral meshes. In Josep Sarrate and Matthew L. Staten, editors, *Proc. of the 22st Int. Meshing Roundtable*, pages 525–540. Springer, 2013.

[52] Ivo G. Rosenberg and Frank Stenger. A Lower Bound on the Angles of Triangles Constructed by Bisecting the Longest Side. *Mathematics of Computation*, 29:390–390, 1975.

[53] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548 – 585, 1995.

[54] Edward A. Sadek. A scheme for the automatic generation of triangular finite elements. *Int. Journal for Numerical Methods in Engineering*, 15(12):1813–1822, 1980.

[55] Stefan Schirra. Chapter 14 - robustness and precision issues in geometric computation*. In J.-R. SackJ. Urrutia, editor, *Handbook of Computational Geometry*, pages 597 – 632. North-Holland, Amsterdam, 2000.

[56] Robert Schneiders. Quadrilateral and Hexahedral Element Meshes. In Joe Thompson, Bharat Soni, and Nigel Weatherill, editors, *Handbook of Grid Generation*. CRC Press, 1999.

[57] Michael Ian Shamos and Dan Hoey. Closest-point problems. *54th Annual Symposium on Foundations of Computer Science*, 0:151–162, 1975.

[58] Jonathan Richard Shewchuk. Delaunay refinement mesh generation. Technical Report CMU-CS-97-137, School of Computer Science, Carnegie Mellon University, 1997.

[59] Jonathan Richard Shewchuk. Mesh generation for domains with small angles. In Siu-Wing Cheng, Otfried Cheong, Pankaj K. Agarwal, and Steven Fortune, editors, *Proc. of the 16th Annual Symposium on Computational Geometry, Clear Water Bay, Hong Kong, China, June 12-14, 2000*, pages 1–10. ACM, 2000.

[60] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry*, 22(13):21 – 74, 2002. 16th ACM Symposium on Computational Geometry.

[61] Jonathan Richard Shewchuk. Unstructured mesh generation. In Uwe Naumann and Olaf Schenk, editors, *Combinatorial Scientific Computing*, chapter 10, pages 259–298. CRC Press, 2011.

[62] Jonathan Richard Shewchuk. Reprint of: Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry*, 47(7):741–778, 2014.

[63] Bruce Simpson and Maria-Cecilia Rivara. Geometrical mesh improvement properties of Delaunay terminal edge refinement. In Myung-Soo Kim and Kenji Shimada, editors, *Geometric Modeling and Processing - GMP 2006*, volume 4077 of *Lecture Notes in Computer Science*, pages 536–544. Springer Berlin Heidelberg, 2006.

[64] Martin Stynes. On Faster Convergence of the Bisection Method for all Triangles. *Mathematics of Computation*, 35:1195–1195, 1980.

[65] J. P. Suárez, A. Plaza, and G. F. Carey. Propagation of longest-edge mesh patterns in local adaptive refinement. *Communications in Numerical Methods in Engineering*, 24(7):543–553, 2008.

[66] José P. Suárez, Ángel Plaza, and Graham F. Carey. Propagation path properties in iterative longest-edge refinement. In *Proc. of the 12th Int. Meshing Roundtable*, pages 79–90, 2003.

[67] José P. Suárez, Tania Moreno, Pilar Abad, and Ángel Plaza. Convergence speed of generalized longest-edge-based refinement. In Gi-Chul Yang, Sio-long Ao, and Len Gelman, editors, *IAENG Trans. on Engineering Technologies*, volume 229 of *Lecture Notes in Electrical Engineering*, pages 511–522. Springer Netherlands, 2013.

[68] José P. Suárez, Ángel Plaza, and Graham F. Carey. The propagation problem in longest-edge refinement. *Finite Elem. Anal. Des.*, 42(2):130–151, November 2005.

[69] Alper Üngör. Off-centers: A new type of steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. In *LATIN*, pages 152–161, 2004.

[70] Alper Üngör. Off-centers: A new type of steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. *Computational Geometry*, 42(2):109–118, 2009.

[71] Olver Vilca, Maria-Cecilia Rivara, and Claudio Gutierrez. Cálculo de la longitud media de propagacin del lepp (in spanish). In *CSPC-2010: IX Conference of the Peruvian Computing Society*, pages 35–42, 2010.

[72] Chee Yap. Theory of real computation according to egc. In Peter Hertling, Christoph M. Hoffmann, Wolfram Luther, and Nathalie Revol, editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, volume 5045 of *Lecture Notes in Computer Science*, pages 193–237. Springer Berlin Heidelberg, 2008.

[73] Chee K. Yap. Robust geometric computation. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition.*, pages 927–952. Chapman and Hall/CRC, 2004.