Aplicación de la Geometría Computacional en la Reconstrucción 3D Basada en Diagramas de Voronoi

por

Ulises Martínez Rodríguez

Tesis presentada en cumplimiento parcial de los requisitos
para la obtención del grado de
Licenciado en Matemáticas Aplicadas

Benemérita Universidad Autónoma de Puebla, Facultad de
Ciencias Físico Matemáticas

Asesor(es):
Dr. Wuiyebaldo Fermín Guerrero Sánchez
Dra. Blanca Bermúdez Juárez
Dr. Carlos Guillén Galván

México, Puebla, marzo de 2015

Al que lucha y no claudica.

Resumen

¿Cuál es la mejor manera de imprimir una pieza tridimensionalmente, tal que ésta no pierda su forma original pero con un ahorro considerado de material? Una buena ayuda para resolver esta situación se halla en los diagramas de Voronoi, que son las regiones en que se divide el plano (espacio), una especie de mosaico con una propiedad muy particular: si en el plano (espacio) se eligen un conjunto de puntos, el diagrama de Voronoi asigna una región a cada punto, de tal modo que todo lo que está contenido en esa región está más cerca de ese punto que de cualquier otro.

El concepto de diagrama de Voronoi es tan basto como antiguo, por ejemplo Descartes lo consideraba en 1644 e incluso se utilizó para evitar un mayor número de muertes durante el brote de cólera en Londres, en 1854. Tan basto dado que sus aplicaciones pueden darse en Ciencias Sociales, Arquitectura, Cristalografía, Biología, Química o incluso Astronomía.

En la presente tesis se muestran conceptos de la Teoría de Grafos, debido a que el diagrama de Voronoi es una construcción derivada de aquélla, así como también los algoritmos más importantes para generar un diagrama de Voronoi y la triangulación de Delaunay.

Al final se describe un método para, lo que nosotros llamamos, *Voronizar* algunas piezas que se imprimen en el Laboratorio de Sistemas Dinámicos Controlables de la Facultad de Ciencias Físico Matemáticas de la BUAP, a cargo del Dr. Wuiyebaldo Fermín Guerrero Sánchez. Para ello, se empleó el Software Libre `MeshLab`, el cual exhibe la superficie del modelo tridimensional a través de mallas, obteniendo muy buenos resultados.

También se exhibe la importancia que ha sabido ganarse esta teoría, básica en problemas de optimización y diseño computacional. Recordar que “En el espacio de Voronoi, quien mucho abarca tiene los amigos muy lejos”.

Contenido

Agradecimientos

Gracias a la voluntad de muchos, que directa o indirectamente colaboraron en el desarrollo de la presente.

A mi madre, Minerva, principal ejemplo de vida. A mi esposa, Ivette, modelo de responsabilidad y superación.

Al Dr. David Herrera Carrasco, quien evitó que renunciara a este sueño.

A Jason Davies, José R. Gómez, la Canadian Forest Service Publications, la Universidad de Harvard, al periódico alemán Badische Zeitung, la Population Association of America, la Universidad de Florida, Taylor and Francis online, Springer Ed. online, IEE Explore, Yahoo! Japan y Okabe et al.

Finalmente, y no menos importantes, al asesor de esta tesis Dr. W. Fermín Guerrero Sánchez, por otorgarme la oportunidad de salir adelante, su orientación, su paciencia y motivación han sido fundamentales para llevar a cabo este trabajo. Al también asesor de tesis Dr. Carlos Guillén Galván por su contribución invaluable al desarrollo y estructuración del estudio, su seriedad, responsabilidad y rigor académico han influido en mi desarrollo como investigador. Y al Dr. Ricardo Agustín Serrano, quien aportó la idea original de aplicar diagramas de Voronoi para la impresión 3D. Sin ellos no fuese posible la culminación de este esfuerzo.

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Acrónimos y Notación

CAD	Diseño Asistido por Computadora.
CCW	*Counter-ClockWise*, sentido antihorario.
CW	*ClockWise*, sentido del reloj.
TVC	Teselación de Voronoi Centroidal.
TDVC	Triangulación de Delaunay basada en Teselaciones de Voronoi Centroidales.
TVCR	Teselación de Voronoi Centroidal Restringida.
TDVCR	Triangulación de Delaunay basada en teselaciones de Voronoi Centroidales Restringidas.
VCG	Visualización y Computación Gráfica.

Θ	Dada una función $g(n)$, se denota por $\Theta(g(n))$ al conjunto de funciones $\Theta(g(n)) = \{f(n) \mid \exists\ c_1, c_2, n_0 > 0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), n \geq n_0\}$.
O	Dada una función g(n), se denota por $O(g(n))$ al conjunto de funciones $(g(n)) = \{f(n) \mid \exists\ c, n_0 > 0 : 0 \leq f(n) \leq cg(n), n \geq n_0\}$.
Ω	Dada una función g(n), se denota por $\Omega(g(n))$ al conjunto de funciones $\Omega(g(n)) = \{f(n) \mid \exists\ c, n_0 > 0 : 0 \leq cg(n) \leq f(n), n \geq n_0\}$.

Símbolos

∂	Frontera de un conjunto.
G	Grafo.
$D\mathcal{G}(P)$	Grafo de Delaunay.
$\mathcal{G}$	Grafo dual del diagrama de Voronoi.
$\mathcal{T}$	Triangulación.
$\mathcal{V}$	Diagrama de Voronoi.

Capítulo 1

Introducción

1.1 Geometría Computacional

La Geometría Computacional es una rama de las ciencias computacionales que se encarga del diseño y análisis sistemático de algoritmos y estructuras de datos necesarios para la solución eficiente de problemas que implican como entrada y salida objetos geométricos. Sus orígenes nos pueden remontar en siglos (hay quien dice que el primer algoritmo de Geometría Computacional nace cuando una serie de pasos correctos no ambiguos y con un final resuelven un problema geométrico, el precursor: Euclides), pero es sólo a principios de 1970 que Michael Shamos comienza un estudio sistematizado de problemas, obsesionado con la idea de crear una nueva disciplina de las ciencias de computación, a la cual llamó *Geometría Computacional* [1].

La investigación en esta área ha encontrado muchas aplicaciones en la vida real: robótica, reconocimiento de voz y de patrones, diseño gráfico, sistemas de información geográfica, etc.

La principal razón que motivó la elaboración de la presente tesis fue el problema de optimizar material para impresoras 3D, específicamente modelos de fisioterapia, como férulas y de ortopedia, como prótesis de dedos, muñeca o brazo. Es decir, de qué forma podría diseñarse una pieza a imprimir, tal que mantenga el mismo

perfil o figura. Se halló que una manera podría ser *voronizando* tales piezas, en otras palabras, generar una modelo 3D del cual sólo se observan las aristas de la malla que constituye al objeto, usando como base un diagrama de Voronoi tridimensional, o teselación de Voronoi, como se verá más adelante.

El estudio del diagrama de Voronoi, así como de su dual, la triangulación de Delaunay, forma parte de la Geometría Computacional.

1.2 Planteamiento del Problema

Uno de los impedimentos fundamentales para la impresión 3D es la falta de material para efectuarla, debido a su alto costo. Cuando se comenzó a analizar la posibilidad de imprimir partes del cuerpo humano previamente escaneadas, este problema debía solucionarse a la brevedad. Después de sugerirse la aplicaciń del diagrama de Voronoi a los modelos 3D, se buscó un procedimiento para modificar los diseños originales.

En esta tesis se pretende describir un método óptimo y sencillo para modifiacr modelos 3D, basado en algoritmos de construcción para diagramas de Voronoi, mediante el uso del Software Libre `MeshLab`.

1.3 Objetivo

Esta investigación busca contribuir al mejoramiento de diseños tridimensionales y así optimizar el uso de material para la impresión 3D. Esto a través del uso de CADs que:

1. Permitan un manejo sencillo del modelo a *Voronizar*.
2. Consideren la aplicación de los algoritmos más eficientes.
3. Sean aplicables a cualquier tipo de modelos o diseños.

1.4 Estructura de la Tesis

En el primer capítulo se puede observar una visión general de la importancia histórica del diagrama de Voronoi y de la triangulación de Delaunay, ambas estructuras estudiadas dentro de la Geometría Computacional. Se hace énfasis en las aplicaciones que han tenido a lo largo de la historia, principalmente en el área de la salud y ciencias sociales.

En el segundo capítulo se ofrece un panorama teórico preliminar, mediante la introducción de los conceptos básicos de la teoría de grafos, necesarios para la comprensión del concepto de diagrama de Voronoi y triangulación de Delaunay.

En el tercer capítulo pueden hallarse los conceptos básicos de diagrama de Voronoi y triangulación de Delaunay, así como algunas de sus principales propiedades, algoritmos de construcción y la complejidad durante su implementación.

En el cuarto capítulo se brinda un marco general de la optimización basada en diagramas de Voronoi utilizando sus centroides como puntos generadores, a la vez que se muestran algunos algoritmos clásicos.

En el quinto capítulo se muestra un método para la voronización de modelos tridimensionales mediante el uso del Software Libre `MeshLab`, dicho método es una serie de pasos sencillos a través de los cuales es rediseñado cualquier modelo 3D.

Finalmente, en el capítulo seis se presentan los resultados de aplicar el método en una pieza escaneada tridimensionalmente en el Laboratorio de Sistemas Dinámicos Controlables, así como las conclusiones de la investigación y adicionalmente una discusión de los desafíos futuros relacionados con la reconstrucción 3D, basada en diagramas de Voronoi.

1.5 Estado del Arte

Hay muchas formas de dividir el mundo. Las más habituales son las *reales*, o las que, al menos, asumimos como tales: las que el curso de la historia ha ido marcando en forma de fronteras. Son asumidas por el conjunto de personas, países y organizaciones que habitamos el planeta; hasta que un conflicto pone a esas fronteras en disputa, tal es el caso reciente de Ucrania y Rusia y sus intereses cruzados en Crimea o en el este del país, en febrero-marzo de 2014.

Pero hay formas mucho más perfectas de dividir el mundo, formas matemáticas. Una de ellas ha sido explorada por el especialista en visualización de datos Jason Davies, radicado en Londres. Él ha aplicado la matemática al reparto político del territorio, mediante diagramas de Voronoi. Así, divide el planeta en regiones que orbitan sobre el punto en el que está la capital o ciudad más cercana (ver Figura 1.1) [2].

Entonces, según el mapa de Davies, Campeche, Yucatán y Quintana Roo quedarían bajo la influencia de Belice, Chiapas de Guatemala y varios estados de la Unión Americana *volverían* a formar parte del territorio mexicano.

Los Diagramas de Voronoi se utilizan en todos aquellos estudios en los que hay que determinar áreas de influencia, como por ejemplo, en la cobertura hospitalaria, cercanía de estaciones de bomberos o del metro, centros comerciales, control del tráfico aéreo o telefonía móvil. Aunque el campo de aplicación es más basto de lo que puede imaginarse, por ejemplo, en palabras de José Ramón Gómez:

> Las aplicaciones en el área de astronomía, mediante la partición basada en el análisis de métodos del modelo de puntos para la investigación de la estructura espacial de varias poblaciones estelares, o en el área médica podemos hallar la reconstrucción tridimensional computarizada y análisis cuantitativo de neuronas y de haces dendríticos en la corteza cerebral [3].

FIGURA 1.1: El mundo dividido en áreas de Voronoi, en función de las capitales.

1.5.1 Historia

Los diagramas de Voronoi son estructuras geométricas que aparecen con frecuencia en la naturaleza, por esta razón se han redescubierto varias veces a lo largo de la historia: reciben el nombre del matemático ruso Georgy Voronoi, también son conocidos como *Polígonos de Thiessen* (por el meteorólogo estadounidense Alfred

Thiessen), *Teselación*[1] *de Dirichlet* (en honor al matemático alemán Peter Gustav Lejeune Dirchlet), *Celdas de Wigner-Seitz* (por el físico matemático húngaro Eugene Wigner y el físico estadounidense Frederick Seitz) o *Zonas de Brillouin* (por el físico francés León Nicolás Brillouin).

Cronología

- 1664: Descartes afirma que el sistema solar se compone de *vórtices*. Sus ilustraciones muestran una descomposición del espacio en regiones convexas, cada una compuesta de la materia que gira alrededor de una de las estrellas fijas (ver Figura 1.2) [4].
- 1850: Dirichlet formaliza el estudio de formas cuadráticas definidas positivas (forma especial de un Diagrama de Voronoi) en $\mathbb{R}^2$ y $\mathbb{R}^3$ [5].
- 1854: Análisis de John Snow (padre de la epidemiología moderna) del brote de cólera que azotó Londres ese año, mediante el uso del método geográfico (ver Figura 1.3) [6].
- 1908: Voronoi formaliza el estudio de formas cuadráticas definidas positivas en $\mathbb{R}^n$.
- 1909: Boldyrev (Anatoly Kapitonovich) estima las reservas de minerales en un depósito usando la información obtenida de taladros, las regiones de Voronoi son denominadas *áreas de polígonos de influencia*.
- 1911: Thiessen calcula y analiza datos meteorológicos en las estaciones pluviométricas.
- 1927: Paul Niggli desarrolla la teoría en Cristalografía [7].
- 1929: Boris Deloné (Delaunay) es el primero en acuñar el término *Dominio de Dirichlet* y *Región de Voronoi*.

[1] El término teselado hace referencia a una regularidad o patrón de figuras que recubren o pavimentan completamente una superficie plana que cumple con dos requisitos: que no queden huecos y que no se superpongan las figuras.

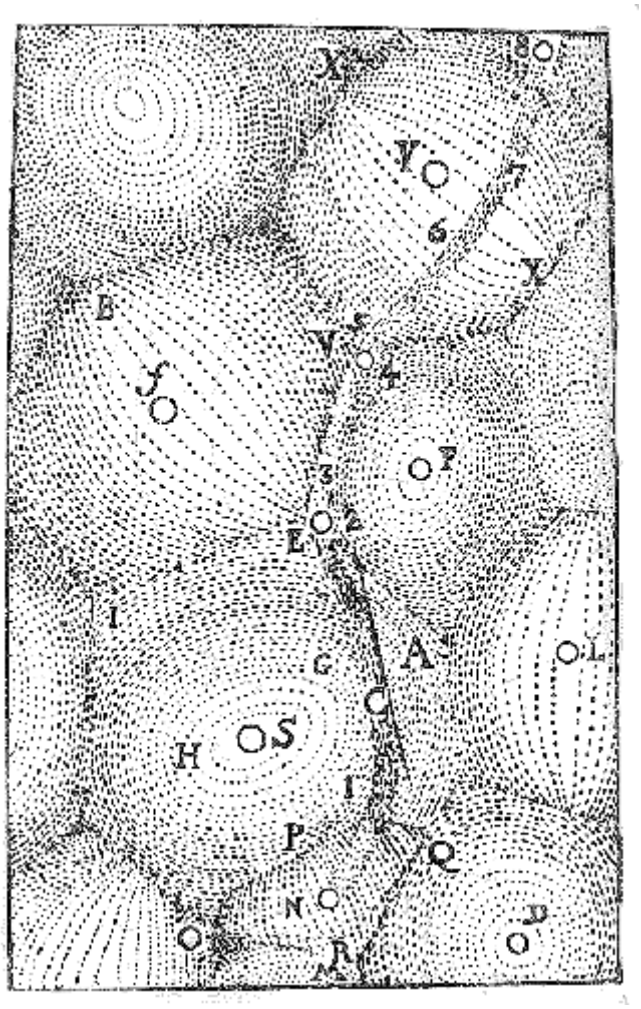


FIGURA 1.2: El concepto de Universo por Descartes.

- 1929: Whitney acuña el término *Polígono de Thiessen*.
- 1933: Wigner y Seitz describen las regiones de Voronoi para los puntos dispuestos en una retícula en el espacio tridimensional [8].
- 1949: Bogue, estudios de mercado.
- 1958: Frank y Kasper, estructuras del átomo [9].
- 1965: Brown estima la intensidad de población de los árboles en un bosque, definiendo una región de Voronoi para un árbol individual, llamándola el *área potencialmente disponible* (APA) de un árbol [10].
- 1985: Hoofd et al. Definió las regiones de Voronoi con respecto a los centros de los capilares en secciones de tejido [11].

FIGURA 1.3: Análisis del brote de cólera en Londres por John Snow, 1854.

- 1987: Icke y van de Weygaert, partición del espacio por medio de un proceso de mosaico de Voronoi [12].

1.5.2 Aplicaciones Históricas

A pesar de que gran parte del desarrollo y muchas de las aplicaciones iniciales ocurrieron en el campo de las Ciencias Naturales, según Okabe et al. [8], la primera aplicación conocida del concepto de Diagrama de Voronoi aparece en un mapa incluido en el *Reporte sobre el brote de Cólera de St. James, Westminster, durante el otoño de 1854*. Este mapa muestra una línea discontinua, descrita como *Frontera con igual distancia entre la Bomba de la Calle Broad y otras Bombas*, la cual encierra un área alrededor de la Bomba de la Calle Broad (ver Figura 1.3). Aunque no hay alguna atribución asociada a este mapa, es más probable que esta obra de John Snow, donde se muestra la distribución de muertes alrededor de la Bomba de la Calle Broad en 1854, se haya convertido en el *mapa más famoso de alguna enfermedad del siglo XIX* que se reproduce en gran variedad de textos, tanto en epidemiología como en cartografía. Hay que notar que la distancia en el mapa no es medida en términos de Distancia Euclidiana, sino en términos de distancia

a lo largo de la red de calles de Westminster, logrando una red de Diagrama de *área-Voronoi*, este concepto aparecería ciento cincuenta años después [13].

Otra aplicación aparece en el área de Ciencias Sociales, en el trabajo del germanista Carl Hagg, que utilizó los Diagramas de Voronoi como un medio para visualizar la variación dialectal y así mismo la identificación de *Isoglosas* (barreras lingüísticas). En su estudio de dialectos al sureste de Alemania (1898) él definió el equivalente a un Diagrama de Voronoi de un conjunto de localidades en las que se habían recogido datos de su dialecto. Posteriormente pintó los bordes de Voronoi compartidos por dos localidades si diferían en términos de características de dialecto, así, bordes muy remarcados indicaban la presencia de Isoglosas (ver Figura 1.4) [14].

Además de estos trabajos, sólo se han venido desarrollando otras aplicaciones en las Ciencias Sociales durante los últimos cincuenta años, uno de los primero fue el desarrollado por Donald Bogue en 1949, quien usó los polígonos de Voronoi definidos alrededor de centros metropolitanos en Estados Unidos (representados como puntos en el mapa) como sustitutos de sus áreas de mercado (ver Figura 1.5) [15]. Apoyado por otros estudios, como los de Snyder (1962) y Dacey (1965) ésta ha seguido siendo la principal área de aplicación con el trabajo que se extiende hasta niveles de tiendas minoristas. Aunque también geógrafos, antropólogos y arqueólogos han usado el concepto para modelar otros tipos de sistemas territoriales humanos. Otros antecedentes relacionados se encuentran en la economía espacial, con la *Ley de las Áreas de Mercado*, la cual considera la forma de la frontera entre las áreas de mercado de dos centros que compiten en diversas condiciones de precios de mercado y costos de transporte. Argumentos similares pueden hallarse definiendo varios tipos de Diagramas de Voronoi Ponderados (balanceados) [16].

Las aplicaciones empíricas permanecieron limitadas debido a la falta de un medio sencillo y eficaz de construcción de Diagramas de Voronoi, dependiendo de métodos que implicaban el uso de regla y compás, así como lleno de ambigüedades. Aunque se trataron de mostrar métodos alternativos para su creación, tal es el caso de Richard Kopec (1963), quien propuso un método alternativo para la construcción

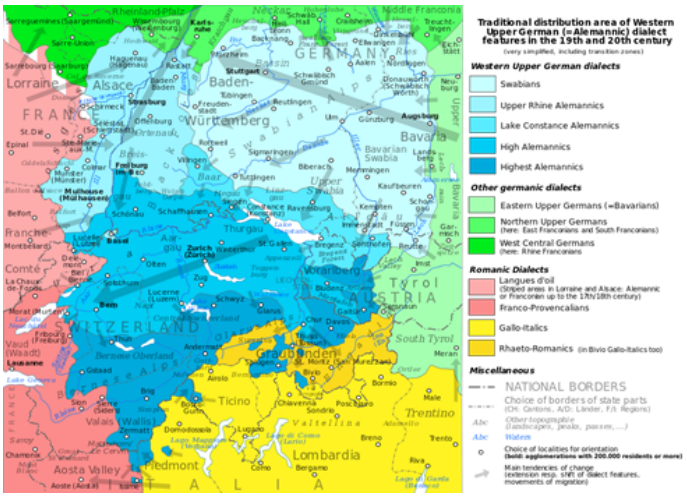
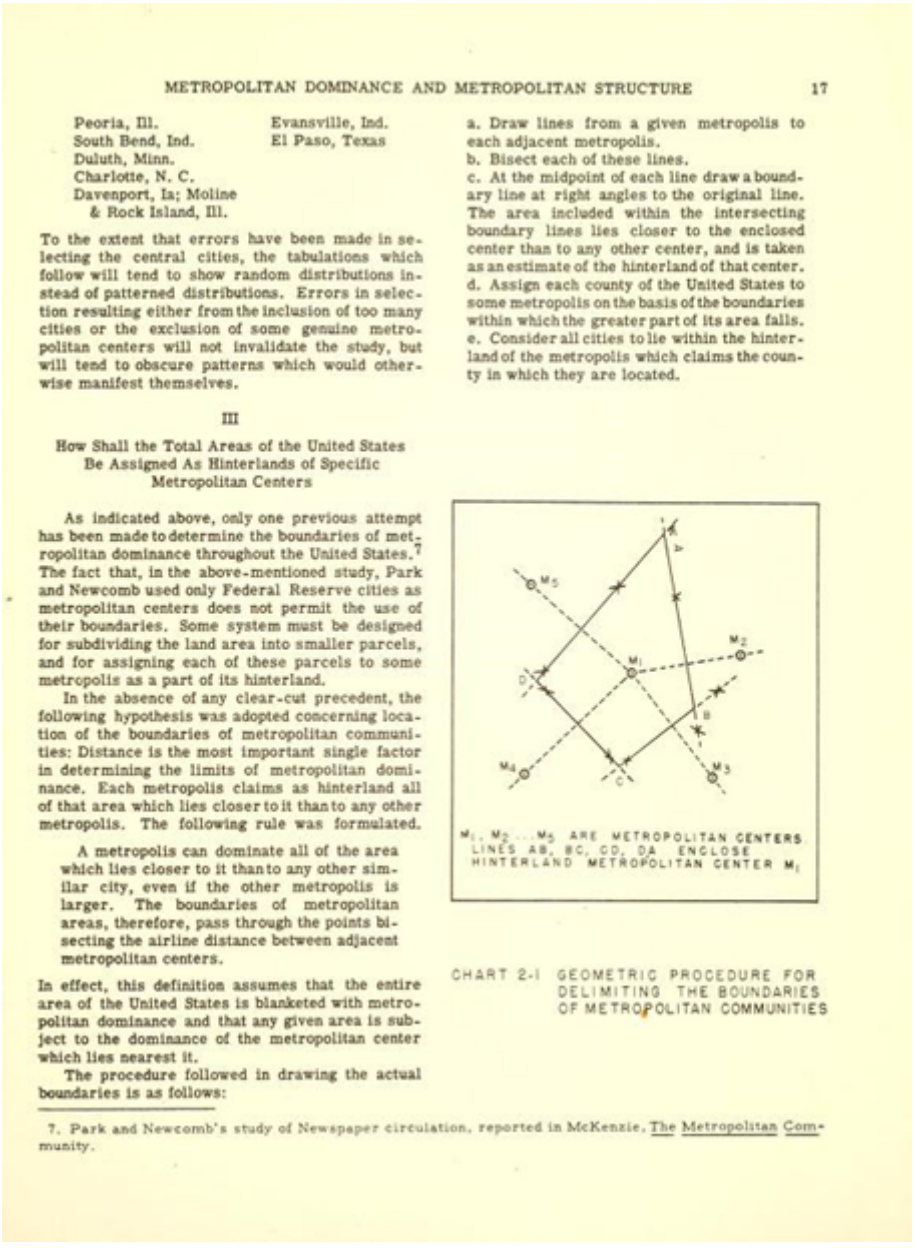


FIGURA 1.4: Geografía de los dialectos de Suabia.

de polígonos de Thiessen [17]. Esto provocó la búsqueda de soluciones desde el área en desarrollo de la informática y en los 70′s ya se habían desarrollado una serie de algoritmos para construir Diagramas de Voronoi en dos y tres dimensiones, que a su vez motivó otros desarrollos en Ciencias de la Computación que contribuyeron al campo de la naciente Geometría Computacional. Este esfuerzo se ve unificado por la obra pionera de Michael Shamos y Dan Hoey (1975) *Closest-point Problems* (Problemas de punto más cercano), donde no sólo presentan un algoritmo para la construcción del Diagrama de Voronoi, sino que también muestran la forma en la que éste se podría utilizar para resolver lo que entonces eran una serie de problemas relativos a un conjunto finito de puntos distintos en el espacio euclídeo, como hallar el árbol de expansión mínimo, identificar el vecino más cercano de cada punto y encontrar el círculo más grande que contiene a los puntos de un conjunto, cuyo centro está en el interior de la envolvente convexa de ese conjunto de puntos (ver Figura 1.6). También propusieron formas de generalizar los Diagramas de Voronoi, considerando las regiones de Voronoi asociadas con subconjuntos de k puntos de todo el conjunto de puntos en lugar de puntos individuales [18]. Como consecuencia de ésta y otras iniciativas de aquella época, el concepto básico se ha extendido en los últimos años en una gran variedad de formas.

Su concepto dual, la Teselación de Delaunay también tiene una historia marcada por redescubrimientos, aunque no tan frecuentes como el caso de los Diagramas de Voronoi. La idea se originó con Voronoi en 1908, que la define por medio de



METROPOLITAN DOMINANCE AND METROPOLITAN STRUCTURE 17

Peoria, Ill.
South Bend, Ind.
Duluth, Minn.
Charlotte, N. C.
Davenport, Ia; Moline & Rock Island, Ill.
Evansville, Ind.
El Paso, Texas

To the extent that errors have been made in selecting the central cities, the tabulations which follow will tend to show random distributions instead of patterned distributions. Errors in selection resulting either from the inclusion of too many cities or the exclusion of some genuine metropolitan centers will not invalidate the study, but will tend to obscure patterns which would otherwise manifest themselves.

III

How Shall the Total Areas of the United States Be Assigned As Hinterlands of Specific Metropolitan Centers

As indicated above, only one previous attempt has been made to determine the boundaries of metropolitan dominance throughout the United States.[7] The fact that, in the above-mentioned study, Park and Newcomb used only Federal Reserve cities as metropolitan centers does not permit the use of their boundaries. Some system must be designed for subdividing the land area into smaller parcels, and for assigning each of these parcels to some metropolis as a part of its hinterland.

In the absence of any clear-cut precedent, the following hypothesis was adopted concerning location of the boundaries of metropolitan communities: Distance is the most important single factor in determining the limits of metropolitan dominance. Each metropolis claims as hinterland all of that area which lies closer to it than to any other metropolis. The following rule was formulated.

> A metropolis can dominate all of the area which lies closer to it than to any other similar city, even if the other metropolis is larger. The boundaries of metropolitan areas, therefore, pass through the points bisecting the airline distance between adjacent metropolitan centers.

In effect, this definition assumes that the entire area of the United States is blanketed with metropolitan dominance and that any given area is subject to the dominance of the metropolitan center which lies nearest it.

The procedure followed in drawing the actual boundaries is as follows:

a. Draw lines from a given metropolis to each adjacent metropolis.
b. Bisect each of these lines.
c. At the midpoint of each line draw a boundary line at right angles to the original line. The area included within the intersecting boundary lines lies closer to the enclosed center than to any other center, and is taken as an estimate of the hinterland of that center.
d. Assign each county of the United States to some metropolis on the basis of the boundaries within which the greater part of its area falls.
e. Consider all cities to lie within the hinterland of the metropolis which claims the county in which they are located.

CHART 2-I GEOMETRIC PROCEDURE FOR DELIMITING THE BOUNDARIES OF METROPOLITAN COMMUNITIES

7. Park and Newcomb's study of Newspaper circulation, reported in McKenzie, The Metropolitan Community.

FIGURA 1.5: Estudio de áreas de mercado por Bogue. Copyright: Universidad de Florida.

relaciones entre vecinos, refiriéndose a la estructura resultante como *lensamble de simplexes*. Sin embargo, fue Deloné quien definió la Teselación (mosaico) usando el método de la esfera vacía [19]. Una de las propiedades de la Teselación de Delaunay en dos dimensiones es que los triángulos individuales son tan equiláteros como sea posible.

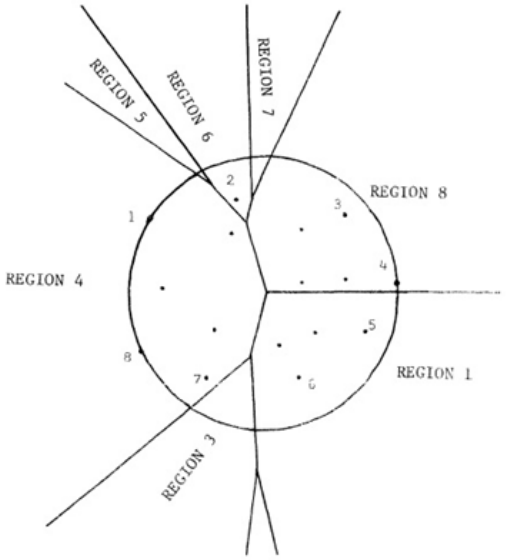


Figure 18. A farthest-point Voronoi diagram.

FIGURA 1.6: Círculo más grande que contiene a los puntos de un conjunto, Shamos y Hoey.

Capítulo 2

Preliminares

2.1 Conceptos básicos de la teoría de grafos

La teoría de grafos es una disciplina antigua con muchas aplicaciones modernas. Las ideas básicas fueron introducidas por el matemático suizo Leonhard Euler en el siglo XVIII. Euler utilizó los grafos para resolver el famoso problema de los puentes de *Königsberg* [20].

Los grafos son estructuras discretas que constan de vértices y de aristas que conectan entre sí a los vértices.

Definición 2.1 (**Grafo**). Un *grafo* G es un par $G = (V, E)$, donde V es un conjunto finito (vértices, nodos) y E es un multiconjunto de pares no ordenados de vértices, denotados por $\{x, y\}$, que se denominan *aristas* (lados).

Definición 2.2 (**Subgrafo**). Sea$G = (V, E)$ un grafo, si $H = (W, F)$ es un grafo tal que $W \subseteq V$ y $F \subseteq E$ decimos que H es un *subgrafo* de G.

En este caso decimos que x y y son extremos de $\{x, y\}$. Denotamos $V(G)$ por el conjunto de vértices del grafo G y por $E(G)$ el conjunto de lados del grafo G. Además $\upsilon(G)$ y $\epsilon(G)$ denotan el número de vértices y el número de aristas de G respectivamente. Puesto que E es un multiconjunto es posible que existan pares repetidos y se dice que G tiene lados múltiples. También es posible que algún par

no ordenado de E tenga el mismo vértice repetido, en este caso decimos que el lado es un *bucle* (*loop*). Cuando existen lados múltiples y/o lazos decimos que G es un *multigrafo*. Si no hay lados múltiples ni lazos decimos que es un grafo simple. Un digrafo G es un par $G = (V, E)$ donde V es un conjunto de vértices y E es un multiconjunto de pares ordenados. Los lados se denotan por pares ordenados, (u, v) denota el lado dirigido que tiene como vértice inicial a u y como vértice terminal a v.

A continuación se presentan unas definiciones, extraídas del libro *Matemáticas discretas y sus aplicaciones*, de Rosen [21].

Definición 2.3 (**Grafo Simple**). Un *grafo simple* $G = (V, E)$ consta de V, un conjunto no vacío de *vértices*, y de E, un conjunto de pares no ordenados de elementos distintos de V, a esos pares se les llama *aristas* o lados (Figura 2.1).

En algunos casos lo grafos simples no bastan para modelar ciertas situaciones en las cuales se requiere de la existencia de múltiples aristas entre par de vértices. En este caso no es suficiente definir las aristas como par de vértices; en su lugar se utilizan los multigrafos, que constan de vértices y de aristas no dirigidas entre ellos, pero admiten la existencia de aristas múltiples entre pares de vértices.

Definición 2.4 (**Multigrafo**). Un *multigrafo* $G(V, E)$ consta de un conjunto V de vértices, un conjunto E de aristas y una función f de E en $\{\{u, v\}|u, v \in V, u \neq v\}$. Se dice que las aristas $e1$, $e2$ son aristas *múltiples* o *paralelas* si $f(e1) = f(e2)$ (Figura 2.2).

Los multigrafos definidos no admiten *bucles* o lazos (aristas que conectan un vértice consigo mismo). Usamos en este caso, pseudografos que son más generales que los multigrafos.

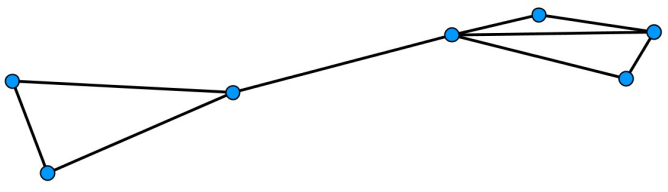


FIGURA 2.1: Grafo simple.

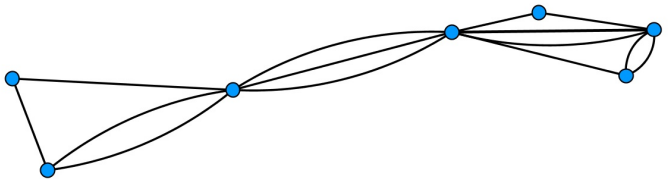


FIGURA 2.2: Multigrafo.

Definición 2.5 (**Pseudografo**). Un *pseudografo* $G = (V, E)$ consta de un conjunto V de vértices, un conjunto E de aristas y una función f de E en $\{\{u, v\}|u, v \in V, u \neq v\}$. Se dice que una arista e es un *bucle* o lazo si $f(e) = \{u, u\} = \{u\}$ para algún $u \in V$ (Figura 2.3).

La diferencia entre grafo y digrafo es que el último tiene los lados dirigidos y se entiende como un grafo dirigido.

Definición 2.6 (**Digrafo**). Un *grafo dirigido* o *digrafo* $G = (V, E)$ consta de un conjunto V de vértices y un conjunto E de aristas que son pares ordenados de elementos de V (Figura 2.4).

Cuando no se consideran las direcciones de las aristas en G, el grafo que se obtiene se llama *grafo subyacente* de G.

Finalmente, pueden existir multigrafos que pueden tener *aristas dirigidas* múltiples desde un vértice a un segundo vértice (que, ocasionalmente, puede coincidir con el primero).

Definición 2.7 (**Multigrafo dirigido**). Un *multigrafo dirigido* $G(V, E)$ consta de un conjunto V de vértices, un conjunto E de aristas y una función f de E en $\{\{u, v\}|u, v \in V, u \neq v\}$. Se dice que las aristas $e1$, $e2$ son aristas múltiples o paralelas si $f(e1) = f(e2)$ (ver Figura 2.5).

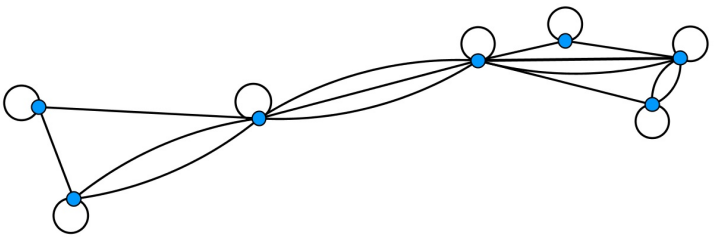


FIGURA 2.3: Pseudografo.

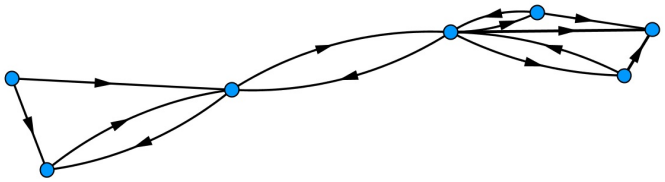


FIGURA 2.4: Grafo dirigido (digrafo).

Se resume la terminología que se emplea para los diferentes tipos de grafos en la Tabla 2.1.

2.2 Adyacencia de vértices, incidencia de aristas y grado de vértices

Veamos en primer lugar la terminología que describe los vértices y las aristas de los grafos no dirigidos.

Definición 2.8. Dos vértices u, v de un grafo *no dirigido* $G = (V, E)$ son *adyacentes* (vecinos) en G si $\{u, v\}$ es una arista de G. Si $e = \{u, v\}$, se dice que la arista e es *incidente* con los vértices u y v (la arista e conecta u y v). Los vértices u y v son *extremos* de la arista e.

Para saber cuántas aristas son incidentes con un vértice, se introduce la siguiente definición.

Definición 2.9. El *grado de un vértice* de un *grafo no dirigido* es el número de aristas incidentes con él (exceptuando los bucles). El grado de un vértice u se denota $\sigma(u)$.

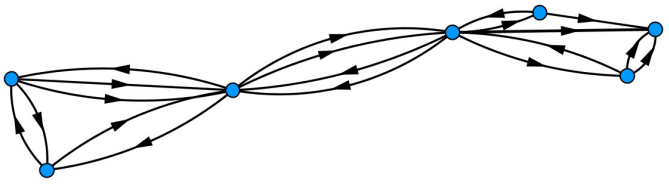


FIGURA 2.5: Multigrafo dirigido.

Tipos	*Aristas*	*¿Admite aristas múltiples?*	*¿Admiten bucles?*
Grafo simple	No dirigidas	No	No
Multigrafo	No dirigidas	Sí	No
Pseudografo	No dirigidas	Sí	Sí
Grafo dirigido	Dirigidas	No	Sí
Multigrafo	Dirigidas	Sí	Sí

TABLA 2.1: Terminología en teoría de grafos.

Los vértices de grado uno son vértices *aislados*, puede verse que un vértice aislado no es adyacente a ningún otro vértice. Se dice que un vértice es *colgante* (*hoja*) si y sólo si tiene grado uno, entonces una hoja es adyacente a un solo vértice distinto de ella.

Cuando se suman los grados de todos los vértices de un grafo G cada arista contribuye con dos unidades a la suma, ya que cada una de ellas es incidente con dos vértices.

Teorema 2.1 (**Teorema del apretón de manos**). Sea $G = (V, E)$ un grafo no dirigido con e aristas. Entonces,

$$2e = \sum_{v \in V} \sigma(V) \tag{2.1}$$

Es decir, la suma de los grados de los vértices es el doble que el número de aristas (notar que esto es cierto incluso cuando existen aristas múltiples y bucles en el grafo). También se sigue que la suma de los grados de los vértices de un grafo no dirigido es un número par.

Teorema 2.2. Todo grafo no dirigido tiene un número par de vértices de grado impar.

La terminología para grafos dirigidos evidencia que a las aristas de un grafo dirigido se les asigna una dirección o sentido.

Definición 2.10. Si $\{u, v\}$ es una arista del grafo dirigido $G = (V, E)$; se dice que u, conocido como *vértice inicial* es *adyacente* a v, *vértice final o terminal*, o

también que v es adyacente desde u. En un bucle el vértice inicial siempre coincide con el final.

Como las aristas de un grafo dirigido son pares ordenados, se puede mejorar la definición de grado de un vértice, con el fin de reflejar el número de aristas que tienen a ese vértice como inicial o final.

Definición 2.11. En un grafo dirigido, el *grado de entrada de un vértice* v, denotado por $\sigma^-(v)$, es el número de aristas que tienen a v como vértice final. El *grado de salida de un vértice* v, denotado por $\sigma^+(v)$, es el número de aristas que tienen a v como vértice inicial. Un bucle contribuye en una unidad tanto al grado de entrada como al grado de salida del vértice correspondiente.

Teorema 2.3. Sea $G = (V, E)$ un grafo dirigido. Entonces,

$$\sum_{v \in V} \sigma^-(v) + \sum_{v \in V} \sigma^+(v) = |E| \tag{2.2}$$

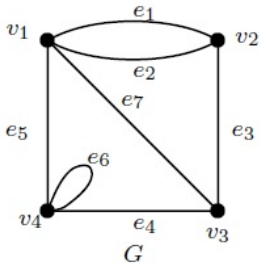
Los teoremas anteriores han sido demostrados por Rosen en [21].

2.3 Representaciones de los grafos

Sea $G = (V, E)$ un grafo con $\upsilon(G)$ vértices y $\epsilon(G)$ aristas, entonces, le corresponde una matriz $\upsilon \times \epsilon$ denominada *matriz de incidencia* de G. Denotamos los vértices de G por $v_1, \ldots, v_\upsilon$ y las aristas por $e_1, \ldots, e_\epsilon$. Entonces la matriz de incidencia de G es la matriz $M(G) = [m_{ij}]$ (ver Figura 2.6), donde m_{ij} es el número de veces que la arista e_j incide en el vértice v_i, dada por

$$m_{ij} = \begin{cases} 1 & \text{si } e_j \text{ es incidente con } v_i \\ 0 & \text{en caso contrario.} \end{cases} \tag{2.3}$$

Otra matriz asociada a G es la *matriz de adyacencia*, la cual es una matriz $\upsilon \times \upsilon$ $A(G) = [a_{ij}]$ (ver Figura 2.6), donde a_{ij} es el número de aristas que van de v_i hasta v_j, dada por



G

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
v_1	1	1	0	0	1	0	1
v_2	1	1	1	0	0	0	0
v_3	0	0	1	1	0	0	1
v_4	0	0	0	1	1	2	0

$\mathbf{M}(G)$

	v_1	v_2	v_3	v_4
v_1	0	2	1	1
v_2	2	0	1	0
v_3	1	1	0	1
v_4	1	0	1	1

$\mathbf{A}(G)$

FIGURA 2.6: Matriz de incidencia y de adyacencia de G.

$$m_{ij} = \begin{cases} 1 & \text{si } \{v_i, v_j\} \text{ es arista de } G \\ 0 & \text{en caso contrario.} \end{cases} \tag{2.4}$$

La matriz de adyacencia de un grafo depende del orden elegido para los vértices. Por lo tanto hay $v!$ para un grafo de v vértices, puesto que hay $v!$ ordenaciones distintas de los v vértices.

2.4 Isomorfismo de grafos

A menudo se requiere conocer si es posible representar dos grafos de la misma forma. Por ejemplo, en Química, se utilizan los grafos para representar compuestos. Diferentes compuestos pueden tener la misma fórmula molecular, pero distinta estructura.

Definición 2.12 (**Isomorfismo de grafos**). Los grafos simples $G_1 = (V_1, E)$ y $G_2 = (V_2, E)$ son *isomorfos* si existe una función biyectiva f de V_1 en V_2 con la propiedad de que, para cada par de vértices $u, v \in V_1$, u y v son adyacentes en G_1 si y sólo si $f(u)$ y $f(v)$ son adyacentes en G_2. Se dice que la función f es un *isomorfismo* (Figura 2.7).

Es decir, cuando dos grafos simples son isomorfos, hay una relación biyectiva entre los vértices de los grafos que preserva la relacin de adyacencia.

Con frecuencia es difícil determinar si dos grafos simples son isomorfos, puesto que existen $v!$ posibles biyecciones entre los conjuntos de vértices de dos grafos

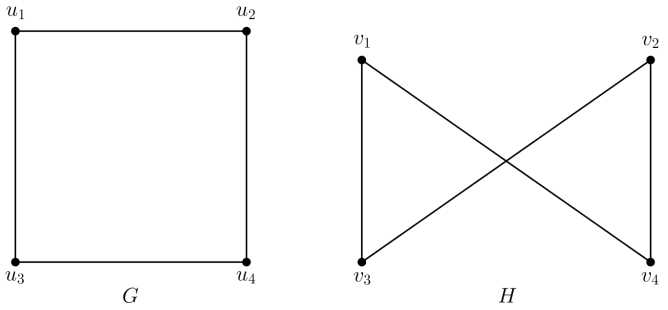


FIGURA 2.7: Grafos simples isomorfos.

simples de v vértices. No obstante, se puede demostrar que dos grafos no son isomorfos si no comparten alguna propiedad que dos grafos isomorfos deberían tener en común. A tales propiedades se les llama *invariantes* bajo isomorfismo de grafos simples, por ejemplo, dos grafos isomorfos deberían tener el mismo número de vértices (puesto que hay una biyección entre los conjuntos de vértices de los grafos), así mismo dos grafos isomorfos deberían tener el mismo número de aristas (ya que la biyección entre los vértices establece una biyección entre las aristas). Además, los grados de los vértices de dos grafos simples isomorfos deben coincidir.

Los mejores algoritmos conocidos para determinar si dos grafos son o no isomorfos tienen complejidad exponencial, aunque se conocen algoritmos de complejidad polinómica en el caso promedio que resuelven el problema. El mejor algoritmo en la práctica, conocido como *NAUTY*, puede usarse para determinar, en menos de un segundo, si dos grafos con menos de cien vértices son isomorfos [22].

2.5 Caminos y ciclos

Hay muchos problemas que se pueden representar por medio de caminos que se forman al ir recorriendo las aristas de un grafo. Por ejemplo, el problema de determinar si se puede enviar un mensaje entre dos ordenadores usando enlaces intermedios puede estudiarse utilizando un modelo de grafos. En algunos textos, por ejemplo Brualdi [23], se distingue entre *cadenas* (chains) y *caminos* (paths),

usando el primer término para grafos y el segundo para digrafos. Aquí usaremos el concepto de camino.

Definición 2.13. Sea n un entero no negativo y sea G un *grafo no dirigido*. Un *camino de longitud* n de u a v en G es una secuencia de n aristas $e_1, \ldots, e_n$ de G tal que $f(e_1) = \{v_0, v_1\}, \ldots, f(e_n) = \{v_{n-1}, v_n\}$, donde $v_0 = u$ y $v_n = v$. Si el grafo es simple, se denota este camino por su secuencia de vértices $v_0, \ldots, v_n$. El camino es un *circuito* si comienza y termina en el mismo vértice, es decir, si $u = v$ y tiene longitud mayor que cero. Se dice que el camino o circuito *pasa por* los vértices $v_1, \ldots, v_{n-1}$, o también, que *recorre* las aristas $e_1, \ldots, e_n$. Un camino o circuito es *simple* si no contiene la misma arista más de una vez.

Definición 2.14. Un *camino* de u a v en el *grafo dirigido* G es una sucesión de aristas $\{v_0, v_1\}, \ldots, \{v_{n-1}, v_n\}$ de G, con n entero no negativo, $v_0 = u$ y $v_n = v$. Este camino se denota como $v_0, \ldots, v_n$ y tiene longitud n. El camino vacío, sin alguna arista, se considera como un camino de u a u. Un camino de longitud $n \geq 1$, que comienza y termina en el mismo vértice, se llama *circuito*.

Definición 2.15. Sea n un entero no negativo y sea G un *multigrafo dirigido*. Un *camino de longitud* n de u a v en G es una secuencia de n aristas $e_1, \ldots, e_n$ de G tal que $f(e_1) = \{v_0, v_1\}, \ldots, f(e_n) = \{v_{n-1}, v_n\}$, donde $v_0 = u$ y $v_n = v$. Si no hay aristas múltiples en el grafo dirigido, denotamos este camino por su secuencia de vértices $v_0, \ldots, v_n$. Un camino de longitud mayor que cero que comienza y termina en el mismo vértice es un *circuito*. Se dice que un camino o circuito es *simple* si no contiene la misma arista más de una vez.

En las tres definiciones notamos que el vértice final de una arista de un camino es el vértice inicial de la siguiente arista del camino.

Definición 2.16 (**Longitud**). La *longitud* de un camino es el número de lados que hay en él.

Definición 2.17 (**Distancia**). La distancia, conocida como *geodésica*, entre dos vértices distintos, es igual a la longitud de la cadena más corta entre ellos; si no hay camino entre ellos la distancia no está definida y la distancia es cero si los vértices son iguales.

Definición 2.18 (**Diámetro**). El *diámetro* de un grafo es el máximo de las distancias entre cualquier par de vértices. Un camino $\alpha = (v_0, \ldots, v_n)$ es cerrado si $v_0 = v_n$.

En el grafo simple que se muestra en la Figura 2.8, a, d, c, f, e es un camino simple de longitud 4, ya que $\{a, d\}, \{d, c\}, \{c, f\}$ y $\{f, e\}$ son aristas. Sin embargo, d, e, c, a no lo es, ya que $\{e, c\}$ no es una arista. Notar que b, c, f, e, b es un circuito de longitud 4, ya que $\{b, c\}, \{c, f\}, \{f, e\}$ y $\{e, b\}$ son aristas, además este camino comienza y termina en b. El camino a, b, e, d, a, b, que tiene longitud 5, no es simple, ya que contiene dos veces la arista $\{a, b\}$.

2.6 Conexidad

Un grafo G es *conexo* si existe un camino entre cualquier par de vértices. Veremos primero la *conexidad* en grafos no dirigidos.

Definición 2.19. Un *grafo no dirigido* G es *conexo* si hay un camino entre cada par de vértices distintos de G.

El grafo G_1 de la Figura 2.9 es conexo, ya que para cada para de vértices distintos hay un camino entre ellos. Sin embargo, el grafo G_2 no lo es, por ejemplo, no hay algún camino entre a y d.

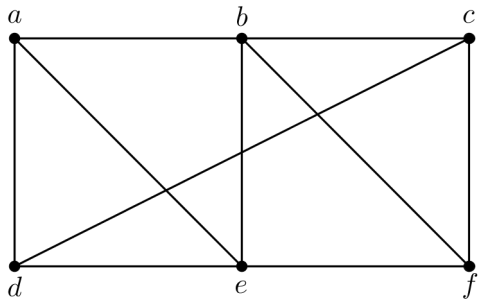


FIGURA 2.8: Caminos en un grafo simple.

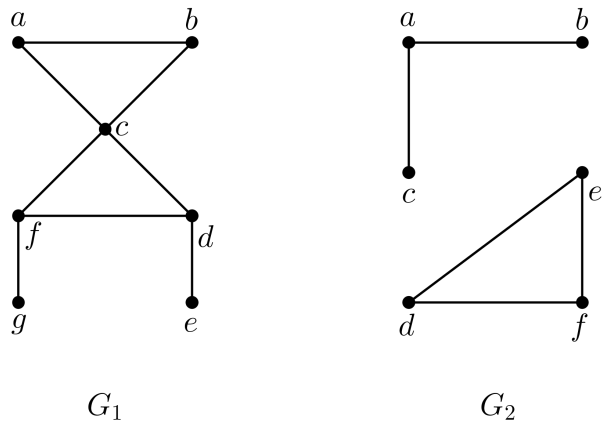


FIGURA 2.9: El grafo G_1 es conexo y el grafo G_2 no lo es.

Un grafo que no es conexo es la unión de dos o más subgrafos conexos que dos a dos no tienen ningún vértice común. A estos subgrafos conexos disjuntos se les llama *componentes conexas* del grafo (ver Figura 2.10)

Hay dos ideas de conexión en grafos dirigidos, depende si se considera o no la dirección de las aristas.

Definición 2.20. Se dice que un *grafo dirigido* G es *fuertemente conexo* si hay un camino de u a v y un camino de v a u para cualesquiera dos vértices u, v de G.

Para que un grafo dirigido sea fuertemente conexo tiene que haber una secuencia de aristas dirigidas desde cualquier vértice del grafo a cualquier otro vértice. Un grafo dirigido puede no ser fuertemente conexo, pero estar formado de una sola pieza.

Definición 2.21. Se dice que un *grafo dirigido* G es *débilmente conexo* si hay un camino entre cada dos vértices del grafo dirigido *subyacente* a G.

Es decir, un grafo no dirigido es débilmente conexo si y sólo si hay siempre un camino entre dos vértices cuando se ignoran las direcciones de las aristas. Notar que cualquier grafo dirigido fuertemente conexo también es débilmente conexo.

El grafo dirigido G de la Figura 2.11 es fuertemente conexo porque hay un camino entre cualesquiera dos vértices (por lo que es débilmente conexo). Mientras, el grafo dirigido H no es fuertemente conexo, ya que no hay algún camino de a a

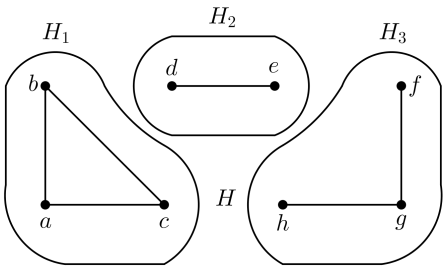


FIGURA 2.10: El grafo H y sus componentes conexas H_1, H_2 y H_3.

b; no obstante es débilmente conexo, puesto que hay un camino entre cada dos vértices en el grafo dirigido subyacente a H.

A aquellos subgrafos de un grafo dirigido G que son fuertemente conexos, pero que no están contenidos en algún subgrafo fuertemente conexo mayor, se les llama *componentes fuertemente conexas* o *componentes fuertes* de G.

2.7 Grafos planos

Decimos que un grafo G es plano si se puede dibujar en el plano sin que los lados se crucen fuera de sus extremos.

Definición 2.22 (**Grafo plano**). Se dice que un grafo G es *plano* (o *planar*) si puede dibujarse en el plano de manera que ningún par de sus aristas se corte (por corte de aristas entendemos la intersección de las líneas que representan a las aristas en un punto distinto de sus extremos). Al dibujo final se le conoce como *representación plana* del grafo.

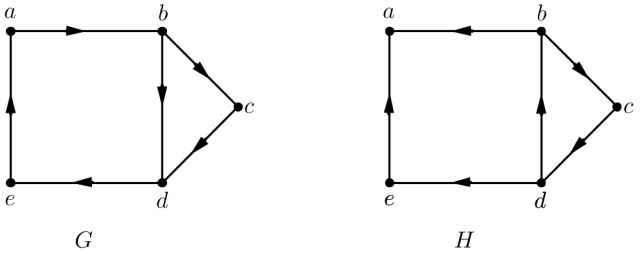


FIGURA 2.11: El grafo G es fuertemente conexo, el grafo H es débilmente conexo.

En la Figura 2.12, el grafo H es plano, dado que se puede hallar una representación sin cortes en sus aristas.

Un grafo puede ser plano aunque regularmente se dibuje con cortes de aristas, ya que se puede dibujar de manera diferente sin algún corte.

2.8 Fórmula de Euler

Una representación plana de un grafo divide el plano en regiones, incluyendo una región no acotada. Por ejemplo, la representación plana del grafo G que se muestra en la Figura 2.13 divide el plano en seis regiones, que están etiquetadas. Euler demostró [24] que todas las representaciones planas de un mismo grafo dividen al plano en igual número de regiones. Para lo cual halló una relación entre el número de regiones, el número de vértices y el número de aristas de un grafo plano. A continuación se presenta este resultado.

Teorema 2.4 (**Fórmula de Euler**). Sea G un grafo simple conexo con e aristas y v vértices. Sea r el número de regiones de una representación plana de G. Entonces,

$$r = e - v + 2 \tag{2.5}$$

Por ejemplo, supongamos que un grafo simple conexo tiene 20 vértices, cada uno de los cuales tiene grado 3. Veamos en cuántas regiones divide al plano una representación plana de ese grafo.

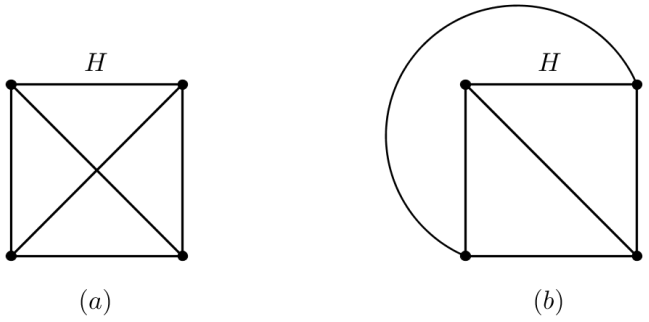


FIGURA 2.12: (a) El Grafo H, con dos cortes de aristas. (b) El grafo H, sin cortes de aristas.

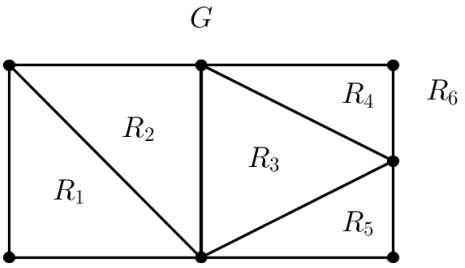


FIGURA 2.13: Regiones de la representación plana del grafo G.

El grafo tiene 20 vértices, cada uno de grado 3, es decir, $v = 20$. Como la suma de los grados de los vértices, $3v = 3 \cdot 20 = 60$, es igual al doble $2e$ del número de aristas, se tiene que $2e = 60$, o que $e = 30$. Por tanto, el número de regiones, según la fórmula de Euler, es $r = e - v + 2 = 30 - 20 + 2 = 12$.

Colorario 2.1. Sea G un grafo simple y conexo con e aristas y $v \geq 3$ vértices. Entonces,

$$e \leq 3v - 6 \tag{2.6}$$

Colorario 2.2. Sea G un grafo simple y conexo. Entonces G tiene un vértice de grado menor o igual que cinco.

Colorario 2.3. Sea G un grafo simple y conexo con e aristas y $v \geq 3$ vértices y que no tiene circuitos de longitud 3. Entonces,

$$e \leq 2v - 4 \tag{2.7}$$

Los corolarios han sido demostrados en [25].

A continuación demostraremos, usando el Corolario 2.1 y el Corolario 2.3, dos resultados que serán de utilidad más adelante. El grafo K_5 mostrado en la Figura 2.14(a) no es plano, así como el grafo $K_{3,3}$ presentado en la Figura 2.14(b).

Primero, para comprobar que K_5 no es plano, notemos que tiene 5 vértices y 10 aristas. La desigualdad 2.6 no se cumple para K_5 ya que $e = 10$ y $3v - 6 = 15 - 6 = 9$. Por lo tanto, K_5 no es plano.

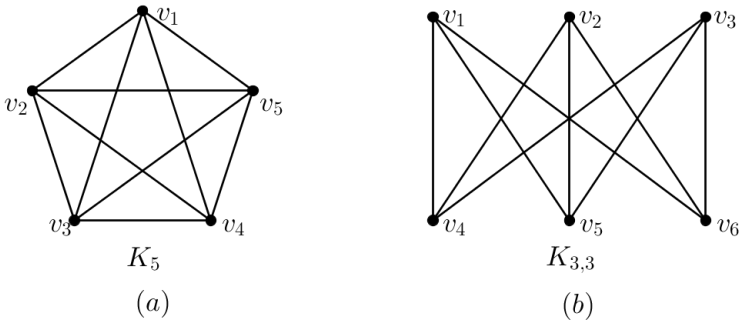


FIGURA 2.14: (a) Grafo K_5. (b) Grafo $K_{3,3}$.

Segundo, para comprobar que $K_{3,3}$ no es plano, notar que es conexo y no contiene circuitos de longitud 3 (puesto que es bipartito), entonces se puede utilizar el Corolario 2.3. El grafo $K_{3,3}$ tiene 6 vértices y 9 aristas. Como $e = 9$ y $2v - 4 = 12 - 4 = 8$, no cumple la desigualdad 2.7, por lo que $K_{3,3}$ no es plano.

2.9 Teorema de Kuratowski

Determinar cuándo un grafo es plano no siempre es tan inmediato porque, por ejemplo, puede contener muchos vértices y aristas. Lo que asegura el teorema de *Kuratowski*, establecido por el matemático polaco Kazimierz Kuratowski en 1930 [26], es que los grafos planos son aquellos que no contienen ni al grafo $K_{3,3}$ ni al grafo K_5 (ver Figura 2.14), los cuales no son planos. Es decir, *caracteriza* los grafos utilizando el concepto de *homeomorfismo de grafos*.

Si un grafo es plano, también lo será cualquier grafo que se obtenga de él eliminando una arista $\{u, v\}$ y añadiendo un vértice w junto con las aristas $\{u, w\}$ y $\{w, v\}$. Se dice que esta operación es una *subdivisión elemental*. También que los grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son *homeomorfos* si se pueden obtener a partir de un mismo grafo por medio de una secuencia de subdivisiones elementales.

Teorema 2.5 (**Teorema de Kuratowski**). Un grafo es plano si y sólo si no contiene ningún subgrafo homeomorfo a K_5 o a $K_{3,3}$.

West realiza una prueba a este teorema en [26].

2.10 Grafos duales

Hasta ahora se sabe que todo grafo plano G divide al plano en regiones acotadas y una que no lo es. Las clausuras de las regiones acotadas se llaman *caras de* G. La Figura 2.15 muestra un grafo plano G con 6 caras $f_1, \ldots, f_6$.

Definición 2.23 (**Grado de una cara**). El *grado de una cara* f es el número de aristas que hay en el borde (en las que la cara incide) de f, denotada como $\sigma(f)$.

Si una arista aparece dos veces en el borde (se pasa dos veces por encima de la arista cuando se dibuja el borde), contribuye al grado con dos unidades. Por ejemplo, en la Figura 2.15 $\sigma(f_5) = 6$.

Dado un grafo plano G, se puede construir otro grafo llamado *dual* de G de la siguiente manera.

Definición 2.24. Dado un grafo plano G, se puede generar otro grafo G^* como sigue: sea un vértice f^* de G^* correspondiendo a cada cara f de G y sea e^* una arista de G^* correspondiendo a cada arista e de G; dos vértices f^* y g^* están unidos por la arista e^* en G^* si y sólo si sus correspondientes caras f y g están separadas por la arista e en G. Se dice que G^* es el *grafo dual* de G (Figura 2.16).

Se debe observar que dos grafos planos isomorfos pueden tener duales no isomorfos, la noción de dualidad sólo tiene significado para grafos planos. A partir de la Definición 2.24 se siguen, de manera trivial, las siguientes propiedades:

Propiedad 2.1. El conjunto de vértices de G^* es el conjunto de caras de G.

Propiedad 2.2. El conjunto de aristas de G^* coincide con el conjunto de aristas de G.

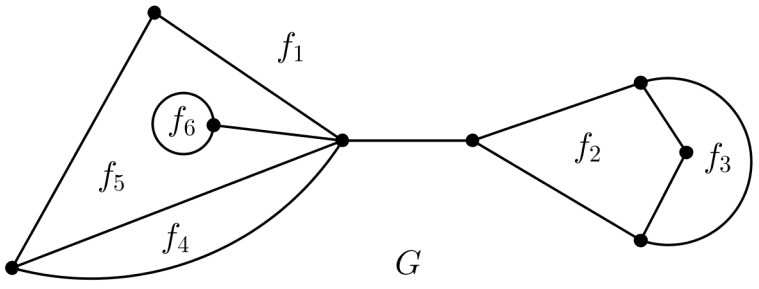


FIGURA 2.15: Ejemplo de caras de un grafo G.

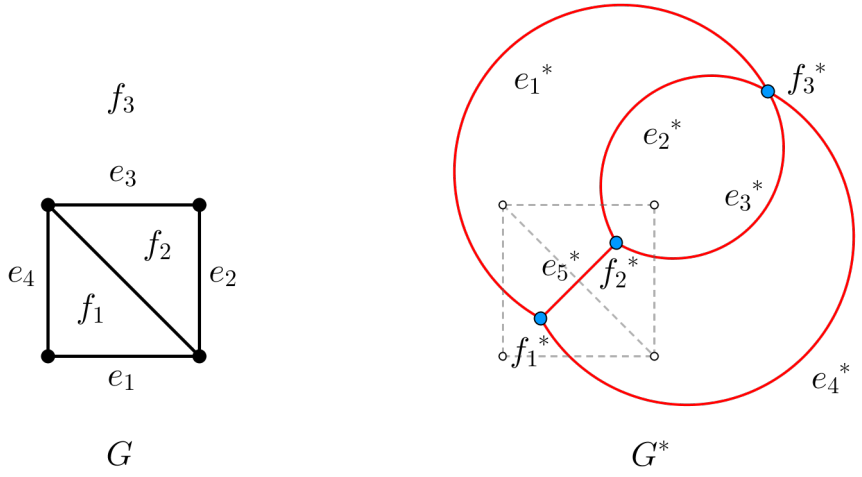


FIGURA 2.16: Grafo G y su dual G^*.

Capítulo 3

Diagramas de Voronoi

3.1 Conceptos Básicos

En esta sección denotaremos como P al conjunto de $n \geq 2$ puntos $p_1, \ldots, p_n$ en el plano Euclidiano, etiquetados mediante las coordenadas $(x_{11}, x_{12}), \ldots, (x_{n1}, x_{n2})$ o vectores de posición $x_1, \ldots, x_n$. Los n puntos son distintos en el sentido de que $x_i \neq x_j$ para $i \neq j, i, j \in I_n = 1, \ldots, n$.

Definición 3.1 (**Distancia euclidiana**). Sea p un punto arbitrario en el plano Euclidiano con coordenadas (x_1, x_2) o un vector de posición x. Entonces, la distancia Euclidiana entre p y p_i está dada por

$$d(p, p_i) = \|x - x_i\| = \sqrt{(x_1 - x_{i1})^2 + (x_2 - x_{i2})^2} \tag{3.1}$$

Si p_i es el punto más cercano a p o uno de los más cercanos, se tiene la relación $\|x - x_i\| \leq \|x - x_j\|$ para $i \neq j, i, j \in I_n$. En este caso, se dice que p está asignado a p_i.

Nota: Se considerará a partir de ahora que se trabaja en el espacio Euclidiano con la *métrica Euclidiana*.

Definición 3.2 (**Diagrama de Voronoi ordinario planar**). Sea el conjunto $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, con $2 < n < \infty$ y $x_i \neq x_j$ para $i \neq j, i, j \in I_n$. Llamamos

la región dada por

$$V(p_i) = \{x | \|x - x_i\| \leq \|x - x_j\|, i \neq j, i, j \in I_n\} \tag{3.2}$$

como el *Polígono de Voronoi Ordinario Planar* asociado a p_i (o el *Polígono de Voronoi* de p_i) y al conjunto dado por

$$\mathcal{V} = \{V(p_1), \ldots, V(p_n)\} \tag{3.3}$$

el *Diagrama de Voronoi Ordinario Planar* generado por P (o el *Diagrama de Voronoi* de P). Llamamos a p_i de $V(p_i)$ como el *Punto Generador* o *Generador* del $i-$ésimo Polígono de Voronoi y al conjunto $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ como el *Conjunto Generador* del Diagrama de Voronoi $\mathcal{V}$.

En la Definición 3.2 notamos que la relación en la ecuación 3.2 está definida en términos de $\leq$ pero no de $<$. Un polígono de Voronoi por lo tanto es un *conjunto cerrado*. Alternativamente, podemos definir un polígono de Voronoi como

$$V^o(p_i) = \{x | \|x - x_i\| < \|x - x_j\|, i \neq j, i, j \in I_n\} \tag{3.4}$$

el cual es un conjunto abierto. Ambas definiciones se aceptan, pero en este estudio definiremos un polígono de Voronoi como un conjunto cerrado.

Definición 3.3 (**Arista**). Puesto que un polígono de Voronoi es un conjunto cerrado, contiene a su frontera, que será denotada como $\partial V(p_i)$. Ésta puede consistir en segmentos de recta, semirrectas o rectas infinitas, las cuales llamamos *Aristas de Voronoi*. Denotamos una arista de Voronoi como e_i.

Notar que un diagrama de Voronoi a veces es definido por la unión de aristas de Voronoi, es decir, $\bigcup_{i=1}^{n} \partial V(p_i)$ en lugar del conjunto $\{V(p_1), \ldots, V(p_n)\}$. Puesto que la unión de aristas de Voronoi puede ser considerada como una red (*malla*), a veces es llamada *Red de Voronoi*.

Al percatarnos que = es incluido en la relación de la ecuación 3.2, podemos definir, alternativamente, una arista de Voronoi como un segmento de recta, semirrecta, o recta infinita compartida por dos polígonos de Voronoi con sus puntos extremos. Matemáticamente, si $V(p_i) \bigcap V(p_j) \neq \emptyset$, el conjunto $V(p_i) \bigcap V(p_j)$ genera una arista de Voronoi (que puede degenerar en un punto). Usamos $e(p_i, p_j)$ para denotar la intersección $V(p_i) \bigcap V(p_j)$, la arista de Voronoi generado por p_i y p_j. Notar que $e(p_i, p_j)$ puede ser vacío. Si $e(p_i, p_j)$ no es vacío ni un punto, decimos que los polígonos de Voronoi $V(p_i)$ y $V(p_j)$ son contiguos o *adyacentes*.

Definición 3.4 (**Vértice de Voronoi**). Un punto extremo de una arista de Voronoi se conoce como *Vértice de Voronoi*. También un vértice de Voronoi puede definirse como un punto compartido por tres o más polígonos de Voronoi. Denotamos un vértice de Voronoi como q_i.

Cuando existe al menos un vértice de Voronoi en el que cuatro o más aristas de Voronoi se encuentran en el diagrama de Voronoi $\mathcal{V}$ (los círculos sin rellenar en la Figura 3.1(a)), decimos que $\mathcal{V}$ es degenerado; de otra manera decimos que $\mathcal{V}$ es no degenerado.

En algunos resultados, un diagrama de Voronoi degenerado necesita tratamientos especiales prolongados que no siempre son indispensables. Para evitar esta dificultad, a menudo se hace la siguiente hipótesis.

Hipótesis 3.1 (**No degeneración**). Cada vértice de Voronoi en un diagrama de Voronoi tiene exactamente tres aristas de Voronoi.

En la Definición 3.2 definimos un diagrama de Voronoi en un plano abierto. En aplicaciones prácticas, frecuentemente se trata con una región acotada S donde se encuentran los puntos generadores (las líneas más gruesas en la Figura 3.2). En este caso, consideramos el conjunto $V_{\cap S}$, dado por

$$V_{\cap S} = \{V(p_1) \cap S, \ldots, V(p_n) \cap S\} \tag{3.5}$$

Lo llamamos *Diagrama de Voronoi Acotado* o el Diagrama de Voronoi Acotado por S. Si un polígono de Voronoi $V(p_i)$ comparte la frontera de S, llamamos a la

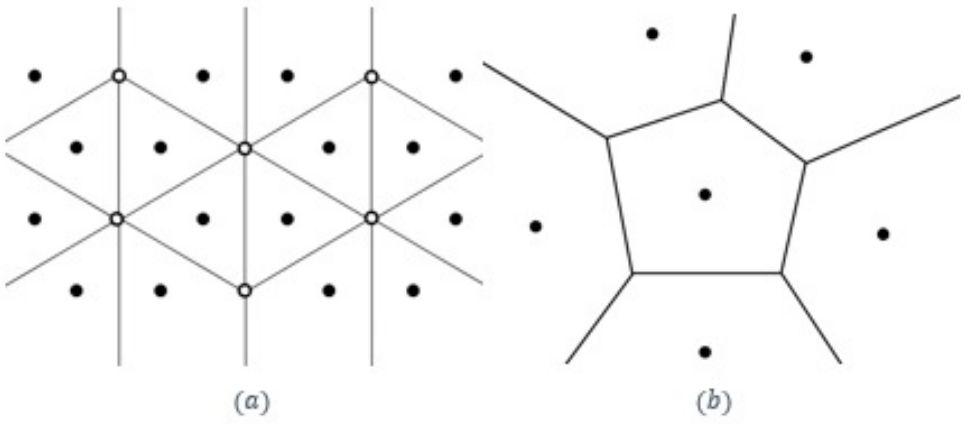


FIGURA 3.1: (a) Diagrama de Voronoi *degenerado*. (b) Diagrama de Voronoi *no degenerado*.

región $V(p_i) \cap S$ *frontera* del polígono de Voronoi o *región* (el término *región* se usa cuando ∂S es *curva* o cuando $V(p_i) \cap S$ es no conexa [27]). En la práctica, generalmente se tratan diagramas de Voronoi acotados bien definidos, como en la Figura 3.2(b), cuya frontera del polígono de Voronoi tiene forma de estrella (*star-shaped polygon*[1]). Una región de frontera de Voronoi inconexa se muestra en la Figura 3.2(a) (zona sombreada) y la frontera del polígono de Voronoi sin forma de estrella (*non-star-shaped polygon*) con respecto a su generador (línea punteada).

Como puede observarse en la Figura 3.2, el diagrama de Voronoi ordinario se compone de polígonos. Recordando que un polígono está definido en términos de semiplanos, se puede presentar una definición alternativa de un diagrama de Voronoi, en términos de semiplanos. Para ello primero debemos considerar la recta perpendicular que biseca al segmento $\overline{p_i p_j}$ que une dos puntos generadores p_i y p_j (Figura 3.3). Llamamos a esta línea la *bisectriz* entre p_i y p_j y se denota como $b(p_i, p_j)$. Como un punto en la bisectriz $b(p_i, p_j)$ es equidistante de los puntos generadores p_i y p_j, entonces podemos escribirla como

$$b(p_i, p_j) = \{x | \|x - x_i\| = \|x - x_j\|, i \neq j\} \tag{3.6}$$

La bisectriz divide el plano en dos semiplanos y genera

[1]Star-shaped polygon: Un polígono que contiene un punto desde el cual toda la frontera del polígono es visible.

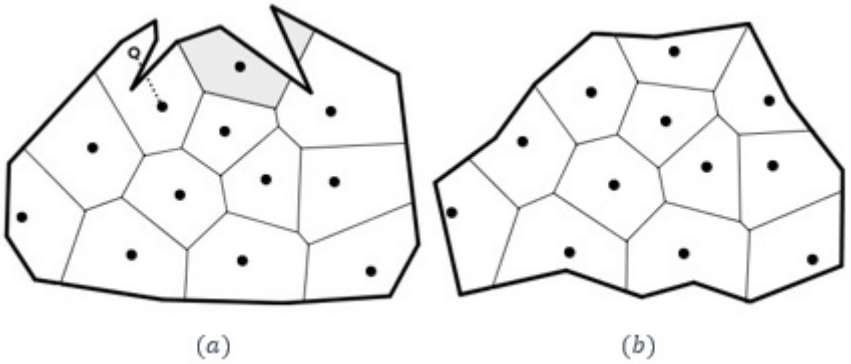


FIGURA 3.2: Diagramas de Voronoi Acotados.

$$H(p_i, p_j) = x \| \|x - x_i\| \leq \|x - x_j\|, i \neq j \tag{3.7}$$

Llamamos a $H(p_i, p_j)$ como *región dominio* de p_i sobre p_j. En la Figura 3.3 se indican las regiones dominio de p_1 sobre p_2, p_3 y p_4 por las regiones rayadas horizontal, diagonal y vertical, respectivamente. Notar que en la región de dominio $H(p_i, p_j)$ la distancia al punto generador p_i es menor o igual que la distancia al punto generador p_j. Por lo tanto, la distancia de cualquier punto p en la intersección de la Figura 3.3 al punto generador p_1 es menor o igual que la distancia de p al punto generador $p_i, i = 2, 3, 4$. Esta relación es equivalente a la ecuación 3.2 y así la intersección $H(p_1, p_2) \bigcap H(p_1, p_3) \bigcap H(p_1, p_4)$ genera el polígono de Voronoi asociado a p_1. De este ejemplo, se entiende que la siguiente definición es alternativa a la Definición 3.2.

Definición 3.5 (**Diagrama de Voronoi ordinario planar definido con semi-planos**). Sea $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, donde $2 \leq n < \infty$ y $x_i \neq x_j$ para $i \neq j, i, j \in I_n$. Llamamos a la región

$$V(p_i) = \bigcap_{j \in I_n \{i\}} H(p_i, p_j) \tag{3.8}$$

como el *Polígono de Voronoi Ordinario* asociado a p_i y al conjunto $\mathcal{V}(P) = \{V(p_1), \ldots, V(p_n)\}$ como el *Diagrama de Voronoi Ordinario Planar* generado por P.

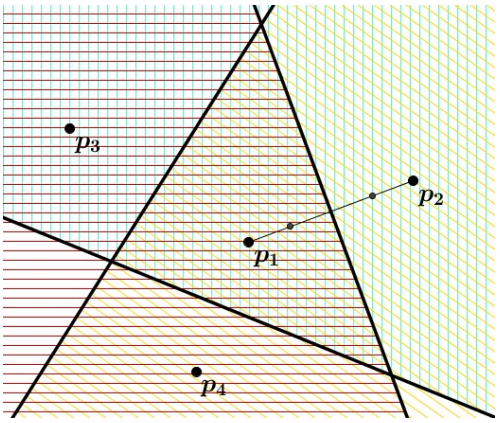


FIGURA 3.3: Un polígono de Voronoi obtenido a partir de semiplanos.

La equivalencia de la Definición 3.5 y la Definición 3.2 es evidente, debido a que $\|x - x_i\| \leq \|x - x_j\|$ si y sólo si $x \in H(p_i, p_j)$, para $i \neq j$. Se puede extender la definición anterior al espacio Euclidiano $n-$dimensional.

Definición 3.6 (**Diagrama de Voronoi ordinario en** $\mathbb{R}^n$). Sea un conjunto $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^n$, donde $2 \leq n < \infty$ y $x_i \neq x_j$ para $i \neq j, i, j \in I_n$. Llamamos a la región

$$V(p_i) = \{x | \|x - x_i\| \leq \|x - x_j\|, i \neq j, i, j \in I_n\} \tag{3.9}$$

$$= \bigcap_{j \in I_n \{i\}} H(p_i, p_j) \tag{3.10}$$

como el *Poliedro de Voronoi ordinario $m-$dimensional* asociado a p_i y al conjunto $\mathcal{V}(P) = \{V(p_1), \ldots, V(p_n)\}$ se le conoce como el *Diagrama de Voronoi Ordinario $m-$dimensional* generado por P, donde $H(p_i, p_j)$ es dado por la ecuación 3.7 para $p_i, p_j \in \mathbb{R}^n$.

Cuando se sobreentienda que se trabaja en $\mathbb{R}^n$, simplemente podemos llamar a $\mathcal{V}(P)$ como *Diagrama de Voronoi ordinario* o sólo *Diagrama de Voronoi* y a los poliedros (polígonos) de Voronoi serán denominados como *Celdas* de Voronoi.

Para el diagrama de Voronoi de tres dimensiones, las fronteras de un poliedro de Voronoi consisten en caras, por lo que son llamadas *caras de Voronoi*. Las fronteras de una cara de Voronoi consisten en segmentos de recta, semirrectas o rectas infinitas (*aristas* o bordes de Voronoi). Las fronteras de una arista de

Voronoi consisten en puntos, los cuales llamamos *vértices* de Voronoi. La Figura 3.4 muestra una vista estereográfica de un diagrama de Voronoi de tres dimensiones obtenido con el Software Libre `Voro++`.

En Astronomía, los poliedros de Voronoi son conocidos como *Espumas de Voronoi* (Icke y van de Weygaert [12]). En redes neurales tipo *el ganador se lleva todo* (winner-take-all), el poliedro de Voronoi $V(p_i)$ es llamado *campo receptivo* de una unidad neuronal i, siendo x_i el vector de fuerza (peso) sináptico de su unidad neuronal [28].

Formalmente, llamamos a las fronteras de un poliedro de Voronoi $n-$dimensional *caras de Voronoi* $(n-1)-$*dimensionales*, a las fronteras de un poliedro de Voronoi $(n-1)-$dimensional *caras de Voronoi* $(n-2)-$*dimensionales*, etc.; a las fronteras de una cara de Voronoi de $3-$dimensional *caras de Voronoi* $2-$*dimensionales* o simplemente *caras de Voronoi*, a las fronteras de una cara de Voronoi $2-$dimensional *caras de Voronoi* de $1-$*dimensionales* o *aristas de Voronoi*, a las fronteras de una cara de Voronoi $1-$dimensional *caras de Voronoi* $0-$*dimensionales* o *vértices de Voronoi*.

Cuando $n = 1$, el diagrama de Voronoi $1-$*dimensional*, o *diagrama de Voronoi en línea* (Figura 3.5), en este caso un *poliedro de Voronoi* $1-$*dimensional* es una semirrecta o segmento de recta llamada *recta de Voronoi* y los vértices de Voronoi son los puntos extremos de las rectas de Voronoi. Podemos notar que el punto frontera entre dos rectas de Voronoi adyacentes es el punto medio de los puntos generadores de aquellas rectas de Voronoi; el número de rectas de Voronoi no delimitadas siempre es dos; el número de rectas de Voronoi adyacentes a una recta de Voronoi es uno o dos.

En las definiciones anteriores, el espacio es representado como un plano continuo. Sin embargo, esta representación no puede ser aceptada en algunos contextos, por ejemplo en el análisis de imágenes de datos tipo *ráster*[2]. En un análisis de

[2]En su forma más simple, un ráster consta de una matriz de celdas (o píxeles) organizadas en filas y columnas (o una cuadrícula) en la que cada celda contiene un valor que representa información, como la temperatura. Los rásteres son fotografías aéreas digitales, imágenes de satélite, imágenes digitales o incluso mapas escaneados [30].

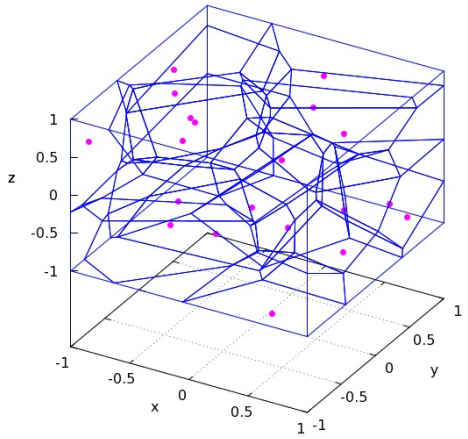


FIGURA 3.4: Vista estereográfica de un diagrama de Voronoi de puntos aleatorios en un cubo.

este tipo, el espacio está representado por un conjunto de celdas cuadradas que forman una cuadrícula con un patrón de *enrejado* y una figura geométrica se representa en función de estas celdas. Veamos, sea $c_{ij} = \{(x, y)|i \leq x \leq i+1, j \leq y \leq j+1\}, i, j = \ldots, -n, \ldots, -1, 0, 1, \ldots, n, \ldots$ Para un conjunto de puntos que representa una figura geométrica, G, definimos $Im(G) = \{c_{ij}|c_{ij} \cap G \neq \emptyset\}$, la *imagen* de G. En términos de $Im(G)$, se da la siguiente definición.

Definición 3.7 (**Diagrama de Voronoi digitalizado planar**). Sea el diagrama de Voronoi $\mathcal{V} = \{V(p_1), \ldots, V(p_n)\}$ descrito en las Definiciones 3.2, 3.5 y 3.6. Llamamos al conjunto $Im(\mathcal{V}) = \{Im(V(p_1)), \ldots, Im(V(p_n))\}$ el *diagrama de Voronoi digitalizado* (ordinario planar) de $\mathcal{V}$, a $Im(V(p_i))$ la *región de Voronoi digitalizada* y a $Im(\partial V(p_i))$ la frontera de la región de Voronoi digitalizada $Im(V(p_i))$.

Se muestra un ejemplo en la Figura 3.6. Dehne expone un método computacional para el diagrama de Voronoi digitalizado ordinario planar, también presenta un método computacional para el diagrama de Voronoi digitalizado generado por un conjunto de objetos con funciones de distancia convexas [29].

Definición 3.8 (**Diagrama de Voronoi digital planar**). Sea $C = \{c_{ij}, i = 1, \ldots, n_h, j = 1, \ldots, n_v\}$ y sea $P_c = \{p_{c1}, \ldots, p_{cn}\}$ un subconjunto de C, donde $2 \leq n \leq n_v n_h < \infty$, con $p_{ci} \neq p_{cj}$, para $i \neq j, i, j \in I_n$ y $d(c_{ij}, p_{ck})$ denota la

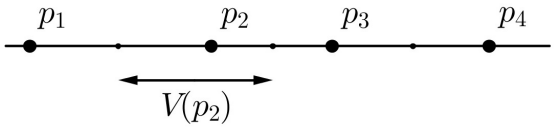


FIGURA 3.5: Diagrama de Voronoi en línea.

distancia Euclidiana entre el centro de c_{ij} y el de p_{ck}. Llamamos a la región

$$\begin{aligned} V(p_{ci}) =& \{c_{kl} | d(c_{kl}, p_{ci}) < d(c_{kl}, p_{cj}), i \neq j, \\ & d(c_{kl}, p_{ci}) = d(c_{kl}, p_{cj}) < d(c_{kl}, p_{cm}), \\ & \text{para } i < j \neq m, k \in I_{n_h}, l \in I_{n_v}\} \end{aligned} \tag{3.11}$$

el *polígono digital de Voronoi* asociado a p_{ci} y a $V(P_c) = \{V(p_{c1}), \ldots, V(p_{cn})\}$ el *diagrama de Voronoi digital* generado por P_c. Notar que la segunda condición en la ecuación 3.11 nos dice que si una celda c_{kl} se encuentra a la misma distancia de la celda generadora p_{ci} y de otra p_{cj}, asignamos, por conveniencia, la celda c_{kl} a la generadora con el índice más pequeño, es decir, a p_{ci}, donde $i < j$ (no ambos como en el diagrama de Voronoi ordinario). Toriwaki y Yokoi [31] y Watanabe y Murashima [32] nombran al diagrama de Voronoi digital como *Diagrama de Voronoi discreto.*

Recordando la Definición 3.2 notamos que $V(p_i)$ está escrito de forma alterna en términos de una función $f_i(x) = \|x - x_i\|$ como

$$V(p_i) = \{x | f_i(x) = \min_{j \in I_n} \{f_j(x)\}\}. \tag{3.12}$$

Geométricamente, $z = f_i(x)$ muestra el cono centrado en x_i en el espacio tridimensional $\{(x_1, x_2, z)\}$ y puesto que $z = f_m in(x) = \min_{j \in I_n}\{f_j(x)\}$ muestra la envolvente inferior de los n conos, $z = f_1(x), \ldots, z = f_n(x)$. Esta envolvente forma una superficie, a la que llamamos *superficie de Voronoi de* $\mathcal{V}$ (ver Figura 3.7).

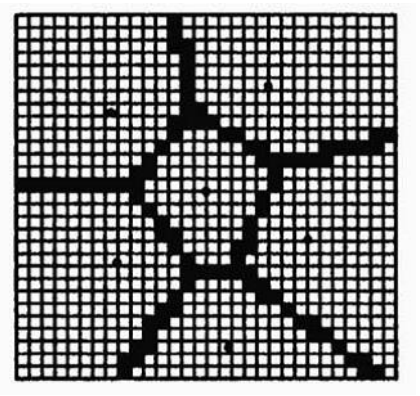


FIGURA 3.6: Diagrama de Voronoi digitalizado ordinario planar.

Alternativamente, podemos reemplazar $f_i(x) = \|x - x_i\|$ por $f_i(x) = \|x - x_i\|^2$ y obtener otra superficie $z = f_{min}(x)$. Esta superficie es diferenciable, excepto en puntos de las fronteras $\bigcup_{i=1}^{n} \partial V(p_i)$. Siersma examina la topología diferencial de esta superficie a través de la teoría de Morse [33].

3.1.1 Propiedades de un diagrama de Voronoi

Propiedad 3.1. Un diagrama de Voronoi es un conjunto de celdas cuyas caras son polígonos convexos (posiblemente no acotados).

Propiedad 3.2. Cada punto en una arista de Voronoi es equidistante de sus dos vecinos más cercanos p_i y p_j. Por lo tanto, existe un círculo centrado en ese punto tal que p_i y p_j se encuentran en este círculo y ningún otro *sitio* (punto) está en el interior de él.

Propiedad 3.3 (**Círculo máximo vacío**). Un vértice de Voronoi (el sitio compartido por tres o más celdas de Voronoi, por ejemplo $V(p_i)$, $V(p_j)$ y $V(p_k)$ es equidistante de los sitios p_i, p_j y p_k. Por lo tanto, es el centro del círculo que pasa por ellos y, adems, no contiene otros sitios en su interior (Figura 3.8).

Propiedad 3.4. Una celda de Voronoi es no limitada si y sólo si el sitio correspondiente se encuentra en la cubierta convexa. Observar que un sitio está en la cubierta convexa si y sólo si es el punto más cercano de algún punto en el infinito, así, dado un diagrama de Voronoi, es fácil extraer la cubierta convexa en tiempo lineal [34].

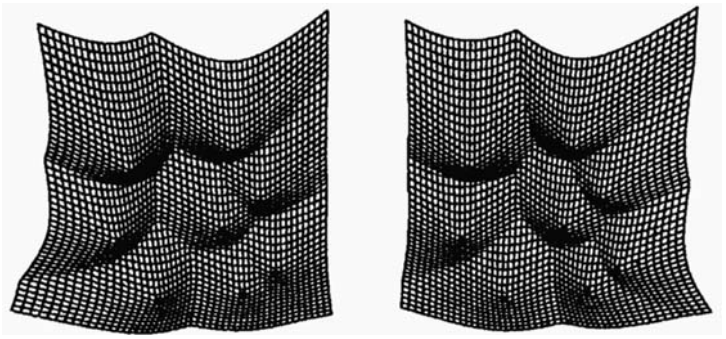


FIGURA 3.7: Superficie de Voronoi.

Propiedad 3.5. Si se tienen n sitios, se desprende de la fórmula de Euler, Teorema 2.4, que el número de vértices de Voronoi es como máximo $2n - 5$ y el número de aristas es a lo más $3n - 6$ (de Berg et al. realiza una demostración en [35]).

3.2 Definiciones de la Triangulación (Teselación) de Delaunay

De la misma manera que un grafo planar (plano) tiene su grafo dual, un diagrama de Voronoi tiene su *teselación* (mosaico) *dual*, llamada *Triangulación de Delaunay* (teselación o mosaico de Delaunay). En esta sección se brindarán los conceptos básicos de lo que es una triangulación de Delaunay.

Definición 3.9 (**Triangulación**). Sea $P = \{p_1, \ldots, p_n\}$ un conjunto de puntos en el plano. Definiremos una *subdivisión maximal* en el plano como una subdivisión S, tal que ningún lado conectado a dos vértices pueda ser añadido a S sin perder su estado plano, es decir, ningún lado de S intersecta con otro lado existente (ver Figura 3.9). Así, una *triangulación* $\mathcal{T}$ es una subdivisión maximal planar cuyo conjunto de vértices es P. Se puede ver de manera sencilla que esta subdivisión está formada por triángulos [35].

Análogamente se define una *triangulación de una nube de puntos* del plano como una partición del cierre convexo en triángulos. La estructura resultante es una familia maximal de triángulos de interiores disjuntos cuyos vértices son puntos de la nube y en cuyo interior no hay algún punto de la nube.

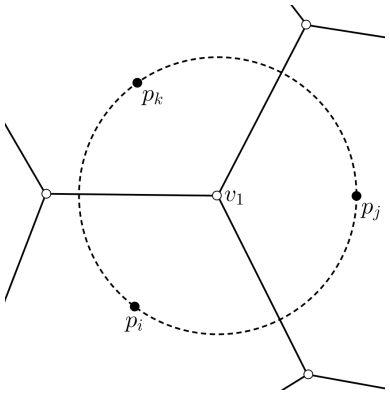


FIGURA 3.8: Círculo máximo vacío centrado en v_1.

Se considera un diagrama de Voronoi en el plano Euclidiano y se asume que los puntos generadores del diagrama de Voronoi no están sobre la misma recta, como se muestra en la Figura 3.10(a). Esta suposición se adoptará de ahora en adelante, nos referiremos a ella como:

Hipótesis 3.2 (**Posición general o hipótesis de no colinealidad**). Para un conjunto de puntos $P = \{p_1, \ldots, p_n\}$ dado, los puntos en P no se encuentran en la misma recta.

Adicionalmente a la hipótesis, asumimos que el número de puntos es tres o más, pero finito. Puede verse fácilmente que la suposición anterior implica que $n \geq 3$ ya que dos puntos siempre están sobre la misma recta. Notar que sin esta suposición es imposible obtener la triangulación y que se puede obtener un diagrama de Voronoi aunque $n = 2$ o cuando los puntos generadores sean colineales (rectas punteadas en la Figura 3.10(a)). Si los sitios no están en *posición general*, en el sentido de que cuatro o más sean cocirculares (ver Figura 3.10(b), entonces la triangulación de Delaunay puede no ser una triangulación, sino sólo un grafo planar.

Bajo la hipótesis de no colinealidad y suponiendo $3 \leq n < \infty$, mostraremos cómo obtener una triangulación de Delaunay a partir de un diagrama de Voronoi.

El grafo dual del diagrama de Voronoi, denotado por $\mathcal{G}$, posee un nodo por cada celda de Voronoi y tiene un arco entre dos nodos si las celdas correspondientes comparten una arista. Notar que $\mathcal{G}$ posee un arco por cada arista de $\mathcal{V}(P)$. Como

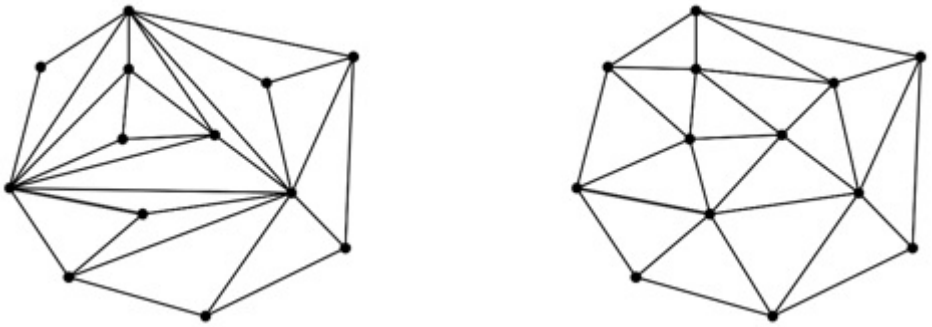


FIGURA 3.9: Diferentes tipos de triangulaciones a partir de una misma nube de puntos.

se puede observar en la Figura 3.11, existe una correspondencia uno a uno entre las caras acotadas de $\mathcal{G}$ y los vértices de $\mathcal{V}(P)$.

Definición 3.10 (**Grafo de Delaunay**). Consideremos las líneas rectas contenidas en $\mathcal{G}$, donde el nodo correspondiente a la celda $V(p_i)$ es el punto p_i y el arco conectando los nodos de $V(p_i)$ y $V(p_j)$ es el segmento $\overline{p_i p_j}$ (ver Figura 3.12). Llamamos a esto el *grafo de Delaunay* de P y lo denotamos como $D\mathcal{G}(P)$.

El grafo de Delaunay de P es una vinculación del grafo dual del diagrama de Voronoi, tiene una cara para cada vértice de $\mathcal{V}(P)$, las aristas alrededor de una cara corresponden a las aristas de Voronoi, encontrándose con los vértices de Voronoi correspondientes. En particular, si un vértice v de $\mathcal{V}(P)$ es un vértice de las celdas de Voronoi para los puntos $p_1, \ldots, p_n$, entonces, la cara correspondiente f (del inglés *face*) en $D\mathcal{G}(P)$ tiene a $p_1, \ldots, p_n$ como vértices. En este caso, los puntos descansan en un círculo alrededor de v, entonces f es un $n-$gono y es convexo (ver Figura 3.2.5). Si un conjunto de puntos se encuentra en *posición general* (Hipótesis 3.2), entonces no contiene cuatro o más puntos en un círculo; si P se encuentra en *posición general* entonces todos los vértices del diagrama de Voronoi son de grado tres y consecuentemente todas las caras acotadas de $D\mathcal{G}(P)$ son triángulos.

De acuerdo con lo anterior, se puede definir una *Triangulación de Delaunay* como cualquier triangulación que se obtiene mediante la adición de aristas al *grafo de Delaunay*. Dado que todas las caras de $D\mathcal{G}(P)$ son convexas, obtener dicha triangulación es fácil. Observar que la triangulación de Delaunay de P es única si y

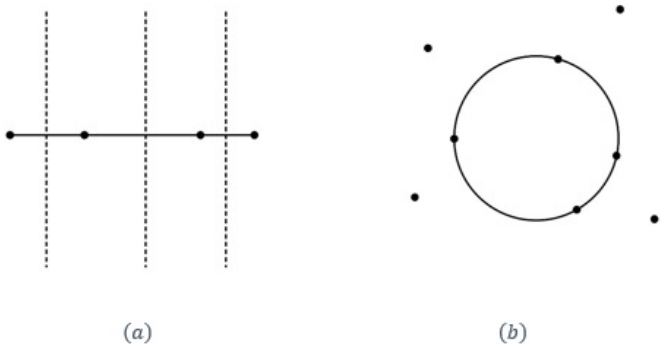


FIGURA 3.10: (a) Puntos generadores colineales. (b) Puntos generadores cocirculares.

sólo si $D\mathcal{G}(P)$ es una triangulación, el cual es el caso si P se encuentra en posición general.

Definición 3.11 (**Triangulación de Delaunay**). Una triangulación $\mathcal{T}$ de un conjunto de puntos $P = p_1, \ldots, p_n$ es considerada una *Triangulación de Delaunay* si se cumple alguna de las siguientes condiciones.

- La triangulación de los n puntos cumple que, para cada arista, existe un círculo que pasa por los puntos extremos de la arista, de modo que los otros puntos en P son exteriores a dicho círculo.
- La triangulación de los n puntos cumple que, para cada triángulo T_i, el interior del círculo definido por los vértices de T_i no contiene puntos en P [36].

Una triangulación de Delaunay maximiza el ángulo mínimo de todos los triángulos que la componen [37].

Recordando que el diagrama de Voronoi de un conjunto de sitios en el plano es una subdivisión planar, el dual de dicha subdivisión es otra subdivisión que se define de la siguiente manera:

1. Para cada cara del diagrama de Voronoi, se crea un vértice (correspondiente al punto).

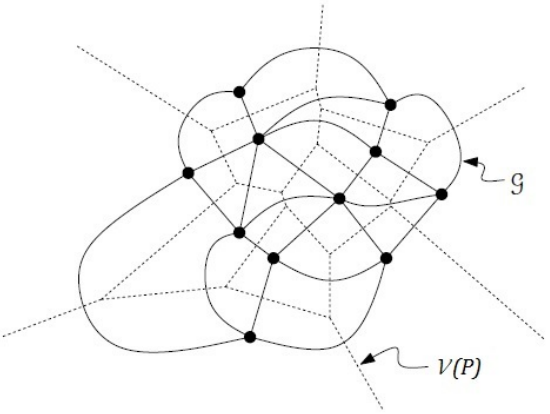


FIGURA 3.11: El grafo de Delaunay $\mathcal{G}$ de un diagrama de Voronoi $\mathcal{V}(P)$

2. Para cada arista del diagrama de Voronoi que cae entre dos sitios p_i y p_j, se crea una arista en el dual conectando estos dos vértices.

3. Finalmente, cada vértice del diagrama de Voronoi corresponde a una cara del dual.

4. El grafo dual resultante es una subdivisión planar, una triangulación de los puntos.

3.2.1 Propiedades de la triangulación de Delaunay

Propiedad 3.6 (**Cubierta convexa**). La frontera de la cara exterior del grafo de Delaunay es la frontera de la cubierta convexa del conjunto de puntos.

Propiedad 3.7 (**Circunferencia circunscrita**). La circunferencia circunscrita de cualquier triángulo en $D\mathcal{G}(P)$ no contiene ningún punto (sitio) de P.

La propiedad anterior se debe a que el centro de ese círculo es el vértice de Voronoi correspondiente y de acuerdo a cómo se define un diagrama de Voronoi, los tres sitios que conforman este vértice son sus vecinos más cercanos.

Propiedad 3.8 (**Círculo vacío**). Dos puntos (sitios) p_i y p_j están conectados por una arista en la triangulación de Delaunay si y sólo si hay un círculo vacío que pasa por p_i y p_j.

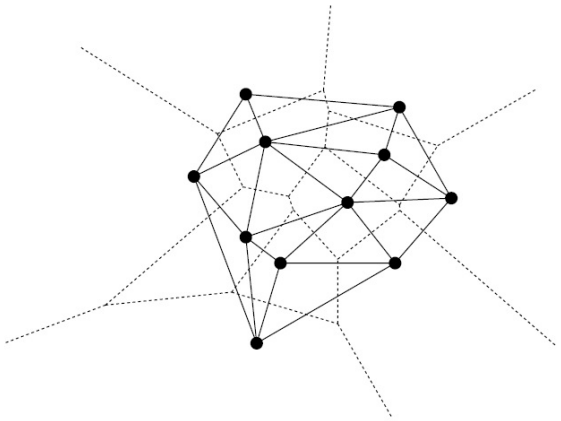


FIGURA 3.12: El grafo dual $D\mathcal{G}(P)$.

Si dos sitios p_i y p_j son vecinos en la triangulación de Delaunay, entonces sus celdas son vecinas en el diagrama de Voronoi y por lo tanto para cualquier punto en la arista de Voronoi entre estos sitios, un sitio centrado en este punto pasando por p_i y p_j no puede contener cualquier otro punto. Por lo tanto, las celdas de Voronoi de dos sitios están adyacentes en el diagrama de Voronoi, implicando que su arista esté en la triangulación de Delaunay.

Propiedad 3.9 (**Par más cercano**). Los pares más cercanos de puntos (sitios) en P son vecinos en la triangulación de Delaunay.

Suponiendo que p_i y p_j son los sitios más cercanos, el círculo que los contiene no puede contener a algún otro (tal sitio estaría más cerca de alguno de esos dos puntos, contradiciendo la hipótesis de que esos puntos son el par más cercano). Por lo tanto el centro de este círculo está en la arista de Voronoi entre estos dos puntos y de esta manera éste es un círculo vacío.

Observación: Dado un conjunto P con n puntos donde hay h de ellos en la cubierta convexa, no es difícil demostrar por la fórmula de Euler, Teorema 2.4, que la triangulación de Delaunay tiene $2n - 2 - h$ triángulos y $3n - 3 - h$ aristas.

Por ejemplo, en la Figura 3.14, se observa que $n = 8$ y $h = 6$, por lo que el número total de triángulos es $2(8) - 2 - 6 = 8$ y el número total de aristas es $3(8) - 3 - 6 = 15$.

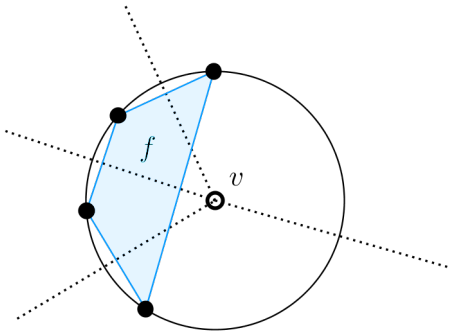


FIGURA 3.13: Cara f en $D\mathcal{G}(P)$, $n-$gono convexo.

La posibilidad de determinar el número de triángulos y aristas sólo funciona en el plano. En $\mathbb{R}^3$ el número de tetraedros en la triangulación de Delaunay puede variar de $O(n)$ hasta $O(n^2)$. En $\mathbb{R}^n$ el número de símplices[3] puede variar hasta $O(n^{\frac{d}{2}})$ [38].

3.3 Construcción

Existen diferentes tipos de algoritmos para la construcción de diagramas de Voronoi, basados en la técnica de *divide y vencerás* descrita por Shamos y Hoey [18], la técnica *línea de barrido* (*sweep-line*) descrita por Fortune [39], o *transformaciones geométricas*, descritas por Brown [40], Edelsbrunner y Seidel [41].

Una aproximación unitaria para los diagramas de Voronoi fue descrita por Klein [42]. Él no usa el concepto de distancia como noción básica, sino más bien el concepto de *curvas bisectrices*, es decir, él asume para cada par $\{p_i, p_j\}$ de sitios la existencia de una curva bisectriz $J(p_i, p_j)$ que divide el plano en una p_i-región y una p_j-región. La intersección de todas las p_i-regiones para diferentes p_j's es entonces la región de Voronoi del sitio p. Él también postuló que las regiones de Voronoi están conectadas simplemente (simples conexas) y particionan el plano.

[3]Simplex: generalización $d-$dimensional de un triángulo.

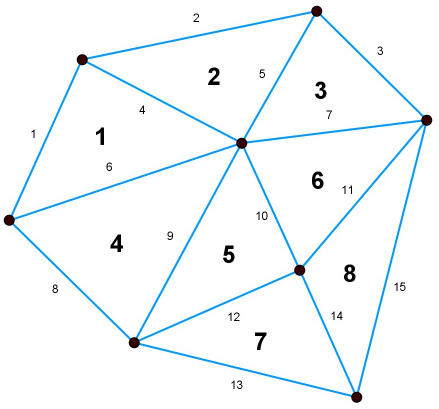


FIGURA 3.14: Número de triángulos y aristas.

La complejidad computacional estudia la eficiencia de los algoritmos estableciendo su efectividad de acuerdo al tiempo de corrida y al espacio requerido en la computadora o almacenamiento de datos, ayudando a evaluar la viabilidad de la implementación práctica en tiempo y costo.

Definición 3.12. Un *algoritmo* es un procedimiento computacional bien definido, el cual considera un valor o conjunto de valores como *valores de entrada* y a través de una secuencia de instrucciones organizadas produce un valor o conjunto de *valores de salida*, en un tiempo determinado, con la finalidad de dar solución a un problema específico.

Las principales características de un algoritmo son:

- Preciso y bien definido.
- Datos de entrada (*Require*).
- Datos de salida (*Ensure*).
- Generalidad.
- Finito.

Al analizar un algoritmo se considera el *tiempo de corrida* y consiste en el número de operaciones elementales que ejecuta en cada paso. Dicho análisis se concentra

generalmente en encontrar el peor caso de *tiempo de corrida*, es decir, el mayor tiempo que tardaráa en obtener los valores de salida.

Definición 3.13 (**Análisis asintótico**). El *análisis asintótico* permite conocer la eficiencia de un algoritmo con base en el tiempo de corrida cuando el tamaño de los datos de entrada es suficientemente grande, de tal forma que las constantes y los términos de menor orden no afectan.

Para expresar la tasa de crecimiento de una función se toma en cuenta al término dominante con respecto a n y se ignoran las constantes. Por ejemplo, la tasa de crecimiento de $k_1 n \log n \sim n \log n$, o la tasa de crecimiento de $k_1 n^2 + k_2 n + k_3 \sim n^2$.

Las notaciones para el tiempo de corrida asintótico de un algoritmo se definen en términos de funciones cuyo dominio es el conjunto de números naturales. La *Notación* O representa una cota asintótica superior, se usa para acotar el peor caso del tiempo de corrida de un algoritmo. La *Notación* Ω representa una cota asintótica inferior, se utiliza para acotar el mejor caso del tiempo de corrida de un algoritmo.

Históricamente se conocieron algoritmos para construir diagramas de Voronoi que corrían en tiempo $O(n^2)$ basados en construcciones elementales, también existe un algoritmo trivial que corre en tiempo $O(n^2 \log n)$, el cual opera construyendo $V(p_i)$ mediante la intersección de $n-1$ semiplanos $H(p_i, p_j)$, para $1 \leq j \leq n$, $i \neq j$.

Sin embargo, existen algoritmos más eficientes que corren en tiempo $O(n \log n)$, los más conocidos serán descritos a continuación.

3.4 Principales algoritmos de construcción del diagrama de Voronoi

3.4.1 Intersección de semiplanos

Recordando la Definición 3.5, un polígono de Voronoi ordinario asociado a p_i,

$$V(p_i) = \bigcap_{j \in I_n \{i\}} H(p_i, p_j)$$

ésta sugiere un método para la construcción inmediata de un diagrama de Voronoi, de hecho, sugiere que cualquier celda de Voronoi se da al conjunto de puntos en común entre los semiplanos $H(p_i, p_j)$, con $i \neq j$. La estrategia nos sugiere una construcción del diagrama celda por celda, o para cada punto se calcula la celda correspondiente como la intersección de semiplanos (ver Figura 3.15). Para llevar a cabo esta intersección se puede utilizar un algoritmo de la siguiente manera:

Algorithm 3.1 Intersecta Semiplano($\mathcal{H}$)

Require: Un conjunto de $\mathcal{H}$ semiplanos
Ensure: La región poligonal convexa

$$C := \bigcap_{H \in \mathcal{H}} H$$

if card($\mathcal{H}$)=1 **then**
  $C \leftarrow$ el único semiplano $H \in \mathcal{H}$
else
  dividir H en dos subconjuntos $\mathcal{H}_1$ y $\mathcal{H}_2$ de magnitud $\frac{n}{2}$ cada uno
end if
$C_1 \leftarrow$ Intersecta Semiplano($\mathcal{H}_1$)
$C_2 \leftarrow$ Intersecta Semiplano($\mathcal{H}_2$)
return $C \leftarrow$ Intersecta Regiones Convexas C_1 y C_2

3.4.2 Algoritmo incremental

La idea es generar celdas de forma incremental. En lugar de hallar la intersección de cada celda con respecto a todas las demás, se llevan a cabo los respectivos

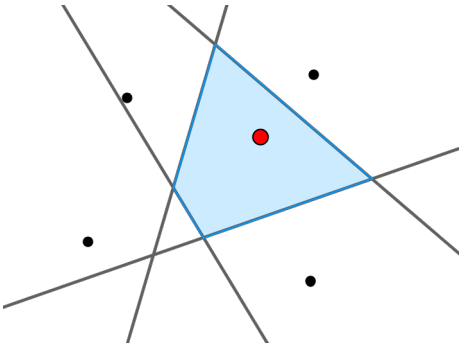


FIGURA 3.15: Intersección de semiplanos.

cálculos sólo con aquellas que son afectadas por una nueva célula que se inserte. En la técnica de barrido de plano [43], esto puede verse como: una vez que se define un ordenamiento de puntos en el plano, se toman uno a uno y se añaden al diagrama (ver Figura 3.16). El diagrama final se construye punto por punto, un primer algoritmo podría ser el siguiente, colocando los eventos en una cola *EventQueue*[4]:

Algorithm 3.2 Voronoi Incremental

Require: Un conjunto de puntos P en el plano
Ensure: Diagrama de Voronoi $\mathcal{V}(P)$
  Ordenar los puntos de P con base a la coordenada X y acomodarlos en una cola *EventQueue*
  while *EventQueue* no está vacío **do**
    Recopilar el siguiente punto
    if p = primer punto **then**
      $\mathcal{V}(P)$ = plano
    else
      Determinar la celda en la que está contenido
      Encontrar el eje entre p y su celda contenedora
      Si el eje influye en otra celda adyacente, encontrar el nuevo eje
      Continuar hasta que sea necesario
      Actualizar el diagrama
    end if
  end while
  return Diagrama obtenido

[4]Fila (cola) de eventos que están esperando a ser procesados por un programa receptor.

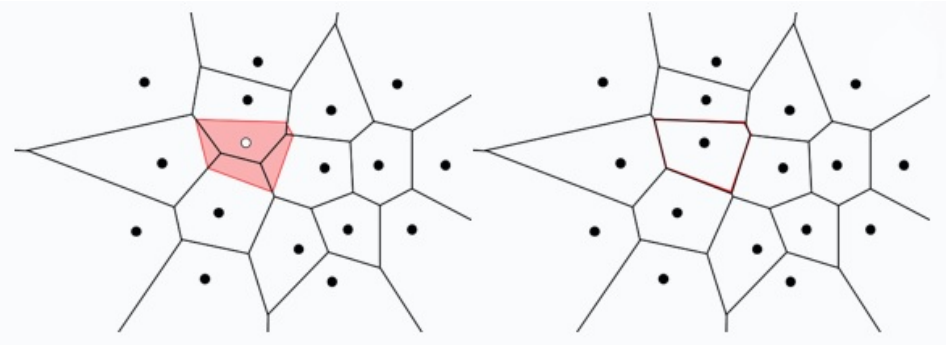


FIGURA 3.16: Algoritmo incremental.

3.4.3 Divide y vencerás

La técnica de *divide y vencerás* consiste en partir un problema grande en problemas pequeños, resolverlos recursivamente y finalmente la solución general se obtendrá al ligar las soluciones de los subproblemas. Suponiendo que se tiene un problema de tamaño n, se aplican los siguientes pasos:

1. Se divide el problema en k subproblemas de tamaño $\frac{n}{k}$.
2. Se resuelven cada uno de los k subproblemas.
3. Se combinan las soluciones de los k subproblemas para obtener la solución general.

La idea del algoritmo consiste en dividir al conjunto P de puntos en el plano en dos subconjuntos P_1 y P_2 mediante una línea vertical L, de tal forma que ambos conjuntos tengan casi el mismo número de puntos. Supongamos que P_1 se encuentra a la izquierda de L y P_2 a su derecha.

Supongamos, además, que se han construido recursivamente los respectivos diagramas de Voronoi para P_1 y para P_2, denotados como $\mathcal{V}(P_1)$ y $\mathcal{V}(P_2)$. Se debe crear un algoritmo para *fundir* (*merge*) $\mathcal{V}(P_1)$ y $\mathcal{V}(P_2)$ para obtener $\mathcal{V}(P)$. Este proceso es un poco complicado y no se abordará aquí, para una mejor referencia, consultar [44].

Denotaremos por σ el subgrafo de Voronoi de un conjunto P, tal que los lados de σ están determinados por parejas de puntos $\{p_i, p_j\}$ tales que $p_i \in \mathcal{V}(P_1)$ y $p_j \in \mathcal{V}(P_2)$; σ es el *grafo frontera*.

Primero se expondrá un algoritmo para construir la cadena de frontera σ que concatenará (encadenará) los diagramas de Voronoi $\mathcal{V}(P_1)$ y $\mathcal{V}(P_2)$. σ posee dos lados que son semirrectas infinitas: uno en la parte *alta*, denotado por L_u y otro en la parte *baja*, denotado por L_d. Para ello, primero se deben unir las *envolventes convexas*[5] de $\mathcal{V}(P_1)$ y $\mathcal{V}(P_2)$, mediante el uso de los algoritmos *Puente Superior* y *Puente inferior* [45].

En la Figura 3.17 puede observarse cómo se aplica este algoritmo. $A(P_1, P_2)$ es el conjunto de aristas que divide a $V(P)$ en dos regiones: $V(P_1)$ y $V(P_2)$.

Algorithm 3.3 Concatenamiento

Require: Conjuntos de Voronoi $\mathcal{V}_1$, $\mathcal{V}_2$, de tamaño $\frac{n}{2}$ cada uno
Ensure: La línea de frontera que los divide
begin

Construir la envolvente convexa de $\mathcal{V}_1$ y $\mathcal{V}_2$

Construir los puentes de unión entre ambas envolventes, usando los algoritmos *Puente Superior* y *Puente inferior*

Construir σ. Hallar las mediatrices L_u del puente superior y L_d del puente inferior

Comenzar en un punto suficientemente alto de L_u y comenzar a bajar, disminuyendo la coordenada Y de los puntos

Continuar bajando, siguiendo la trayectoria de las mediatrices determinadas por los puntos a la derecha de la envolvente de $\mathcal{V}_1$ y los puntos a la izquierda de la envolvente de $\mathcal{V}_2$

Detenerse cuando se intersecte la semirrecta L_d

Eliminar aquellas partes de los vértices de Voronoi que intersectan a σ

Hacer $V(P) = \mathcal{V}(S_1) \bigcup \mathcal{V}(S_2) \bigcup \sigma$

end

[5]De acuerdo con O'Rourke, la envolvente convexa de un conjunto de puntos P es la intersección de todos los semiplanos que contienen a P [47]

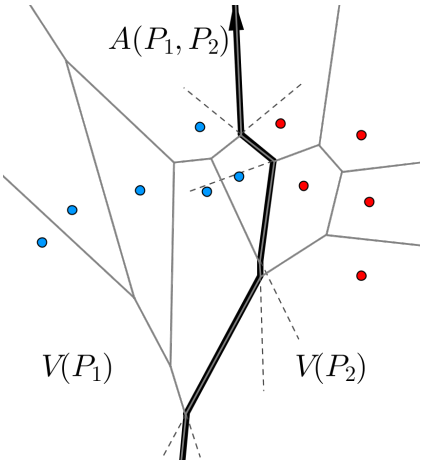


FIGURA 3.17: Algoritmo divide y vencerás.

A continuación se aplica el algoritmo *Divide y vencerás* propiamente dicho, el cual se auxilia del algoritmo de las *Medianas* [46].

Algorithm 3.4 Divide y Vencerás Voronoi

Require: Un conjunto P de n puntos en el plano
Ensure: Diagrama de Voronoi de $\mathcal{V}(P)$
begin

Usando el algoritmo de las *Medianas*, dividir el conjunto P en dos partes iguales, P_1 y P_2 mediante una línea vertical

Recursivamente aplicar el algoritmo de *Concatenamiento* a P_1 y P_2

end

3.4.4 Algoritmo de Fortune

Este algoritmo tiene como principio *barrer* (*sweeping*) con una línea horizontal l (*sweep line*) el plano, desde arriba hacia abajo, la cual desciende sobre cada punto que pasa. Fortune [39] hizo la observación de que podía calcularse el diagrama de Voronoi mediante un barrido del plano construyendo una versión *distorsionada* de éste, pero que es topológicamente equivalente.

Esta nueva versión se basa en una transformación que modifica la forma en que las distancias son medidas en el plano. El diagrama resultante tiene la misma estructura topológica que el diagrama de Voronoi, pero sus aristas son arcos parabólicos, en vez de segmentos de línea recta.

Una vez que este diagrama distorsionado es obtenido, puede ser corregido en tiempo $O(n)$ para producir el diagrama de Voronoi correcto dentro de un rectángulo en una estructura $DCEL$[6].

Algorithm 3.5 Fortune(P)

Require: $P := \{p_1, \ldots, p_n\}$ un conjunto de n puntos en el plano
Ensure: Diagrama de Voronoi $\mathcal{V}(P)$ dentro de un rectángulo en una estructura $DCEL$
Inicializar la cola de eventos *EventQueue* Q con todos los *SiteEvents*
while *EventQueue* Q no está vacío **do**
  Considerar el evento con mayor coordenada Y de Q
  if el evento es un *Site Event* del punto p_i **then**
    Tratar_Site_Event(p_i)
  else
    Tratar_Circle_Event(p_c), donde p_c es el punto más bajo del círculo que causa el evento
  end if
  Eliminar el evento de Q
end while
Los nodos internos presentes todavía en T (el árbol binario) pertenecen a los medios-lados infinitos del diagrama de Voronoi. Calcular un rectángulo que contenga todos los vértices del diagrama de Voronoi en su interior, uniéndole los medios-lados infinitos, modificando la estructura $DCEL$ apropiadamente.
return Diagrama obtenido

Los procedimientos *Tratar_Site_Event*(p_i)[7] y *Tratar_Circle_Event*(p_c) han sido descritos a detalle por M. Abellanas en [48], la implementación del algoritmo se ilustra en la Figura 3.18.

[6]De acuerdo con Abellanas, siglas inglesas de la *lista de lados doblemente enlazados* (*Doubly Connected Edge List*): vértices, caras y lados [48].

[7]Evento de Punto: cuando la línea de barrido pasa justamente por encima de uno de los puntos de la nube [48].

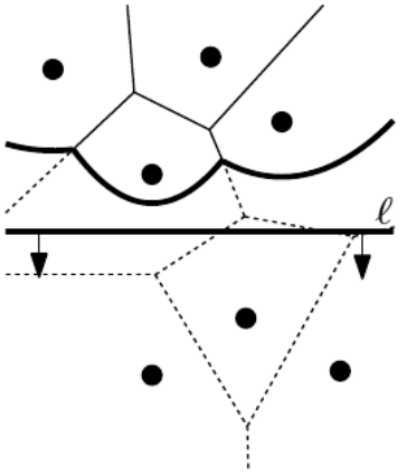


FIGURA 3.18: Algoritmo de Fortune.

3.5 Principales algoritmos de construcción de la triangulación de Delaunay

3.5.1 Algoritmo Incremental

Consiste en comenzar con un triángulo inicial, el cual contiene a todos los puntos de un conjunto P. A continuación se inserta cada uno de los vértices en la triangulación. El siguiente algoritmo fue consultado en [49].

Algorithm 3.6 Delaunay Incremental

Require: Vértice v, triangulación $\mathcal{T}$
Ensure: Triangulación de Delaunay de P
  Triángulo $t_p = locate(v, T)$
  Triángulo[ ] Nuevos_Triángulos = $Split(t_p, v)$
  $\mathcal{T}.agregar\_Triángulos$(Nuevos_Triángulos)
  Cola Triángulos_Pendientes
  Triángulos_Pendientes.$push$(nuevos_Triángulos)
  while !Triángulos_Pendientes.$Cola\_Vacía$( ) **do**
    Triángulo t_i = Triángulos_Pendientes.pop( )
    if !$es\_Válido(t_i, v)$ **then**
      Triángulo[ ] a_Verificar = $Flip(t_i, v)$
      Triángulos_Pendientes.$push$(a_Verficar)
    end if
  end while

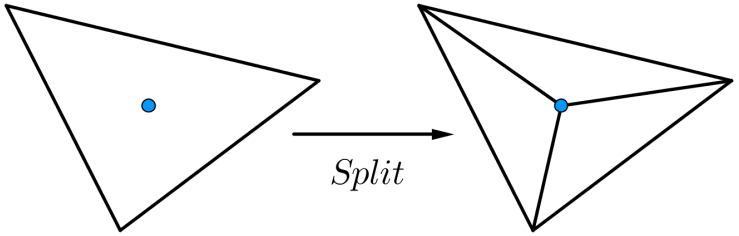


FIGURA 3.19: Operación *Split* sobre un triángulo y un punto en su interior.

El algoritmo parte buscando el triángulo t_p que contiene al vértice a insertar v. Una vez localizado, el procedimiento *Split* lo divide en tres triángulos resultantes de agregar tres nuevas aristas, desde los vértices de t_p a v (ver Figura 3.19).

Los triángulos resultantes deben ser verificados tal que cumplan con el *test del círculo*. Si la triangulación $\mathcal{T}$ es de Delaunay, entonces sólo habrá que comprobar que el vértice del triángulo adyacente al triángulo que se está verificando (opuesto a la arista por la que ambos son vecinos) no está dentro del círculo definido por los vértices de dicho triángulo. La arista por la cual estos triángulos son vecinos es la opuesta al vértice insertado v (ver Figura 3.20).

Si algún triángulo no cumple con la verificación, se aplica la operación *Flip* sobre la arista opuesta al vértice v (ver Figura 3.21). Ésta consiste en intercambiar la arista opuesta a v en el triángulo por la arista $\{v, c\}$. Los nuevos triángulos que resulten de dicha operación deben ser añadidos a la *cola* de triángulos por verificar. La inserción de v termina cuando la *cola* de triángulos por verificar está vacía.

3.5.2 Divide y vencerás

Construye de manera recursiva la triangulación de Delaunay de un conjunto de puntos, construyendo primero la triangulación de Delaunay de dos mitades del conjunto; el algoritmo presentado fue descrito por Guibas y Stolfi [50].

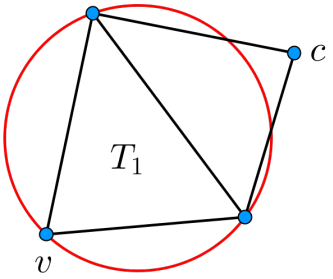


FIGURA 3.20: Verificación del triángulo T_1.

Para dividir el conjunto de puntos a triangular y generar instancias más pequeñas del problema, se ordenan los puntos según su primera coordenada. Una vez ordenados los puntos, el conjunto es dividido en dos mitades según la primera coordenada de los puntos (digamos L y R, ver Figura 3.22). Con cada una de las mitades se construye la triangulación de Delaunay correspondiente a los puntos de esa mitad. Primero se presenta el algoritmo general y en seguida el algoritmo de unión de las dos triangulaciones.

Algorithm 3.7 Divide Delaunay

Require: Un conjunto P de n puntos en el plano (**int** *low*, **int** *high*)
Ensure: Triangulación de Delaunay de P

```
Size = high − low + 1
if Size == 2 then
    return addEdge(low, high)
else if Size == 3 then
    return addFace(low, high)
end if

middle = low + Size/2
ldi = middle − 1
rdi = middle

divideAndConquerDelaunay(low, ldi)
divideAndConquerDelaunay(rdi, high)

merge(low, ldi, rdi, high)
```

Queda resolver cómo realizar la unión de dos triangulaciones que sean correspondientes a las dos mitades de un conjunto de puntos (ver Figura 3.23).

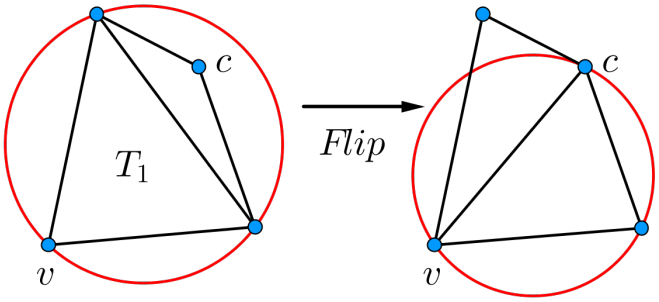


FIGURA 3.21: Operación *Flip* con respecto al vértice v en el triángulo T_1.

Algorithm 3.8 Procedimiento `merge`

Require: L y R (**int** *ldo*, **int** *ldi*, **int** *rdi*, **int** *rdo*)
Ensure: Unión de L y R
  rBase = *rdi*
  lBase = *ldi*
  while *true* **do**
    if *right*(rBase, *nextCCW*(rBase), lBase) **then**
      rBase = *nextCCW*(rBase)
    else if *left*(lBase, *nextCW*(lBase), rBase) **then**
      lBase = *nextCW*(*lBase*)
    else
      break
    end if
  end while

  `addEdge`(rBase, lBase)

  while *true* **do**
    lCand = `getLeftCandidate`(rBase, lBase)
    rCand = `getRightCandidate`(rBase, lBase)
    if !*is_Valid*(lCand) && !*is_Valid*(rCand) **then**
      break
    end if
    if !*is_Valid*(lCand) —— (!*is_Valid*(rCand) && *inCircle*(lBase, rBase, lCand, rCand)) **then**
      `addFace`(lBase, rBase, rCand)
      rBase = rCand
    else
      `addFace`(lBase, rBase, lCand)
      lBase = lCand
    end if
  end while

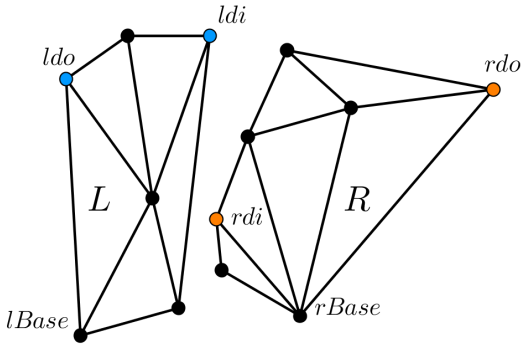


FIGURA 3.22: Las mitades L, R y sus respectivos índices.

El procedimiento de unión comienza buscando la arista $\{lBase, rBase\}$, denotada como $L - R$ Base, desde la cual se irán añadiendo nuevos triángulos [50]. Para encontrar la arista $L - R$ Base, se recorre la mitad derecha, R, partiendo desde rdi en sentido CCW y la mitad izquierda, L, partiendo desde ldi en sentido CW.

El test del círculo vacío considera los triángulos definidos por la arista $L-R$ Base y ambos candidatos. Se escoge el candidato que cumpla con el test (ver Figura 3.24).

La actualización de la $L - R$ Base, luego de la primera iteración se aprecia en la Figura 3.25.

El siguiente algoritmo es utilizado para encontrar el vértice candidato en L (se omite el algoritmo análogo para R, en donde cambia el sentido en el que se recorren los posibles candidatos).

Algorithm 3.9 Borrar arista

Require: **int** rBase, **int** lBase
Ensure: Vértice candidato válido
  lCand = `nextNeighbourCCW`(lBase)
  if *is_Valid*(lCand) **then**
    lNextCand = `nextCandidateCCW`(lCand)
    while *inCircle*(lBase, rBase, lCand, lNextCand) **do**
      `deleteEdge`(lBase, lCand)
      lCand = lNextCand
      lNextCand = `nextCandidateCCW`(lNextCand)
    end while
    return lCand
  end if

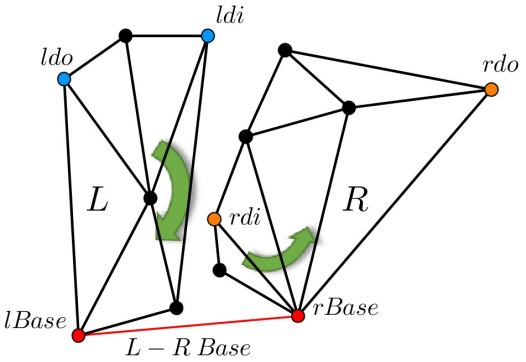


FIGURA 3.23: La arista $L-R$ Base para la unión de L y R. Las flechas indican el sentido en el que el algoritmo recorre cada mitad para determinar la arista $L-R$ Base.

El algoritmo verifica que el vértice candidato sea válido y que el círculo definido por el vértice candidato, lBase y rBase, no contenga al próximo candidato. Si la segunda condición no se cumple, entonces la arista que une lBase con el vértice candidato es eliminada y el vértice candidato es descartado, ver Figura 3.26.

3.6 Complejidad computacional

3.6.1 Intersección de semiplanos

La intersección de semiplanos de dos polígonos convexos se puede realizar en un tiempo proporcional a (n) usando la técnica de *barrido de plano* [43]. En el ejemplo propuesto, C_1 y C_2 pueden no ser necesariamente regiones poligonales, de hecho pueden ser no acotadas, pero también degenerar en una línea recta o en un punto. Sin embargo, esto no tiene algún efecto sobre el resultado. Entonces, tenemos:

$$\begin{aligned} T(n) &= O(1) && \text{si } n = 1 \\ T(n) &= O(n) + 2gT(\tfrac{n}{2}) && \text{si } n > 1 \end{aligned} \tag{3.13}$$

Resolviendo, tenemos que $T(n) = O(ng\log^2 n) = O(nlog^2 n)$. Resultado que implica un tiempo de cálculo para el diagrama de Voronoi completo proporcional a $O(n^2 \log n)$.

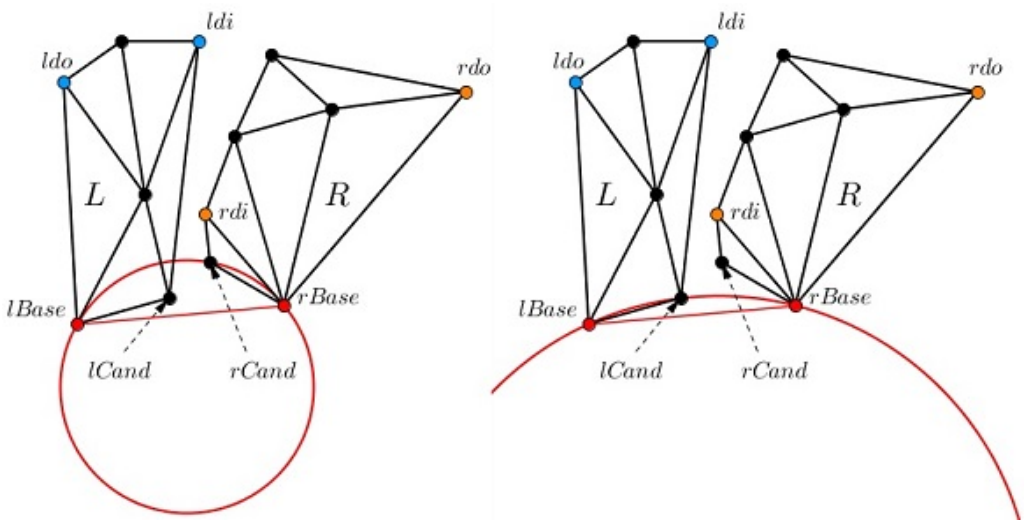


FIGURA 3.24: Vértices candidatos lCand y rCand, se escoge lCand ya que cumple el test del círculo vacío.

3.6.2 Algoritmo incremental

Este algoritmo tiene una complejidad en el caso promedio del tipo $O(n \log n)$.

3.6.3 Divide y vencerás

Si el tiempo para resolver el problema de tamaño s es $T(s)$, hay que tener en cuenta que el tiempo para resolver el problema de tamaño 1 es constante: $T(1) = b$. Los pasos (1) y (2) de la técnica de divide y vencerás se pueden ejecutar en tiempo cn, c constante, cada subproblema de tamaño $\frac{n}{k}$ se resuelve en tiempo $T(\frac{n}{k})$. Entonces se tiene la siguiente relación de recurrencia.

$$T(1) = b \tag{3.14}$$

$$T(n) = kT(\frac{n}{k}) + cn \tag{3.15}$$

Se puede hallar un estimado del tiempo total explícitamente. En el peor caso, dividimos por la mitad cada problema, obteniendo las siguientes ecuaciones.

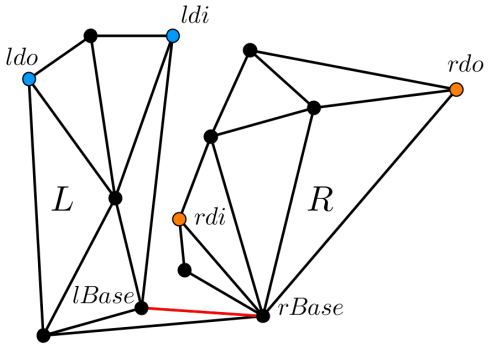


FIGURA 3.25: Actualización de la arista $L - B$ Base.

$$
\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + cn && (3.16)\\
T\left(\frac{n}{2}\right) &= 2T\left(\frac{n}{4}\right) + c\frac{n}{2} && (3.17)\\
&\vdots \\
T\left(\frac{n}{2^s}\right) &= 2T\left(\frac{n}{2^{s-1}}\right) + c\frac{n}{2^s} && (3.18)\\
T(1) &= b && (3.19)
\end{aligned}
$$

Agrupando, obtenemos $T(n) = 2^s + cns = nb + cn\log n$.

Por lo tanto,

$$T(n) = O(n\log n) \tag{3.20}$$

3.6.4 Algoritmo de Fortune

Operaciones primitivas, tales como la búsqueda de un elemento en el árbol o en la cola o la eliminación de eventos que ya no se pueden ejecutar, pueden ser realizadas en tiempo $O(\log n)$. En cada evento, realizamos un número constante de estas operaciones.

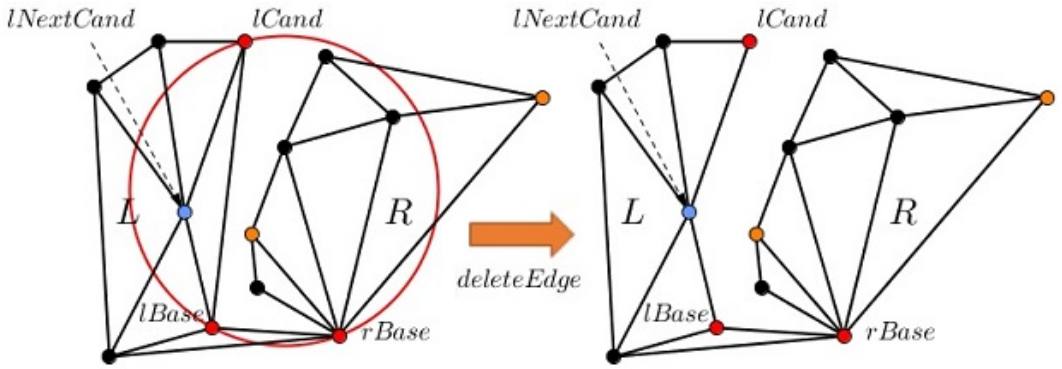


FIGURA 3.26: El círculo definido por lCand, lBase y rBase contiene a lNextCand, por lo cual la arista que une lCand con lBase es eliminada.

El número de eventos de punto (*site events*) es N, el de eventos de círculo (*circle events*) es a lo sumo $2N - 5$. En cada uno realizamos c operaciones primitivas. Por lo tanto, la complejidad final es $O(n \log n)$.

3.6.5 Algoritmo incremental, triangulación de Delaunay

1. Tomar un punto de la nube aún no triangulada. Los puntos serán elegidos en el mismo orden en que son introducidos por el usuario, por lo que para una nube de n puntos esto demoraría $O(n)$.

2. Averiguar el triángulo o la arista sobre la que cae el punto introducido, por lo que habrá que recorrer toda la lista de triángulos existente. Esto supone un tiempo de $O(n \log n)$.

3. Trazar una arista desde el punto a triangular a cada uno de los tres vértices del triángulo que lo contiene. Trazar una arista requiere un tiempo de $O(1)$, para n puntos, requiere $O(n)$.

4. Averiguar si las nuevas aristas son legales (comprobar la propiedad del círculo vacío), por lo que el tiempo en realizar esta comprobación para n puntos es de $O(n)$.

5. Si alguna de las nuevas aristas es ilegal hay que borrarla y crear una legal, esta operación se puede hacer en un tiempo constante $O(1)$, por lo que el tiempo en realizar esto para n puntos es de $O(n)$.

Por lo tanto $T(n) = O(n) + O(n \log n) + O(n) + O(n) + O(n)$

Es decir, $T(n) = O(n \log n)$.

Capítulo 4

Optimización basada en diagramas de Voronoi

En los últimos años, ha ocurrido un impulso en la comunidad científica informática para desarrollar metodologías automatizadas para hallar la solución de problemas complejos. Un elemento clave para ello es la generación de *mallas* (del inglés *mesh*, ver Figura 4.1), que son óptimas en el sentido relacionado de minimizar el costo de resolver el problema complejo. La búsqueda de metodologías para la generación de mallas ha motivado numerosos estudios de algoritmos rápidos, automáticos y óptimos para la creación de mallas de calidad [51].

Definición 4.1 (**Malla**). Una *malla* es una estructura geométrica lineal incrustada a trozos en $\mathbb{R}^n$.

Definición 4.2 (**Malla triangular**). Una *malla triangular* es un tipo de malla poligonal que comprende un conjunto de triángulos que se conectan por sus lados o vértices más comunes y aproximan una superficie.

En esta sección, estudiaremos las mallas triangulares en tres dimensiones y, a menos que se indique lo contrario, nos referiremos a ellas sólo como mallas. Aunque el campo de aplicación de la generación automática de mallas triangulares ha sido tradicionalmente la obtención de modelos digitales de elevaciones del terreno, hoy

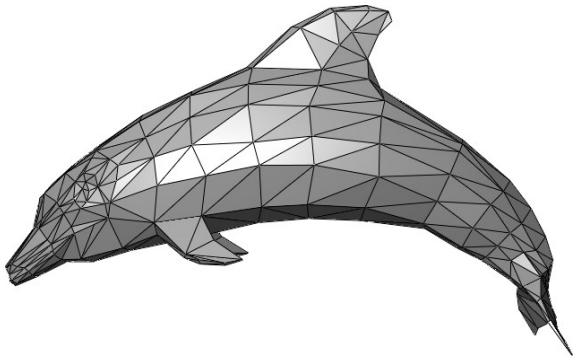


FIGURA 4.1: La malla contiene una estructura combinatoria (no una *sopa* de triángulos).

en día son usadas en *renderización*[1], animación, realidad virtual y el análisis de elementos finitos [52].

Las mallas se clasifican regularmente como *estructuradas* y *no estructuradas* [53]. Aunque también se aplica a las primeras, esta investigación se centrará en las segundas. Un método muy popular para la generación de mallas no estructuradas son las *Triangulaciones de Voronoi-Delaunay* [54]. Dado un conjunto de entrada de puntos $P = \{p_1, \ldots, p_k\} = \{p_i\}_{i=1}^{k}$ que pertenece al dominio $\Omega \in \mathbb{R}^n$, existe un diagrama de Voronoi $\mathcal{V} = V(p_1), \ldots, V(p_k) = \{V_1, \ldots, V_k\} = \{V_i\}_{i=1}^{k}$ que forma una partición de Ω y sus puntos generadores son $\{p_i\}_{i=1}^{k}$. Su dual (la triangulación de Delaunay), está formado mediante la conexión de los puntos generadores que corresponden a las regiones de Voronoi adyacentes.

En el contexto de generación de mallas no estructuradas, las triangulaciones de Delaunay se han usado como un buen punto de partida. Pueden contener triángulos con ángulos pequeños, o triángulos con áreas muy variadas, por lo que la mayoría de los algoritmos relacionados con las triangulaciones de Voronoi-Delaunay no proporcionan una garantía acerca de la calidad de la malla resultante. Otro factor influyente es la distribución de los puntos generadores, para hallar una buena distribución de puntos, muchas técnicas han sido propuestas como el método *de frente de avance* (*advancing front method* [55]), el método *de empaquetamiento de esferas* (*the method of sphere packing* [56]) y el método *de inserción sucesiva de puntos de*

[1]Generación de una imagen a partir de un modelo con ayuda de programas computacionales (CADs).

Steiner (*the method of successive insertion of Steiner points* [57]). Estas técnicas pueden producir mallas que satisfacen algunas restricciones de calidad, por ejemplo, el ángulo mínimo puede ser mayor que un límite inferior proporcionado por el usuario [51].

4.1 Teselaciones de Voronoi Centroidales (TVC)

Las Teselaciones de Voronoi Centroidales, denotadas como TVC (del inglés *Centroidal Voronoi Tessellations*, CVT) son un tipo especial de teselación, que ofrecen propiedades superiores comparadas con las ordinarias. Sus puntos generadores son también *centroides*[2] de las regiones de Voronoi asociadas, con respecto a una función de densidad dada y un *error funcional*. Por lo tanto, se puede esperar que una Triangulación de Delaunay basada en Teselaciones de Voronoi Centroidales (TDVC) puede proporcionar mejores alternativas a los métodos existentes para la generación de mallas de alta calidad [58].

Definición 4.3 (**Centroide**). Dada una partición $\{V_i\}_{i=1}^{k}$ de $\Omega \in \mathbb{R}^n$ y una función de densidad[3] $\rho(x)$ definida para cada $x \in \Omega$, el *centroide* (centro de masa, ver Figura 4.2) de cada celda V_i es el punto $\overline{z_i} \in \mathcal{V}$, definido por

$$\min_{z_i \in \mathcal{V}} \left\{ \int_{\mathcal{V}} \rho(x)||x - z_i||^2 dx \right\} \tag{4.1}$$

Entonces, la forma explícita de $\overline{z_i}$ está dada por

$$\overline{z_i} = \frac{\int_{\mathcal{V}} x\rho(x)dx}{\int_{\mathcal{V}} \rho(x)dx} \tag{4.2}$$

Definición 4.4 (**TVC**). Si los generadores de las regiones $\{V_i\}_{i=1}^{k}$ de Ω coinciden con sus correspondientes centroides, es decir,

[2]Un centroide puede entenderse como la posición media de todos los puntos en la totalidad de las direcciones de coordenadas.

[3]La función de densidad está basada en una función de distribución de densidad [59].

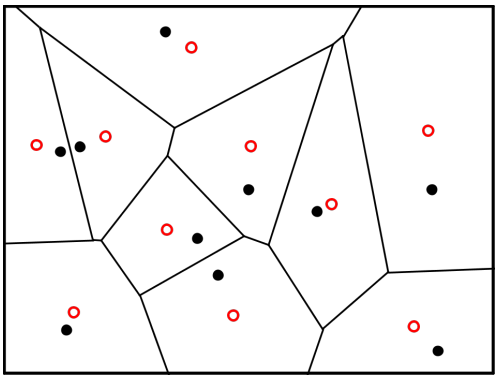


FIGURA 4.2: Diagrama de Voronoi de un rectángulo con 10 puntos aleatorios ((○) son los centroides, (•) son los generadores) y la función de densidad es una constante. Notar que los generadores y los centroides no coinciden.

$$z_i = \overline{z_i}, \qquad \text{para } i = 1, \ldots, k \tag{4.3}$$

entonces llamamos a la teselación de Voronoi $\{V_i\}_{i=1}^k$ una *Teselación de Voronoi Centroidal* (TVC) de Ω y $\{z_i\}_{i=1}^k$ son los *generadores* de TVC que corresponden. El dual concerniente se conoce como *Triangulación de Delaunay basada en teselaciones de Voronoi Centroidales* (TDVC).

Notar que para un dominio dado Ω, la TVC asociada puede no ser única [58]. Por lo tanto, determinar una TVC de Ω es un proceso para encontrar un conjunto de generadores $\{z_i\}_{i=1}^k$ tal que son al mismo tiempo centroides de las regiones de Voronoi asociadas $\{V_i\}_{i=1}^k$.

Definición 4.5 (**Error funcional**). Para cualquier teselación $\{V_i\}_{i=1}^k$ del dominio Ω y un conjunto de puntos $\{z_i\}_{i=1}^k$ en Ω, se define el *error* (costo, energía [60]) *funcional*:

$$\mathcal{F}(\{V_i\}_{i=1}^k, \{z_i\}_{i=1}^k) = \sum_{i=1}^{k} \int_{\mathcal{V}} \rho(x)||x - z_i||^2 dx \tag{4.4}$$

Las TVC estándar, junto con sus generadores son puntos críticos de este error funcional.

En la práctica, las posiciones de los generadores pueden estar limitadas por varias restricciones. Por ejemplo, en el contexto de generación de una malla, puede ser necesario que un cierto número de generadores se encuentren en la frontera de

Ω. Esto motiva otra generalización del concepto de TVC: La TVC *restringida*, definida como el mínimo de $\mathcal{F}$ bajo algunas restricciones específicas.

Definición 4.6 (**TVCR**). Dado un conjunto de puntos $\{z_i\}_{i=1}^k$ en Ω, una función de densidad ρ y un conjunto de restricciones R, una teselación de Voronoi es llamada *Teselación de Voronoi Centroidal Restringida* (TVCR) si $(\{V_i\}_{i=1}^k, \{z_i\}_{i=1}^k)$ es una solución del problema

$$\min_{\{z_i\}_{i=1}^k \in R, \{V_i\}_{i=1}^k} \left\{ \mathcal{F}(\{V_i\}_{i=1}^k, \{z_i\}_{i=1}^k) = \sum_{i=1}^k \int_{\mathcal{V}} \rho(x)||x - z_i||^2 dx \right\} \tag{4.5}$$

La triangulación de Delaunay relacionada se conoce como *Triangulación de Delaunay basada en teselaciones de Voronoi Centroidales Restringidas* (TDVCR).

4.2 Algoritmos para TDVCs

Usualmente existen dos componentes en los algoritmos para hallar TDVCs, siendo el primero el cálculo de los generadores de la TVC y el segundo es la construcción de la correspondiente triangulación de Delaunay de los generadores. Una vez hallados los generadores, la segunda componente se puede realizar usando algoritmos de triangulación de Delaunay estándar. Nos centraremos en los métodos para la búsqueda de los generadores de TVCs. Se discuten dos de estos métodos, que son los representantes más simples de dos clases de enfoques: *probabilístico* y *determinístico*.

4.2.1 Enfoque probabilístico (McQueen)

Nos centramos en el *método aleatorio de McQueen*, el cual tiene la ventaja de que no se requiere el cálculo de la teselación de Voronoi hasta el paso final. Para otros estudios del algoritmo, consultar [61] y [62]. Se emplea el método de Montecarlo para generar un conjunto de puntos inicial [63]. El método de McQueen está definido como sigue.

Algorithm 4.1 Método de McQueen

Require: Un conjunto Ω, un entero positivo k y una función de densidad de probabilidad ρ definida en $\overline{\Omega}$

Ensure: TDVC de Ω

begin

1. *Elegir* un conjunto inicial de k puntos $\{z_i\}_{i=1}^{k}$ en Ω, usando un método de Montecarlo
2. *Inicializar* el índice $j_i = 1$ para toda $i = 1, \ldots, k$
3. *Seleccionar* un punto $y \in \Omega$ al azar de acuerdo con la función de densidad de probabilidad $\rho(y)$
4. *Hallar* el punto z_i en $\{z_i\}_{i=1}^{k}$ más cercano a y
5. *Denotar* el índice de z_i como i^*
6. *Fijar*
$$z_{i^*} \leftarrow \frac{j_{i^*} z_{i^*} + y}{j_{i^*} + 1} \qquad \text{y} \qquad j_{i^*} \leftarrow j_{i^*} + 1$$
7. El nuevo z_{i^*}, junto con $\{z_i\}_{i \neq i^*}$, *forman* el nuevo conjunto $\{z_i\}_{i=1}^{k}$

if El nuevo $\{z_i\}_{i=1}^{k}$ cumple algún criterio de convergencia **then**
Hallar la correspondiente triangulación de Delaunay y terminar
else
Volver al paso 3
end if
end

Notar que el índice j_i representa el número de veces que el punto z_i ha sido actualizado.

El análisis de la complejidad computacional de este algoritmo es extenso, no será revisado en esta sección. Ju, Du y Gunzburger han realizado experimentos numéricos, los cuales pueden consultarse en [64].

4.2.2 Enfoque determinístico (Lloyd)

Se describe un algoritmo basado en el *método de Lloyd*, que es una iteración entre la construcción de teselaciones de Voronoi y los centroides. También hace uso del método de Montecarlo para generar un conjunto de puntos inicial.

Algorithm 4.2 Método de Lloyd

Require: Un conjunto Ω, un entero positivo k y una función de densidad de probabilidad ρ definida en $\overline{\Omega}$
Ensure: TDVC de Ω
begin

1. *Elegir* un conjunto inicial de k puntos $\{z_i\}_{i=1}^{k}$, por ejemplo, mediante el uso de un *método de Montecarlo*
2. *Construir* la teselación de Voronoi $\{V_i\}_{i=1}^{k}$ de Ω, asociada con los puntos $\{z_i\}_{i=1}^{k}$
3. *Calcular* los centroides de las regiones de Voronoi halladas
4. Estos centroides son el nuevo conjunto de puntos $\{z_i\}_{i=1}^{k}$

if El nuevo $\{z_i\}_{i=1}^{k}$ cumple algún criterio de convergencia **then**
Encontrar la triangulación de Delaunay y terminar
else
Volver al paso 2
end if
end

El método de Lloyd original fue usado sólo para encontrar la TVC y puede ser interpretado como una iteración de punto fijo para el mapeo entre los generadores y los centroides [65]. Una discusión más detallada puede encontrarse en [58]. El análisis de la complejidad computacional de este algoritmo tampoco será revisado en esta sección. Ju, Du y Gunzburger han realizado experimentos numéricos, los cuales pueden consultarse en [64].

4.2.3 Algoritmos para TDVCs restringidos (TDVCRs)

Existen varias generalizaciones para la construcción de TDVCRs. En la configuración más simple del caso unidimensional, dos de los generadores pueden limitarse a ser los puntos límite del intervalo $\Omega = [a, b]$. Du y Gunzburger proponen una modificación al método de McQueen [66](Algoritmo 4.3), así mismo se realizaron experimentos numéricos en [64] para analizar la complejidad computacional del algoritmo.

Algorithm 4.3 Método de McQueen Modificado

Require: Un conjunto $\Omega = [a, b]$, un entero positivo k y una función de densidad de probabilidad ρ definida en $\overline{\Omega}$

Ensure: TDVCR de Ω

begin

1. *Hacer* $z_1 = a$ y $z_k = b$
2. *Elegir* un conjunto inicial de $k-2$ puntos $\{z_i\}_{i=1}^{k}$, usando un *método de Montecarlo*, pertenecientes a un conjunto de restricciones R
3. *Fijar* $j_i = 1$ para toda $1 < i < k$
4. *Seleccionar* $y \in \Omega$ al azar de acuerdo con $\rho(y)$
5. *Hallar* el punto z_i en $\{z_i\}_{i=1}^{k}$ más cercano a y
6. *Denotar* el índice de z_i como i^*

if $i^* = 1$ o $i^* = k$ **then**
Volver al paso 4
else
Fijar

$$z_{i^*} \leftarrow \frac{j_{i^*} z_{i^*} + y}{j_{i^*} + 1} \qquad \text{y} \qquad j_{i^*} \leftarrow j_{i^*} + 1$$

end if
El nuevo z_{i^*}, junto con $\{z_i\}_{i \neq i^*}$, *forman* el nuevo conjunto $\{z_i\}_{i=1}^{k}$
if El nuevo $\{z_i\}_{i=1}^{k}$ cumple algún criterio de convergencia **then**
Hallar la correspondiente triangulación de Delaunay y terminar
else
Volver al paso 4
end if
end

Otras generalizaciones a conjuntos de restricciones más complicados pueden ser realizadas análogamente [64].

Similarmente, dado conjunto de restricciones R, Du y Gunzburger proponen una modificación al método de Lloyd [66](Algoritmo 4.4).

El algoritmo de Lloyd también tiene la propiedad de que $\mathcal{F}$ es *monótona decreciente* a lo largo de la iteración.

Algorithm 4.4 Método de Lloyd Modificado

Require: Un conjunto Ω, un entero positivo k y una función de densidad de probabilidad ρ definida en $\overline{\Omega}$

Ensure: TDVCR de Ω

begin

1. *Elegir* un conjunto inicial de k puntos $\{z_i\}_{i=1}^k$, mediante el uso de un *método de Montecarlo*, pertenecientes a R
2. *Construir* la teselación de Voronoi $\{V_i\}_{i=1}^k$ de Ω, asociada con los puntos $\{z_i\}_{i=1}^k$
3. *Calcular*, en el conjunto R, el *mínimo* de

$$\mathcal{F}(\{V_i\}_{i=1}^k, \{z_i\}_{i=1}^k) = \sum_{i=1}^{k} \int_{\mathcal{V}} \rho(x)||x - z_i||^2 dx \tag{4.6}$$

4. El mínimo es el nuevo conjunto de puntos $\{z_i\}_{i=1}^k$

if El nuevo $\{z_i\}_{i=^1}^k$ cumple algún criterio de convergencia **then**
Encontrar la triangulación de Delaunay y terminar
else
Volver al paso 2
end if
end

4.3 Aplicaciones de TDVCs para la generación de mallas

Du, Faber y Gunzburger han comentado muchas aplicaciones de las TVC, como las *reglas de cuadratura óptima*, *representación óptima*, *cuantificación*, *agrupación* (*clustering*), *colocación óptima de recursos*, *división celular* y el *comportamiento territorial de los animales* [58]. En esta sección se discutirá la aplicación de las TVC y TDVCs para la generación de mallas.

4.3.1 Generación de mallas sin restricciones

Existen propiedades geométricas asociadas con la triangulación de Delaunay, a continuación se describen.

Propiedad 4.1 (**Criterio de Delaunay**). El interior de la esfera circunscrita de un simplex en la triangulación no contiene puntos generadores.

Propiedad 4.2 (**Propiedad Dual de Delaunay**). La arista es perpendicular a alguna cara de la teselación de Voronoi.

Propiedad 4.3 (**Propiedad del círculo vacío**). Para cada arista, se puede encontrar una esfera que contiene los puntos extremos de la arista, pero no contiene cualquier otro punto generador.

La aplicación de la TDVC para la generación de mallas sin restricciones trata principalmente con la distribución de generadores de acuerdo a alguna función de densidad ρ, la cual refleja las propiedades de la solución.

4.3.2 Generación de mallas con restricciones

En las aplicaciones prácticas, las mallas están sujetas a restricciones sobre la posición de los puntos generadores, también de las aristas y caras de los *simplex*. Es importante tomar en cuenta la restricción de marcar las mallas de acuerdo a la frontera geométrica, algunos enfoques para manejar estas restricciones son las siguientes.

- *De la frontera al interior*: Se puede determinar un subconjunto de puntos generadores en la frontera, por ejemplo, a través de una construcción de TVC dimensionalmente inferior, basada en una función predefinida. Su ventaja radica en el uso consistente del concepto de TVC que resulta en un algoritmo relativamente simple. La desventaja es que puede no ser fácil de determinar *a priori* el número de puntos generadores en la frontera y la función de densidad apropiada.

- *Del interior a la frontera*: Se puede construir la TVC y la TDVC sin aplicar restricciones usando un algoritmo estándar, como la iteración de Lloyd. Cada vez que se encuentra un nuevo conjunto de generadores, se determina si alguna de las correspondientes regiones de Voronoi se extienden hacia el

exterior del dominio. Algunos procedimientos pueden ser aplicados para proyectar varios o todos los generadores correspondientes hacia la frontera del dominio y luego continuar el proceso de iteración.

- *Formulación variacional*: En una configuración más general, considerar las TDVCs para un dominio acotado donde se tienen N puntos generadores z_i en el interior del dominio Ω, y M puntos y_j en la frontera $\partial\Omega$. Considerar minimizar la función modificada

$$\mathcal{F}(\{z_i, V_i\}_{i=1}^N, \{y_j, W_j\}_{j=1}^M) = \sum_{i=1}^{N} \int_{\mathcal{V}} \rho_1(x)||x-z_i||^2 d\mathcal{V} + \sum_{j=1}^{M} \int_{\mathcal{W}} \rho_2(x)||x-y_j||^2 d\mathcal{W}$$

con respecto a $\{z_i\}_{i=1}^N \subset \Omega$, $\{y_j\}_{j=1}^M \subset \partial\Omega$, $\mathcal{V} = \{V_i\}_{i=1}^N$ y $\mathcal{W} = \{W_j\}_{j=1}^M$, donde los conjuntos $\mathcal{V}$ y $\mathcal{W}$ forman una partición de Ω conjuntamente. Las funciones de densidad pueden ser definidas de manera diferente para reflejar los diferentes *pesos* colocados en los puntos generadores del interior y la frontera. La triangulación de Delaunay correspondiente con la minimización puede ser usada para generar mallas con relación a la frontera. Las funciones de densidad ρ_1 y ρ_2 pueden estar basadas en información previa o en un *estimador*[4] de error local.

4.3.3 Adaptación y suavizado local (Smoothing)

La adaptación de las mallas generalmente implica *refinamiento*, *engrosamiento* y *suavizado* [67, 68, 69]. Un enfoque popular es el *Suavizado de Laplace* (*Laplacian Smoothing*) [67, 69]. Al mover sucesivamente cada punto al centroide de sus vecinos, a menudo se mejoran las *rejillas* o *cuadrículas* (del inglés *grids*) resultantes.

El concepto de TDVC proporciona una buena explicación teórica para el proceso de suavizado: moviendo sucesivamente generadores hacia los centroides de las regiones de Voronoi, el coste funcional se reduce. Con una buena elección de la función de densidad, el coste funcional puede estar relacionado a los estimadores de

[4]Un estimador es una medida cuantitativa usada para valorar o inferir un parámetro desconocido.

error para el subproblema. Los promedios de los generadores vecinos proporcionan aproximaciones a los centroides de las regiones de Voronoi. Por lo tanto, el proceso de suavizado imita el proceso de construcción iterativa de TVCs (como el algoritmo de Lloyd) y, en consecuencia, contribuye a la reducción del *error de discretización*[5].

[5]Se refiere al error que se comete al utilizar un algoritmo determinado.

Capítulo 5

Reconstrucción 3D

En este capítulo se expondrá un método para la reconstrucción tridimensional de objetos, basada en teselaciones de Voronoi, su implementación ha sido realizada en el Software Libre `MeshLab`, el cual es un sistema de código abierto, portátil y extensible para el procesamiento y la edición de mallas triangulares tridimensionales no estructuradas [70]. El sistema está basado en gran medida en la librería `VCG`, desarrollado en el Laboratorio de Computación Visual del Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" [71]. Antes de desarrollar el método, es necesario tener en consideración algunos conceptos básicos del procesamiento y edición de mallas.

5.1 Procesamiento y edición de mallas

El proceso de convertir un conjunto de puntos de muestra en $\mathbb{R}^n$ en un modelo de gráficos por ordenador generalmente implica varios pasos: la reconstrucción de un modelo lineal inicial a trozos, limpieza, simplificación y a veces el ajuste con *parches* (del inglés *patches*) de superficie curvadas.

La reconstrucción de superficies 3D a partir de puntos de muestreo es un problema bien estudiado por la teoría de *Gráficos por ordenador*.

Definición 5.1 (**Reconstrucción de superficies**). La *reconstrucción de superficies* es una serie de métodos y técnicas computacionales que permiten un ajuste de los datos escaneados, relleno de huecos en la superficie y un *remallado* (del inglés *remeshing*) de los modelos tridimensionales existentes.

El algoritmo usado en `MeshLab` (**Surface reconstruction** `VGC`) es en su mayoría una variante de la *aproximación volumétrica* de Curless y Levoy [72] con algunos sistemas de ponderación originales, una regla de expansión diferente y del Laboratorio de Computación Visual *ISTI* [73] y otro enfoque para el relleno de huecos a través del volumen de *dilatación/relajación*. Un filtro es aplicado a todas las capas visibles del modelo a modificar. En la práctica, las *nubes* de mallas (o puntos) que son visibles son usadas para construir el campo de distancia volumétrica.

Definición 5.2 (**Subdivisión de superficies**). La *subdivisión de superficies* es un método de representación de una superficie *lisa* (*suave*[1]), a través de la especificación de una malla poligonal lineal a trozos[2] más gruesa.

La superficie lisa se puede calcular como el límite de un proceso recursivo de subdividir cada cara poligonal en caras más pequeñas que aproximen mejor la superficie lisa.

`MeshLab` aplica el algoritmo de *Loop* para la subdivisión de superficies [74], que funciona para cada triángulo y que tiene reglas para vértices extraordinarios.

En el muestreo estocástico, cada punto tiene la misma probabilidad finita de ser elegido, el patrón de *Muestreo de Poisson* se genera al agregar puntos en posiciones aleatorias hasta que se alcanza la densidad deseada (o hasta que la superficie se llena).

Definición 5.3 (**Muestreo de Poisson-disk**). El *muestreo de Poisson-disk* es una generalización del muestreo con distribución de *Poisson*, donde cada punto muestra satisface la restricción de una distancia mínima. Este patrón se forma

[1] Una superficie es suave cuando no tiene picos ni pliegues, es decir, cuando el vector producto elemental es diferente de cero.

[2] Estructura topológica en la cual se puede pasar de un gráfico a otro a través de una función lineal a trozos.

por la generación de puntos uniformemente distribuidos como en el muestreo de *Poisson* y se retienen aquellos que satisfacen la restricción de distancia mínima [75].

`MeshLab` crea una capa rellena con puntos de muestreo de la malla actual; las muestras son generadas de acuerdo a la distribución de Poisson-disk, usando el algoritmo descrito por Corsini et al [76].

Definición 5.4 (**Coloración de vértices**). Dada una malla M y un conjunto de puntos P, una *coloración de vértices* ocurre cuando se proyecta cada vértice de P sobre M y pinta a M de acuerdo a determinada propiedad de los puntos proyectados.

`MeshLab` proyecta los puntos sobre la malla según la distancia *geodésica* (ver Definición 2.17) entre ellos, mediante la opción **Coloración de vértices de Voronoi**. La coloración y la proyección se hacen en función de cada vértice.

Definición 5.5 (**Remuestreo de malla**). Un *remuestreo* (del inglés *resampling*) se realiza mediante la construcción de una *representación volumétrica*[3] *uniforme*, donde cada *vóxel*[4] posee una *distancia orientada*[5] de la superficie original, creando una nueva malla.

`MeshLab` crea una nueva malla, la cual es una versión de remuestreo de la última, la superficie *remuestreada* se reconstruye utilizando el algoritmo de *Marching Cubes* [77].

Definición 5.6 (**Suavizado de Taubin**). El algoritmo de *Taubin* es utilizado para el suavizado de mallas, éste abarca dos características: progresiva y gradual, para cada iteración [78].

Ahora nos encontramos en condiciones de exponer un método para *Voronizar* cualquier modelo tridimensional haciendo uso del Software Libre `MeshLab`.

[3] Visualización gráfica que forma una representación óptica de un objeto en tres dimensiones físicas.

[4] Unidad cúbica que compone un objeto tridimensional, el equivalente de un píxel en 3D.

[5] La función distancia orientada, o función distancia con signo, mide cuán cerca se encuentra un punto de un conjunto, otorgándole un signo dependiendo del lado en que se encuentre del conjunto.

5.2 Método de reconstrucción 3D con `MeshLab`

Después de ejecutar `MeshLab`, asegurarse de mostrar el **Diálogo de capas**, así como de activar la **Vista de líneas planas** (*Flat Lines*), de ser el caso, para observar mejor la triangulación de la superficie. A continuación se debe importar un modelo tridimensional cuya extensión sea soportable por el Software, una lista de las extensiones se muestra en la Tabla 5.1. El modelo aparecerá en el diálogo de capas como "0 xxx.yyy". Por ejemplo, si nuestro modelo se llama "Pieza_1" y tiene extensión ".stl", será nombrado "0 Pieza_1.stl" (0 se refiere al número de capa).

Paso 1: Crear una nueva superficie

Aplicar el filtro **Surface Reconstruction: VCG**, ubicado en la pestaña *Filters* → *Remeshing, Simplification and Reconstruction*. Se abrirá un cuadro de diálogo en la cual podremos introducir los parámetros de nuestra preferencia, se recomienda cambiar únicamente el parámetro de la característica **Voxel Side** con un porcentaje de 0.5; la casilla se presenta como *perc on*, abarcando los valores $(0, 119.501)$. Una vez aplicada tal característica, cerrar el cuadro de diálogo (en adelante, **Aplicar y cerrar**). Esto creará una nueva capa que se mostrará en el diálogo de capas, generalmente con el nombre de "1 plymcout.ply".

Para mayor comodidad, se aconseja eliminar la capa 0. Para ello, primero se debe seleccionar, dar clic secundario y elegir la opción *Delete Current Mesh*.

Paso 2: Crear una subdivisión de la superficie

Aplicar el filtro **Subdivision Surfaces: Loop**, ubicado en *Filters* → *Remeshing, Simplification and Reconstruction*. En el cuadro de diálogo abierto, se sugiere cambiar sólo el parámetro de la característica **Edge Threshold**, que será el umbral de las aristas, con un porcentaje de 0.5. **Aplicar y cerrar**.

Formato	*Extensión*
Stanford Polygon File Format	(*.ply)
STL File Format	(*.stl)
Alias Wavefront Object	(*.obj)
Quad Object	(*.qobj)
Object File Format	(*.off)
PTX File Format	(*.ptx)
VCG Dump File Format	(*.vmi)
Breuckmann File Format	(*.bre)
Collada File Format	(*.dae)
OpenCTM Compressed Format	(*.ctm)
Expe's Point Set (Binary)	(*.pts)
Expe's Point Set (ASCII)	(*.apts)
XYZ Point Cloud (With or Without Normal)	(*.xyz)
GNU Triangulated Surface	(*.gts)
Protein Data Bank	(*.pdb)
TRI (Photogrammetric Reconstructions)	(*.tri)
ASC (ASCII Triplets of Points)	(*.asc)
TXT (Generic ASCII Point List)	(*.txt)
X3D File Format - XML Encoding	(*.x3d)
X3D File Format - VRML Encoding	(*.x3dv)
VRML 2.0 File Format	(*.wrl)

Tabla 5.1: Extensiones soportadas por `MeshLab`.

Una vez aplicado el filtro, el modelo tendrá un mayor número de triangulaciones en su superficie.

Paso 3: Crear los generadores (centroides)

Aplicar el filtro **Poisson-disk Sampling**, ubicado en *Filters* → *Sampling*. En el cuadro de diálogo elegir el número de muestras (puntos) deseado, mediante la característica **Number of Samples**, así como un radio específico, característica **Explicit Radius**. Esto creará una nueva capa en el diálogo de capas.

Dependiendo del modelo, se sugiere probar distintos parámetros hasta encontrar una configuración oportuna. Haciendo esto se producirán varias capas, y para comprobar cuál será elegida, se debe desactivar la vista de la capa 1, haciendo clic sobre el ícono  de la misma. **Aplicar y cerrar**.

Paso 4: Colorear los vértices

Aplicar el filtro **Voronoi Vertex Coloring**, ubicado en *Filters* → *Sampling*. Seleccionar la característica **Back Distance** y la vista previa (**Preview**), con lo cual el modelo será coloreado en base a sus vértices y podremos observar una primera vista de las aristas de Voronoi en el modelo. **Aplicar y cerrar**.

Es importante recalcar que antes de aplicar este filtro, debe estar seleccionada la capa 1.

Paso 5: Elegir las caras por sus vértices

Aplicar el filtro **Select Faces by Vertex Quality**, ubicado en *Filters* → *Selection*. Seleccionar la vista previa (**Preview**) y no cerrar cuadro de diálogo.

Este filtro permite elegir las caras que serán finalmente eliminadas, para obtener sólo las aristas de Voronoi. Para ello primero se deben aplicar ciertas configuraciones para que `MeshLab` determine de manera correcta cuáles componentes deben suprimirse, a continuación descritas:

1. Ocultar el color de selección: *Render* → *Color* → **None**.
2. En el cuadro de diálogo de **Select Faces by Vertex Quality**, cambiar el parámetro de la característica **Min Quality** a 0, mientras que el parámetro de **Max Quality** dependerá del grosor deseado en las aristas. **Aplicar y cerrar**.
3. Debido a que `MeshLab` reconoce los colores de manera particular (rojo es el color de la selección), debemos *invertir* la selección. Para borrar las caras, primero debemos elegirlas y después aplicar el filtro **Invert Selection**, ubicado en *Filters* → *Selection*. **Aplicar y Cerrar**.

Paso 6: Eliminar las caras y los vértices

Aplicar el filtro **Delete Selected Faces and Vertices**, ubicado en *Filters* → *Selection*. Este filtro borra las caras y los vértices del modelo; en la vista final aparecerán las aristas y los centroides calculados en el Paso 3, lo cual no es motivo de inquietud, puesto que no es determinante en el resultado final, sólo debe desactivarse la vista de la capa 2 (o la capa elegida en el Paso 3).

Paso 7: Suavizar las aristas

Aplicar el filtro **Laplacian Smooth**, ubicado en *Filters* → *Smoothing, Fairing and Deformation*, las veces que sean necesarias hasta obtener un modelo más estético.

Paso 8: Crear un remuestreo de malla

Aplicar el filtro **Uniform Mesh Resampling**, ubicado en *Filters* → *Remeshing, Simplification and Reconstruction*. En el cuadro de diálogo se sugiere modificar los parámetros de las características **Precision** (con un porcentaje de 0.5) y **Offset** (con un porcentaje de 52), así como dejar activadas las casillas **Multisample** y **Absolute Distance**.

Esto creará la nueva capa *Offset mesh*, que es un nuevo muestreo de superficie del modelo, es decir, se crearán nuevas mallas a partir de la existente, otorgando volumen a las aristas. El uso de memoria con el algoritmo de *Marching Cubes* es elevado, por lo que esta etapa puede durar incluso minutos, dependiendo de la complejidad del modelo.

En este punto dejar sólo la vista *Offset mesh* activada.

Paso 9: Suavizado del modelo final

Aplicar el filtro **Taubin Smooth**, ubicado en *Filters* → *Smoothing, Fairing and Deformation*. Recordar que debe estar seleccionada la capa *Offset mesh*, ya que es el modelo final al que se va a aplicar este filtro. Se recomienda no modificar los parámetros por defecto en el cuadro de diálogo y usar el filtro las veces necesarias hasta obtener un buen resultado.

Para observar mejor los efectos de cada filtro se aconseja aumentar o disminuir el *zoom*, dependiendo de cuál se esté utilizando.

Finalmente, si se requiere, guardar el modelo voronizado exportándolo como malla en el formato que se desee: *File* → *Export Mesh As...* Antes se debe seleccionar la capa final que hemos obtenida, llamada "Offset mesh", a continuación elegir el formato en la sección **Files of type** y nombrarlo como se prefiera. Alternativamente se puede exportar directamente mediante la combinación de teclas `Ctrl+E`.

Capítulo 6

Resultados

6.1 Aplicaciones analizadas

Antes de exponer los resultados del método, serán consideradas algunas aplicaciones del diagrama de Voronoi analizadas durante la investigación.

6.1.1 El brote de cólera en Londres

En los años 1853 y 1854, Londres enfrentó una tercera epidemia de cólera. Por aquél entonces los habitantes de ciertos distritos del sur de la ciudad extraían agua del Támesis o la obtenían de bombas de uso público. Pero los desechos humanos eran vertidos en alcantarillas improvisadas o directamente al río. Snow sostenía que la gente, al beber agua contaminada extraída del río, ingería materia insana y de esta manera contraía el cólera [79].

A principios de septiembre de 1854, el sector *Golden Square* fue escenario de un brote epidémico de gran intensidad (500 muertes en 10 días). Snow era vecino del área y sabía que la mayoría extraía agua de una bomba en la calle *Broad Street*, por ello registró las direcciones de 83 personas fallecidas en el área. Pronto confirmó que la mayoría de los moradores se abastecían de la bomba mencionada, dado que calculó la distancia entre la residencia de cada víctima y la bomba más cercana.

FIGURA 6.1: Análisis del brote de cólera en Londres, por John Snow, la cruz roja representa la bomba de *Broad Street*.

En la Figura 6.1, los puntos identifican los domicilios de las personas fallecidas, mientras que las cruces representan las bombas, se ha resaltado la bomba de *Broad Street* con una cruz roja.

Probamos si en verdad la bomba de *Broad Street* es la más cercana al domicilio de las muertes por cólera. Para ello dibujamos el diagrama de Voronoi tomando como puntos generadores cada bomba en el mapa, el resultado se expone en la Figura 6.2. Éste y los consecuentes diagramas fueron construidos con el Software Libre `GeoGebra` [80].

Se mostró gráficamente la relación espacial entre las muertes por cólera y la bomba de *Broad Street*. Tras la inhabilitación de la bomba, se observó una reducción en la incidencia y mortandad por cólera. Aunque posteriormente, debido a la incredulidad de las autoridades y la presión popular, se habilitó nuevamente su uso.

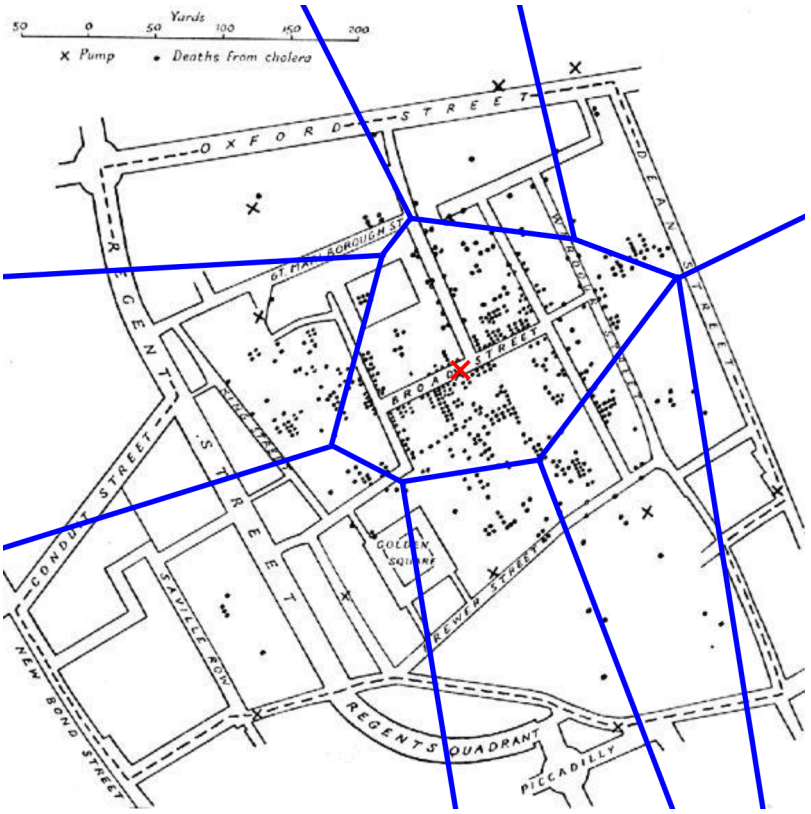


FIGURA 6.2: Diagrama de Voronoi del análisis de brote de cólera en Londres.

6.1.2 Ataque a Pearl Harbor

El 26 de noviembre de 1941 una fuerza de ataque japonesa compuesta por seis portaaviones zarpó desde el norte de Japón en ruta hacia el noroeste del archipiélago de Hawái, desde donde lanzaría sus unidades aéreas para atacar Pearl Harbor, acción sucedida en la mañana del 7 de diciembre. El ataque conmocionó al pueblo estadounidense y llevó directamente la entrada de los Estados Unidos en la *Segunda Guerra Mundial*. La Figura 6.3 muestra el trayecto seguido por los japoneses [81], se ha marcado con una cruz roja la base naval de Pearl Harbor y las bases navales cercanas (islas Aleutianas, Midway y Wake) a ella mediante una cruz azul. Hay que aclarar que la ruta es aproximada.

Nuestro propósito fue mostrar si en realidad el plan fue rigurosamente estructurado, con base en la distancia a las bases navales cercanas a Pearl Harbor, que pudieran detectar el ataque antes de cometerse. Trazamos el diagrama de Voronoi y el resultado se observa en la Figura 6.4.

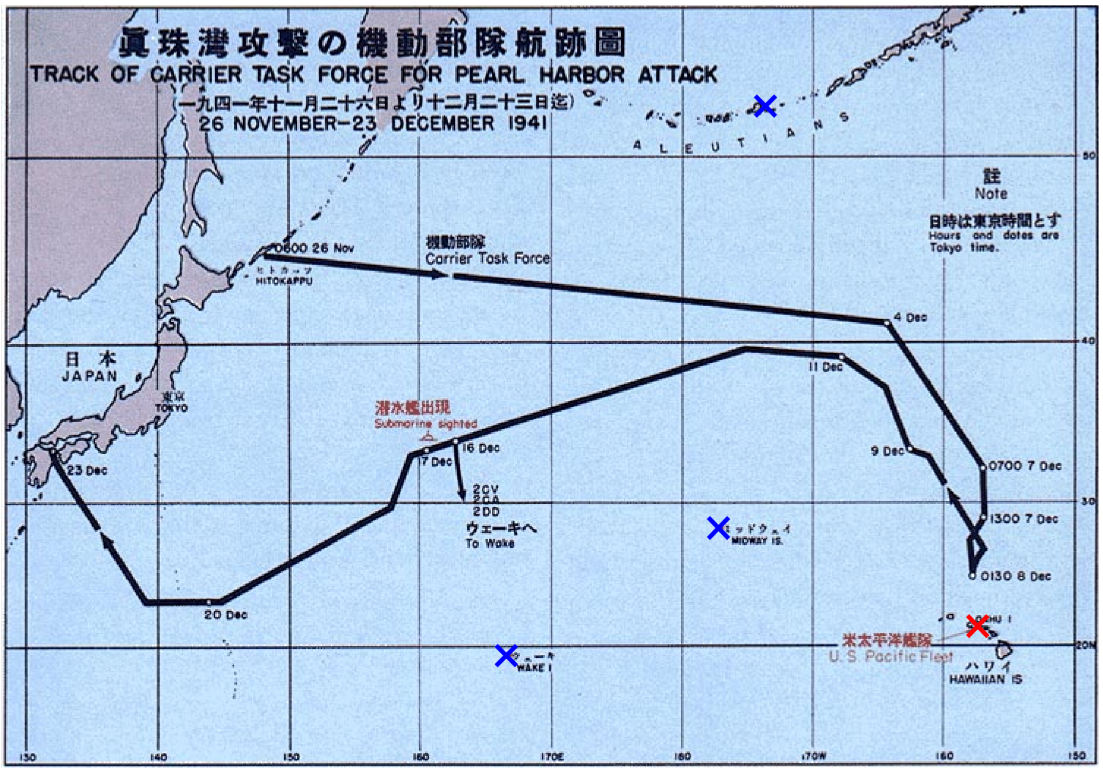


FIGURA 6.3: Ruta del ataque a Pearl Harbor recorrida por la Fuerza Especial de Portaaviones Japonesa de ida y vuelta.

Podría suponerse que la flota japonesa escogió uno de los puntos más alejados de las bases en las islas Aleutianas y Wake para salir. La ruta siguió la línea mediatriz entre las islas Aleutianas y Midway antes de desviarse en el punto más alejado entre las islas Aleutianas, Midway y Pearl Harbor.

6.1.3 Patrones en la piel de una jirafa

Un reciente estudio efectuado por investigadores del *King's College* de Londres acaba de confirmar la hipótesis planteada por el matemático británico Alan Turing hace medio siglo [82]. Según su teoría, los patrones biológicos como las rayas del tigre y las manchas del leopardo aparecen por la interacción de un par de *morfógenos* (sustancias que gobiernan el desarrollo tisular), uno activador y otro inhibidor. Estos dos morfógenos se combinarían para crear el patrón de rayas o manchas así: el activador forma la raya, pero, al interaccionar con el inhibidor, deja

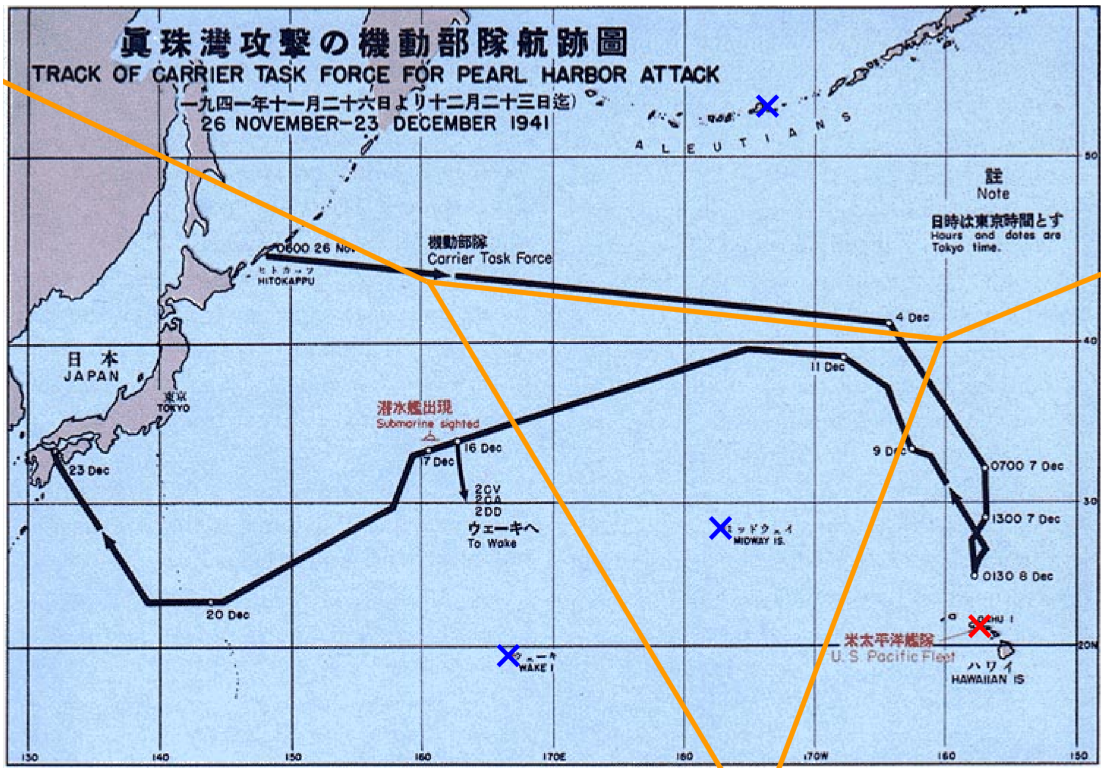


FIGURA 6.4: Diagrama de Voronoi aplicado a la ruta de ataque a Pearl Harbor.

de manifestarse y da lugar a un *espacio en blanco* antes de volver a manifestarse en forma de otra raya.

Las jirafas tienen un sistema de vasos sanguíneos que pasan por debajo de las manchas rojizas de sus pelajes (ver Figura 6.5). Un vaso sanguíneo más grande rodea cada mancha y envía vasos más pequeños al centro de la mancha.

FIGURA 6.5: Patrones en la piel de una jirafa.

En el centro de cada mancha, los vasos liberan calor desde el cuerpo, regulando la temperatura corporal del animal. Sus manchas sirven para deshacerse del calor ya que ellas no pueden sudar, el centro de las manchas es por donde emana la temperatura más caliente del cuerpo.

Para asegurarnos de que la piel de una jirafa sigue un diseño de diagrama de Voronoi, se trataron de ubicar los puntos generadores a partir del color más obscuro dentro de las regiones, es decir, se intentaron hallar los vasos que liberan el calor, el resultado puede observarse en la Figura 6.6.

Como se esperaba, los vasos que liberan calor poseen un color más obscuro en relación al resto de la piel, también se constata que los patrones de ésta siguen un diseño de diagrama de Voronoi.

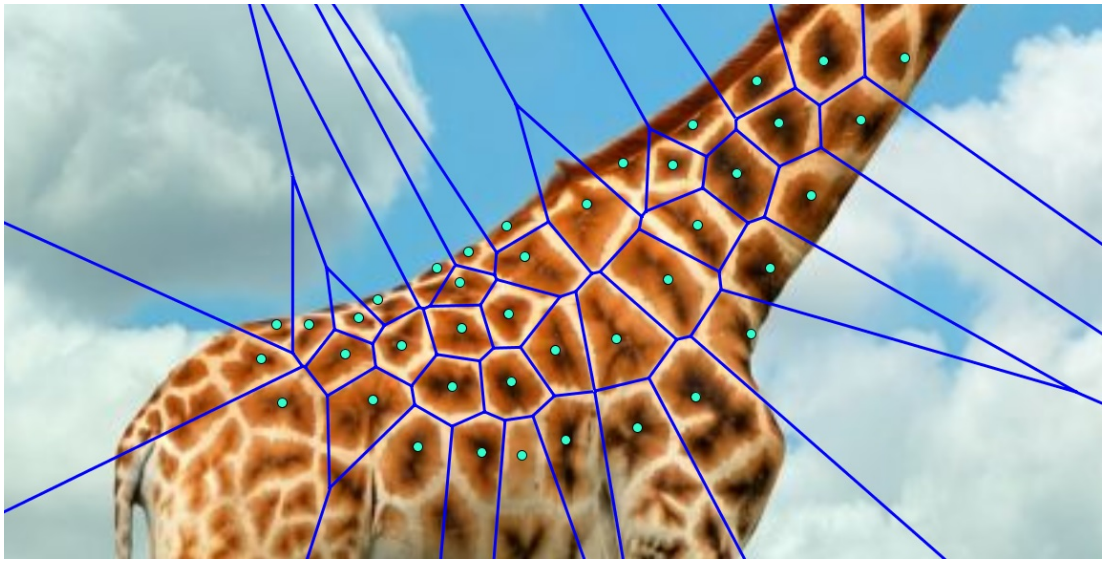


FIGURA 6.6: Aproximación del diagrama de Voronoi en la piel de una jirafa.

6.1.4 Oxxos cerca de CU

Este análisis surgió como una necesidad de proximidad en nuestro entorno, específicamente la ubicación de los Oxxos cercanos a Ciudad Universitaria (BUAP). Supongamos que nos encontramos en la Facultad de Ciencias Físico Matemáticas (marcada con una cruz naranja en la Figura 6.7) y se desea comprar una bebida energética, café o realizar una recarga telefónica; sería de gran utilidad conocer

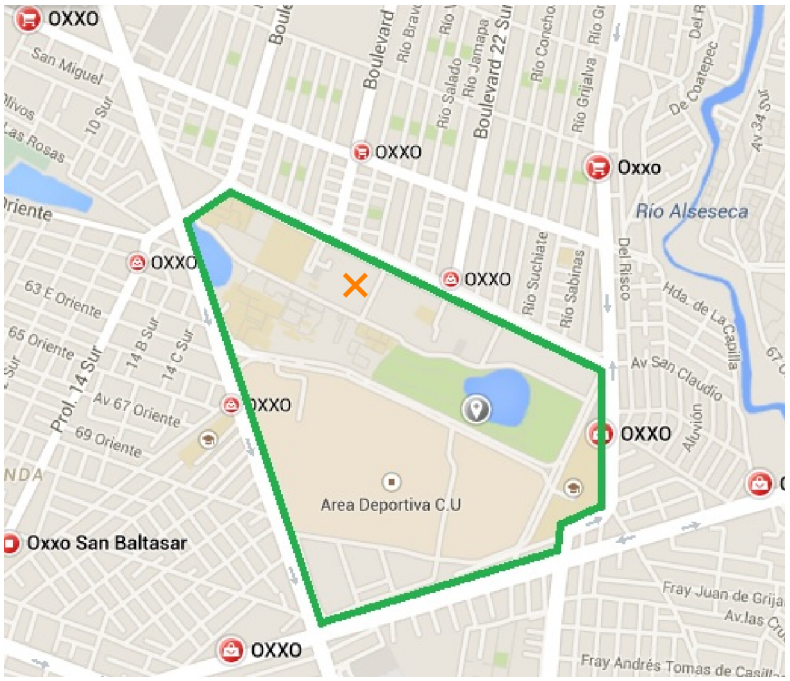


FIGURA 6.7: Vista del área cercana a CU.

qué tienda es más cercana a la Facultad para evitar un gasto innecesario de tiempo y espacio. Para ello se trazó el diagrama de Voronoi tomando como generadores cada Oxxo en los alrededores a CU (ver Figura 6.8). Como puede observarse, el Oxxo más cercano a la facultad es el de la Calle 22 Sur, esquina con Boulevard Valsequillo, por lo que en este caso debemos dirigirnos a él.

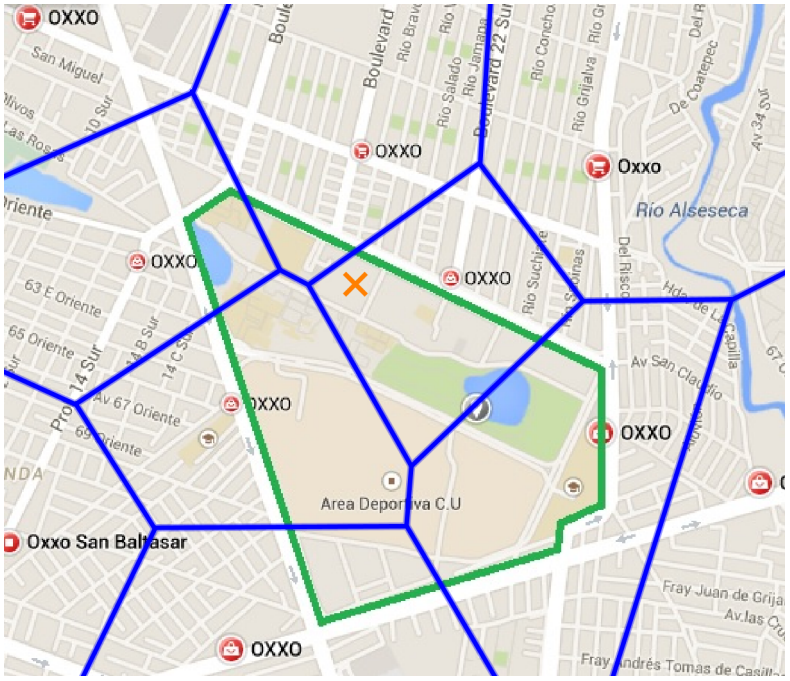


FIGURA 6.8: Diagrama de Voronoi aplicado al área cercana a CU.

6.2 Resultados de la aplicación del método

A continuación se presentan imágenes de los pasos principales a seguir del método de reconstrucción 3D. Se ha elegido una *pieza ornamental* que fue *escaneada tridimensionalmente* en el Laboratorio de Sistemas Dinámicos Controlables.

Pieza original

En el diseño original de la Figura 6.9, puede apreciarse que la superficie del modelo aún no ha sido suavizada, es por ello que utilizamos la instrucción de **remallado** y la de **suavizado**. Se desactivó la vista de *líneas planas*, en su lugar se activó la vista de *superficie plana* (*Flat*).

Cabe resaltar que el modelo 3D, al haber sido escaneado, contiene una cantidad grande de información, lo cual aumenta un poco más el tiempo de respuesta a cada instrucción.

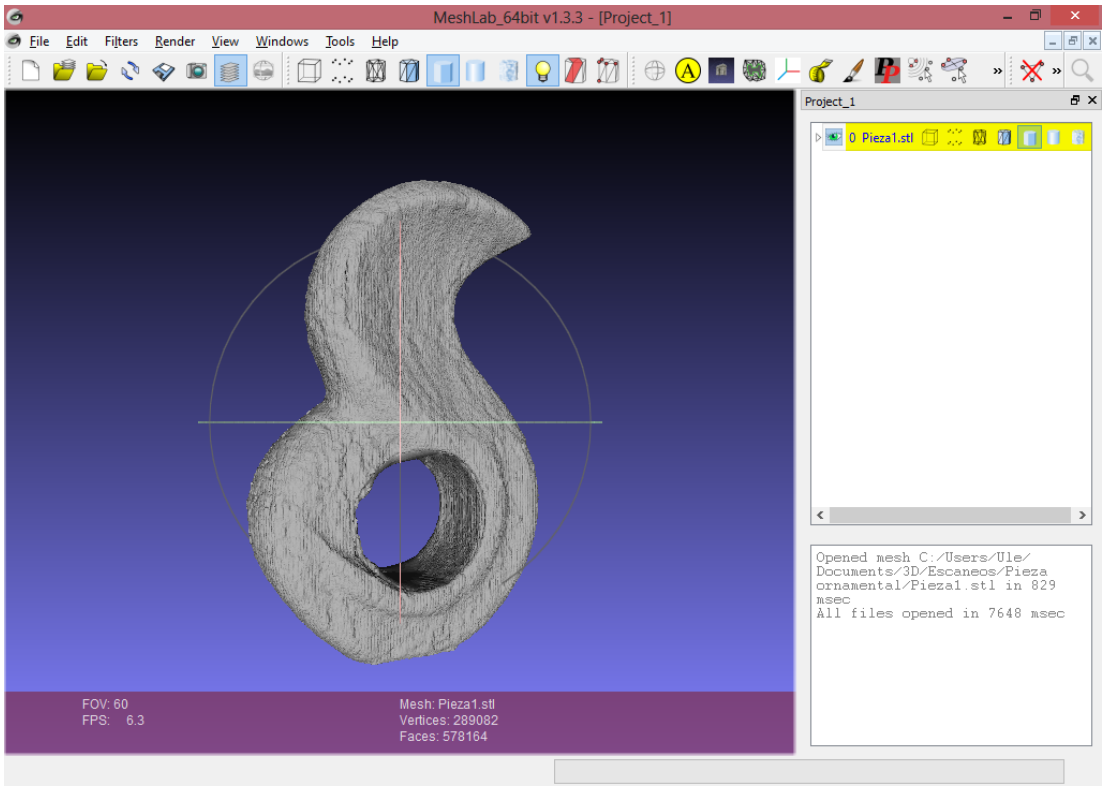


FIGURA 6.9: Diseño original de la pieza ornamental.

Creación de los puntos generadores (centroides)

La figura 6.10 expone los puntos creados; para este ejemplo se tomaron 222 muestras y un radio explícito del 5%. Para ello la vista de la capa 1 fue desactivada y la capa 2 es mostrada, que lleva por nombre "Poisson-disk Samples".

Esta parte es importante, ya que el número de puntos depende directamente del modelo original, así como saber aproximadamente cuántos de ellos serán necesarios para que nuestro modelo final sea el mejor.

Notar que los generadores de las teselaciones de Voronoi fueron creados aleatoriamente, pero de tal manera que cubren la totalidad de la superficie del modelo original.

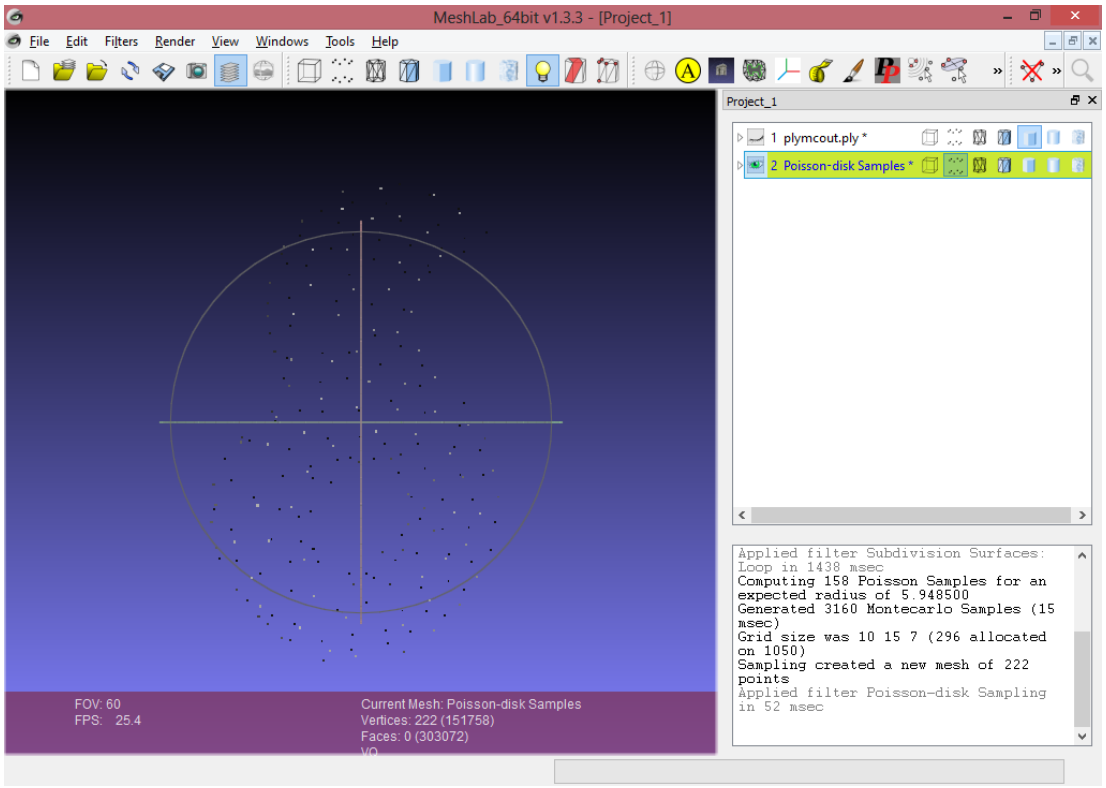


FIGURA 6.10: Vista de *Puntos* con la distribucin Poisson-disk.

Delimitación de aristas

En el cuadro de diálogo **Selected Faces by Vertex Quality** se cambió el parámetro **Min Quality** a un valor de 0 y el parámetro **Max Quality** a 1.2, de tal manera que las aristas quedaron formadas como se exhiben en la Figura 6.11.

Notar que las aristas de Voronoi están marcadas en rojo.

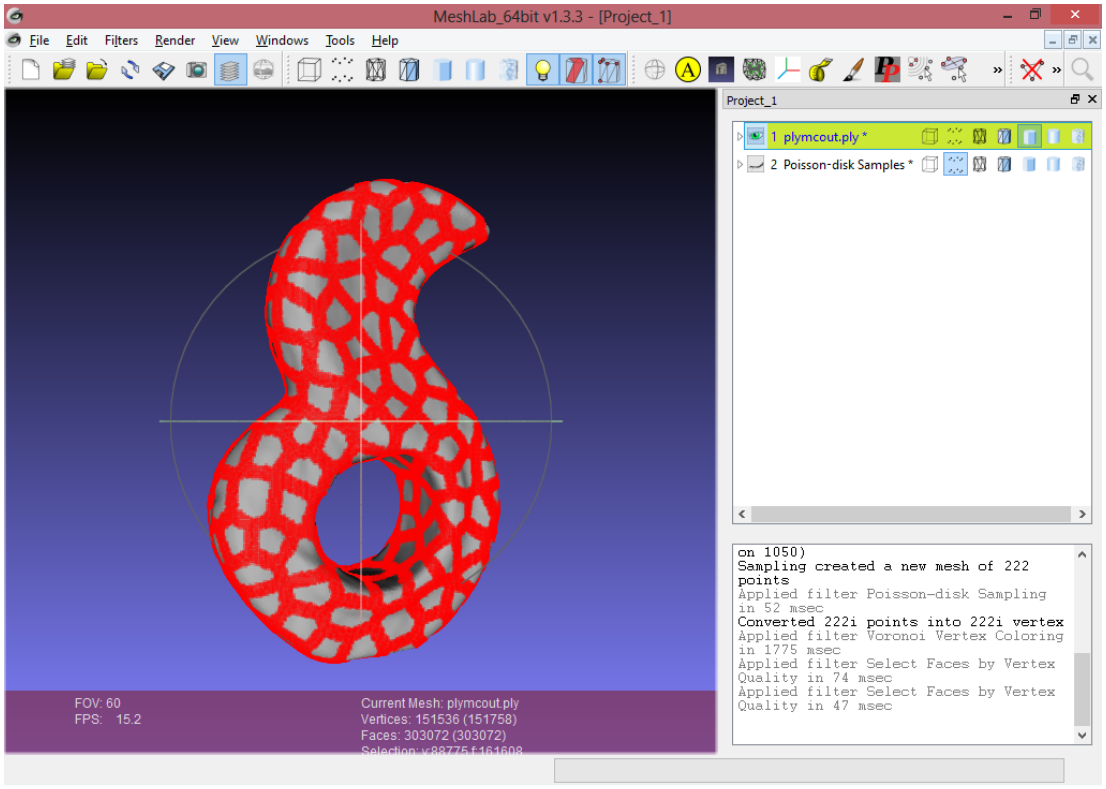


FIGURA 6.11: Vista con la calidad de vértice modificada.

Eliminación de caras y vértices

La Figura 6.12 muestra la vista de eliminación de vértices y caras. Notamos que las aristas aún exhiben vértices que no han sido suavizados.

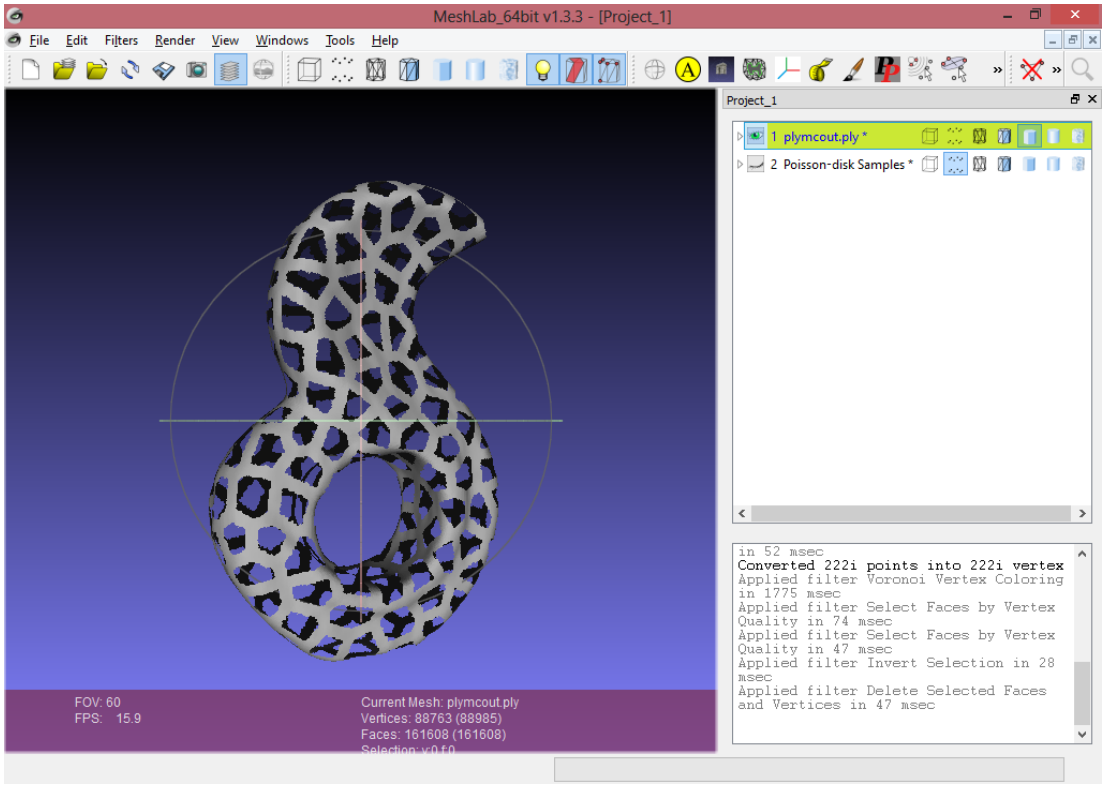


FIGURA 6.12: Vista de supresión de caras y vértices seleccionados.

Suavizado final

Finalmente se aplicó 4 veces el algoritmo de *Taubin* para suavizar mallas, otorgando una vista mejorada a nuestro modelo, el resultado se observa en la Figura 6.20.

Se guardó el archivo final con el nombre "Pieza ornamental.stl" para su posterior impresión.

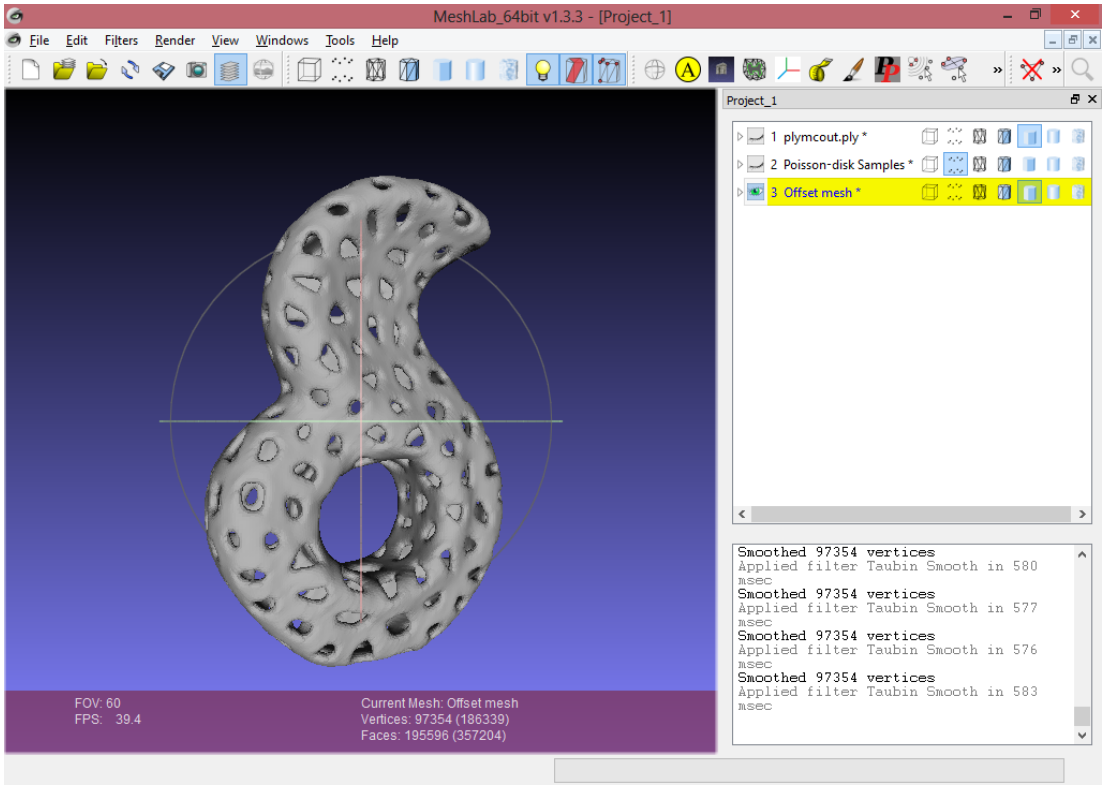


FIGURA 6.13: Vista de las aristas suavizadas mediante la aproximación de Taubin.

Capítulo 7

Conclusiones y Recomendaciones

Apéndice A

Figuras

Creación de una subdivisión de la superficie

Después de formarse la nueva superficie, se añadirá una triangulación al modelo, para nuestros fines se ha creado una superficie con más vértices, aristas y caras (**Voxel Side** = 0.5%).

Puede observarse en la Figura A.1 que el modelo ha sufrido una ligera modificación, la superficie se nota más suavizada, debido a que se añadieron nuevos elementos, lo cual lo hace un poco más estético.

Esto hace pensar a la instrucción **Surface Reconstruction: VCG** como un auxiliar importante para la reconstrucción de superficies en cualquier ámbito.

Se produjo una nueva capa: "1 plymcout.ply" y se eliminó la capa 0 (modelo original).

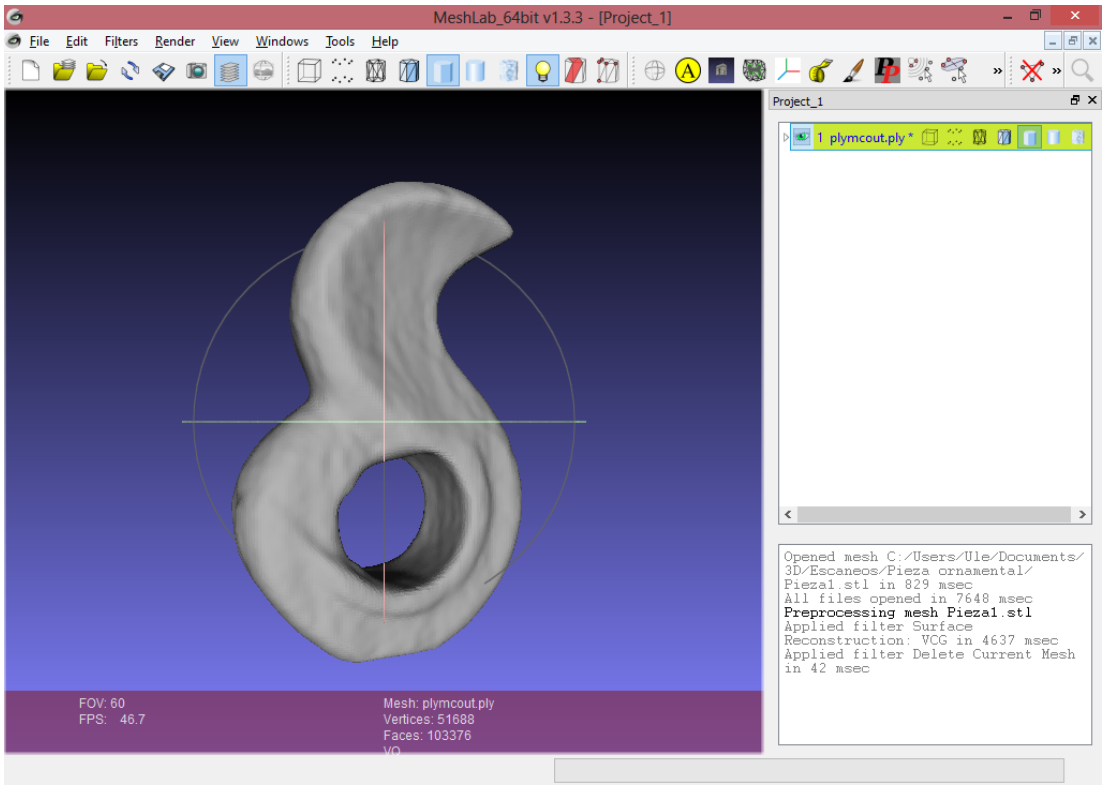


FIGURA A.1: Modelo con superficie reconstruida.

Coloreado de los vértices

La Figura A.2 muestra el coloreado de los vértices, se puede observar la aplicación del proceso de *voronización* en la superficie del modelo.

Así mismo, notar que cada región de Voronoi tiene como punto generador su correspondiente centroide creado anteriormente.

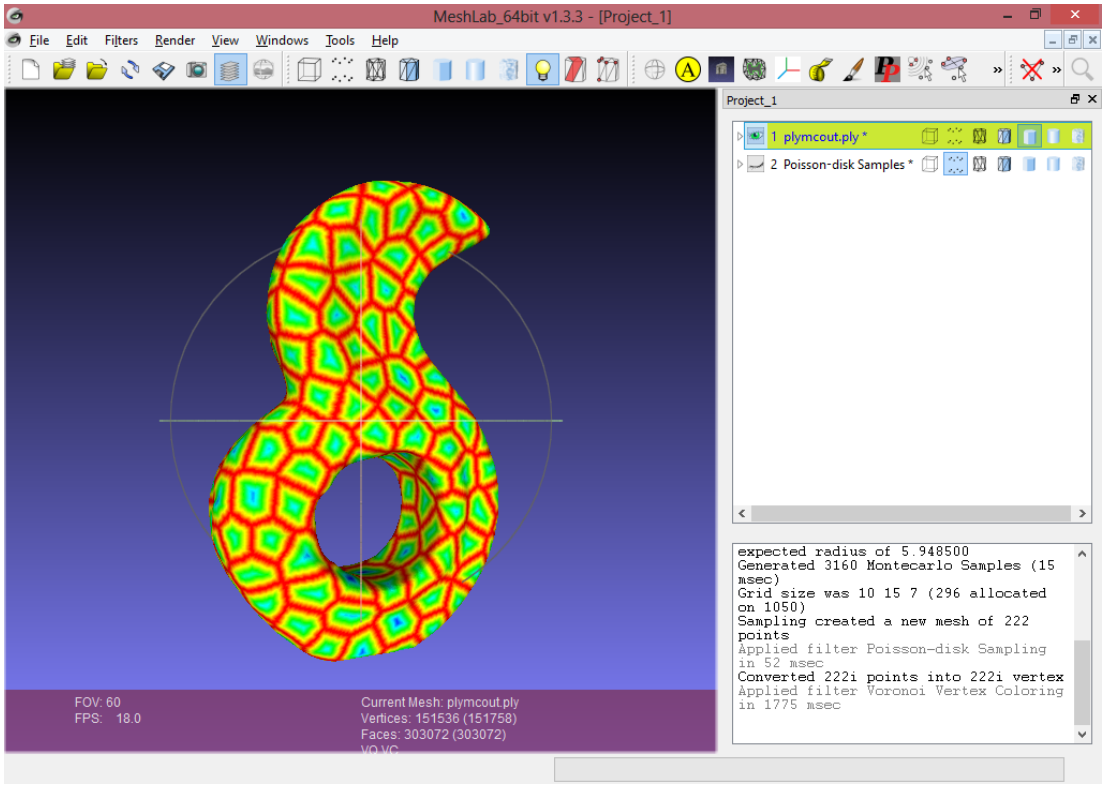


FIGURA A.2: Vista de filtro de coloración de vértices de Voronoi (**Back distance** activado).

Elección de las caras por sus vértices

En la Figura A.3 pueden observarse las caras que serán eliminadas, para ello antes se configuró la elección, ya que `MeshLab` debió reconocer los elementos a suprimir.

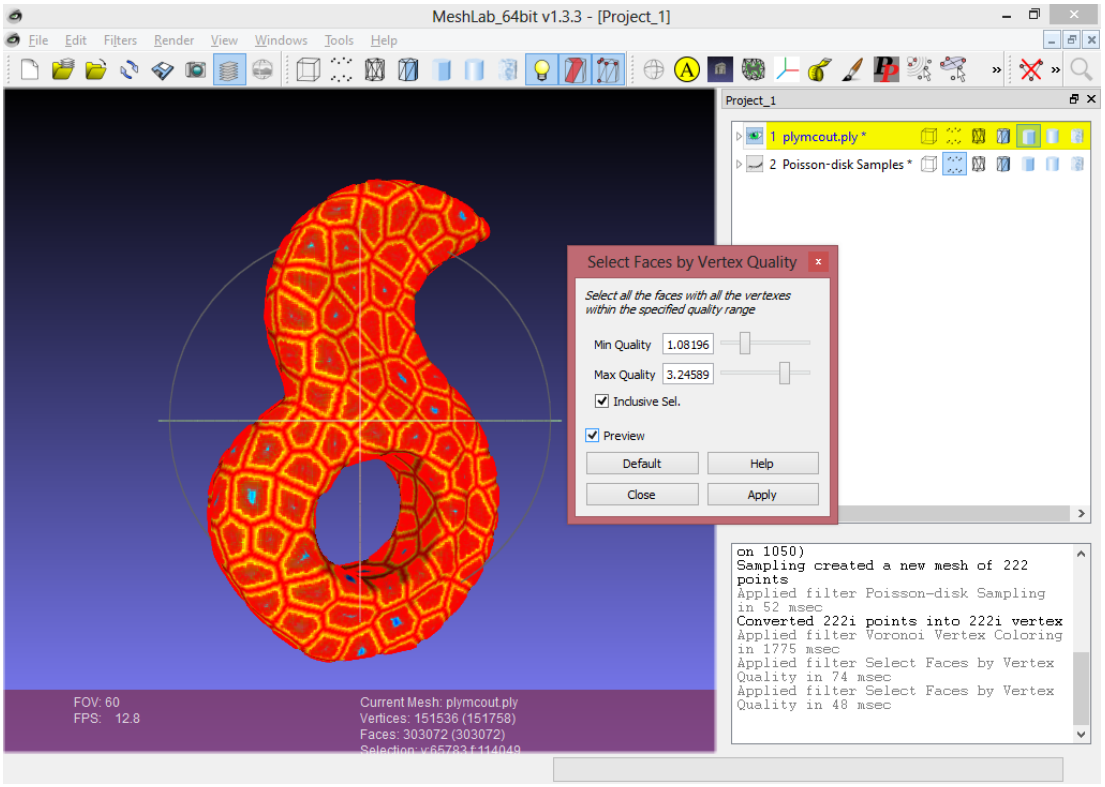


FIGURA A.3: Vista de selección de caras por su rango de calidad sin modificar.

Color oculto

Primero se ocultó el color, de tal manera que los colores mostrados en la Figura A.4 (rojo y gris) son los cuáles `MeshLab` detectará para efectuar la eliminación.

Notar que las caras aún no están bien definidas.

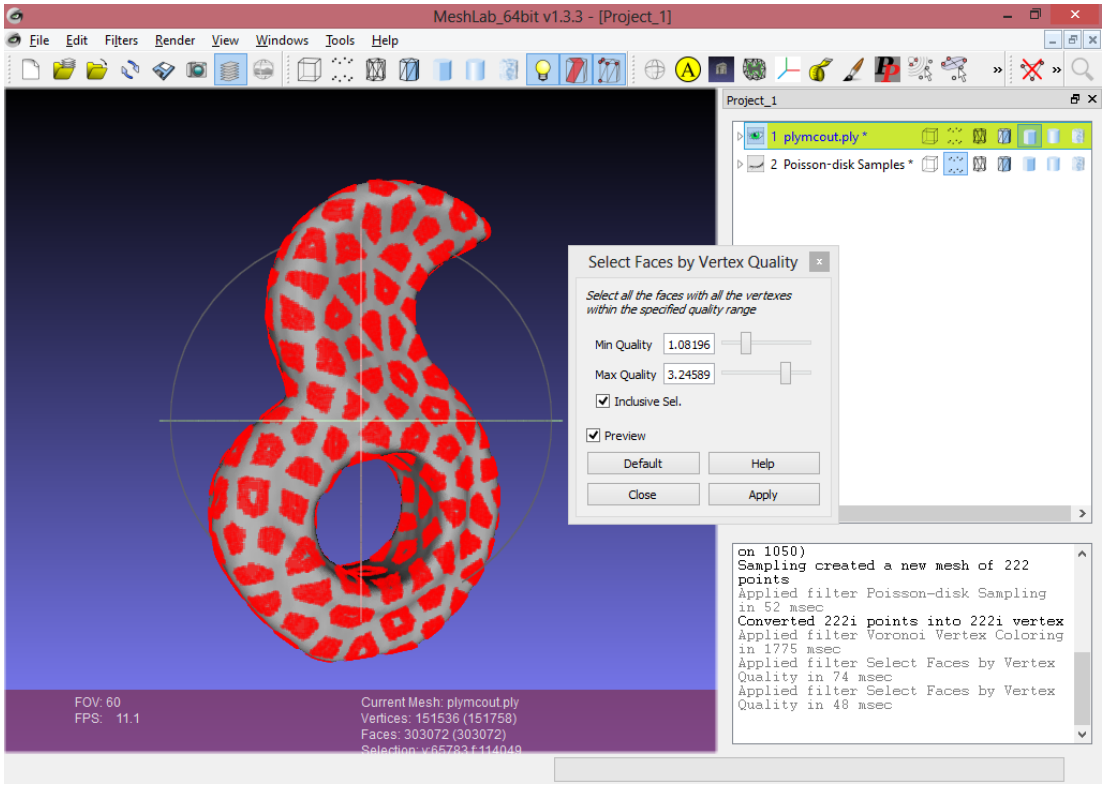


FIGURA A.4: Vista *sin color*.

Color invertido

`MeshLab` reconoce al rojo como tono de distinción, es por esta razón que se invirtió el color de selección. En la Figura A.5 se observan las caras que serán eliminadas.

De igual manera, notar que las caras están completamente definidas.

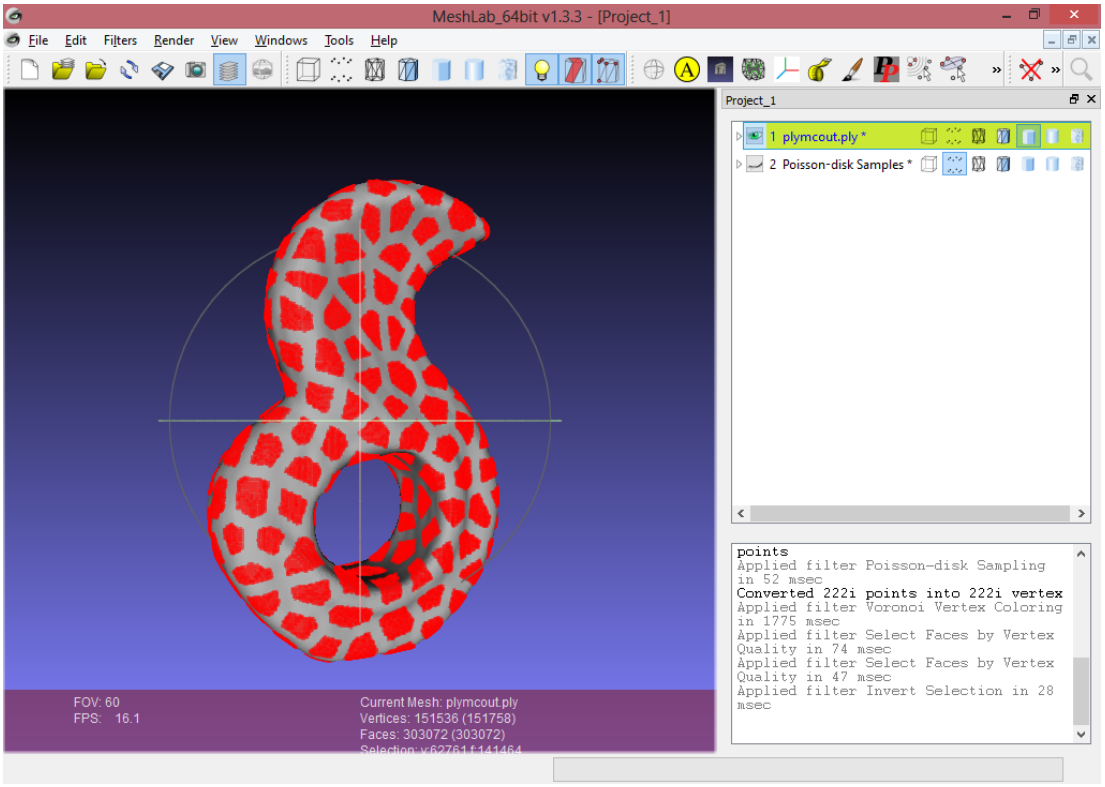


FIGURA A.5: Vista con la selección de caras y vértices invertida.

Suavizado de aristas

La Figura A.6 muestra el efecto del algoritmo de *Suavizado de Laplace* para el suavizado de mallas. Notamos que las aristas se observan más estéticas y delimitadas.

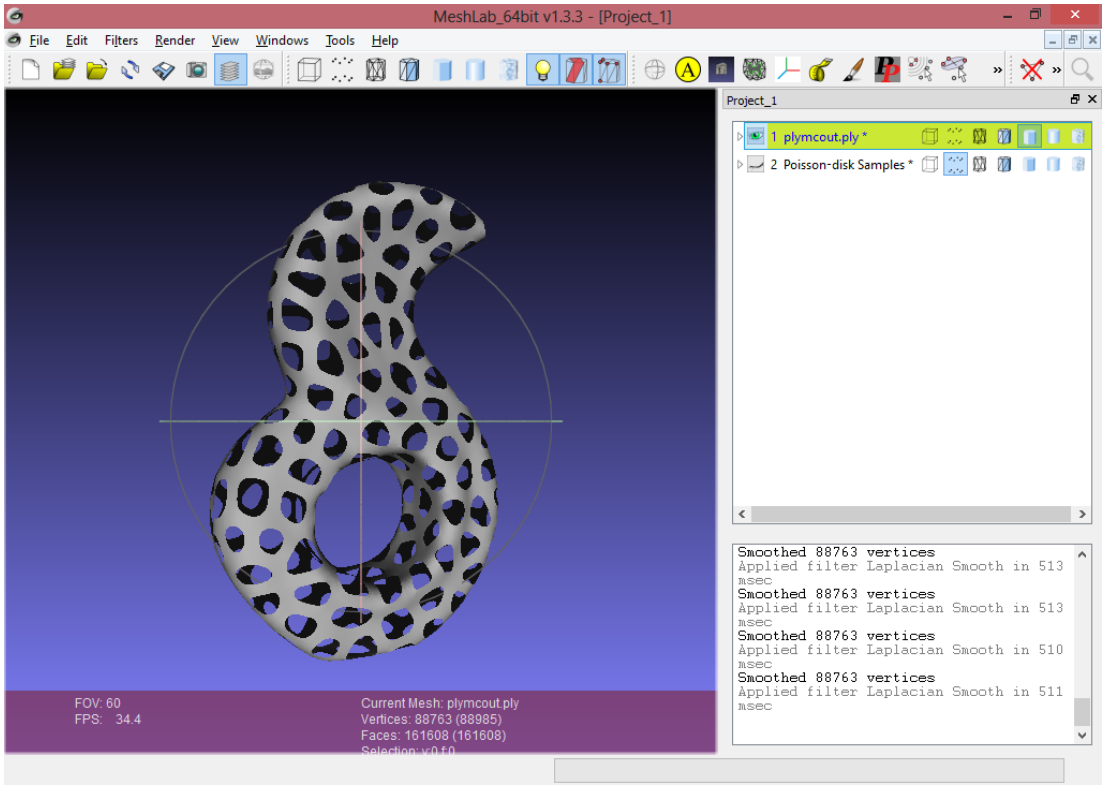


FIGURA A.6: Vista de aristas suavizadas.

Aristas con volumen

La aplicacin del algoritmo de *Marching Cubes* se prolongó aproximadamente 8 minutos, ya que nuestro modelo ejemplo provenía de un escaneo tridimensional con muchos vértices.

En la Figura A.7 se aprecia que las aristas han adquirido volumen, si bien no han sido suavizadas.

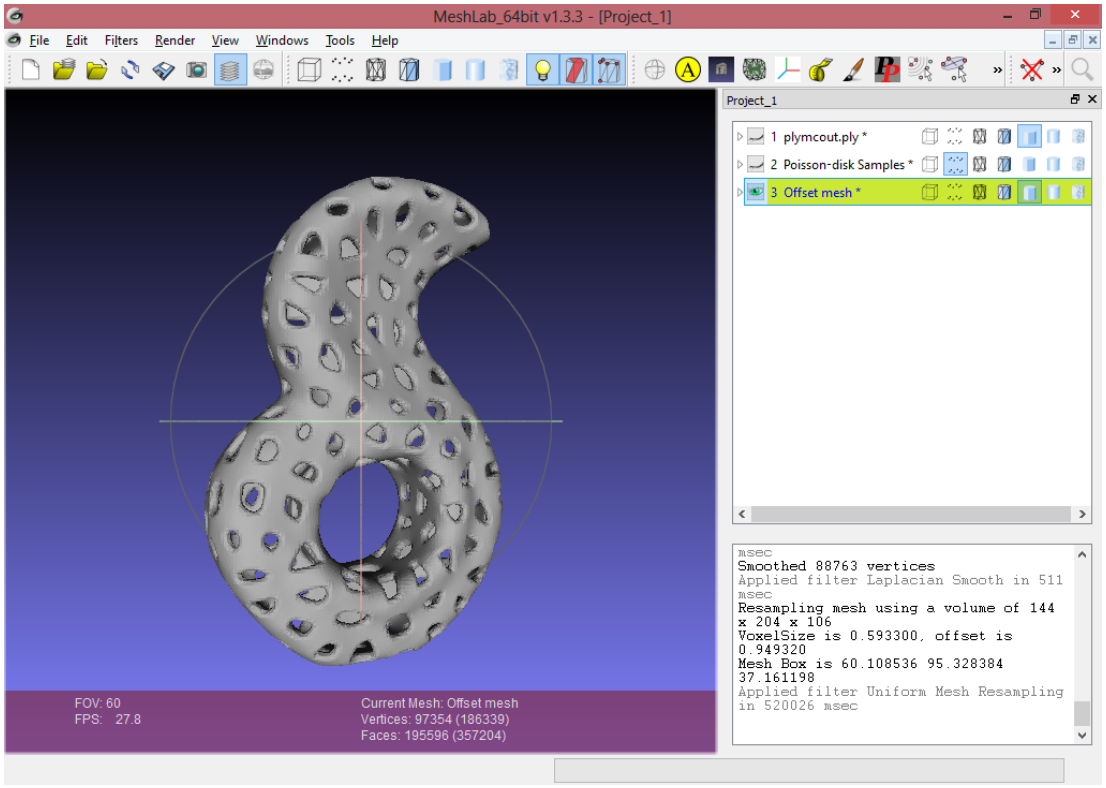


FIGURA A.7: Vista de remuestreo de mallas uniforme.

Apéndice B

Mathematica

Algoritmo aplicado en el Software `Mathematica`.

```
n = Input[``Puntos deseados, n= ''];
Manipulate[If[Length[ptos] < 3, AppendTo[ptos, RandomReal[{0, 10}, {2}]]];
vor = ListDensityPlot[

   Map[Flatten, Transpose[{ptos, Range[Length[ptos]]}]],

   PlotRange → {{0, 10}, {0, 10}},InterpolationOrder → 0,

   Mesh → If[mesh, All, None], ColorFunction → (ColorData[colors][#] &),

   FrameTicks → False],

``Ninguno'' → {{ptos, RandomReal[{0, 10}, {n, 2}]}, {0, 0}, {10, 10},
Locator, LocatorAutoCreate → True},
{{colors, ``BrightBands'', ``Esquema de Colores''}, ColorData[``Gradients'']},
{{mesh, False, ``Mostrar Fronteras''}, {True, False}}]
```

Si $n = 10$ se produce una imagen como la que se muestra en la Figura B.1. Ésta consiste en 10 puntos generados aleatoriamente y que cuenta con la posibilidad de moverlos y una opción para ver las aristas (fronteras).

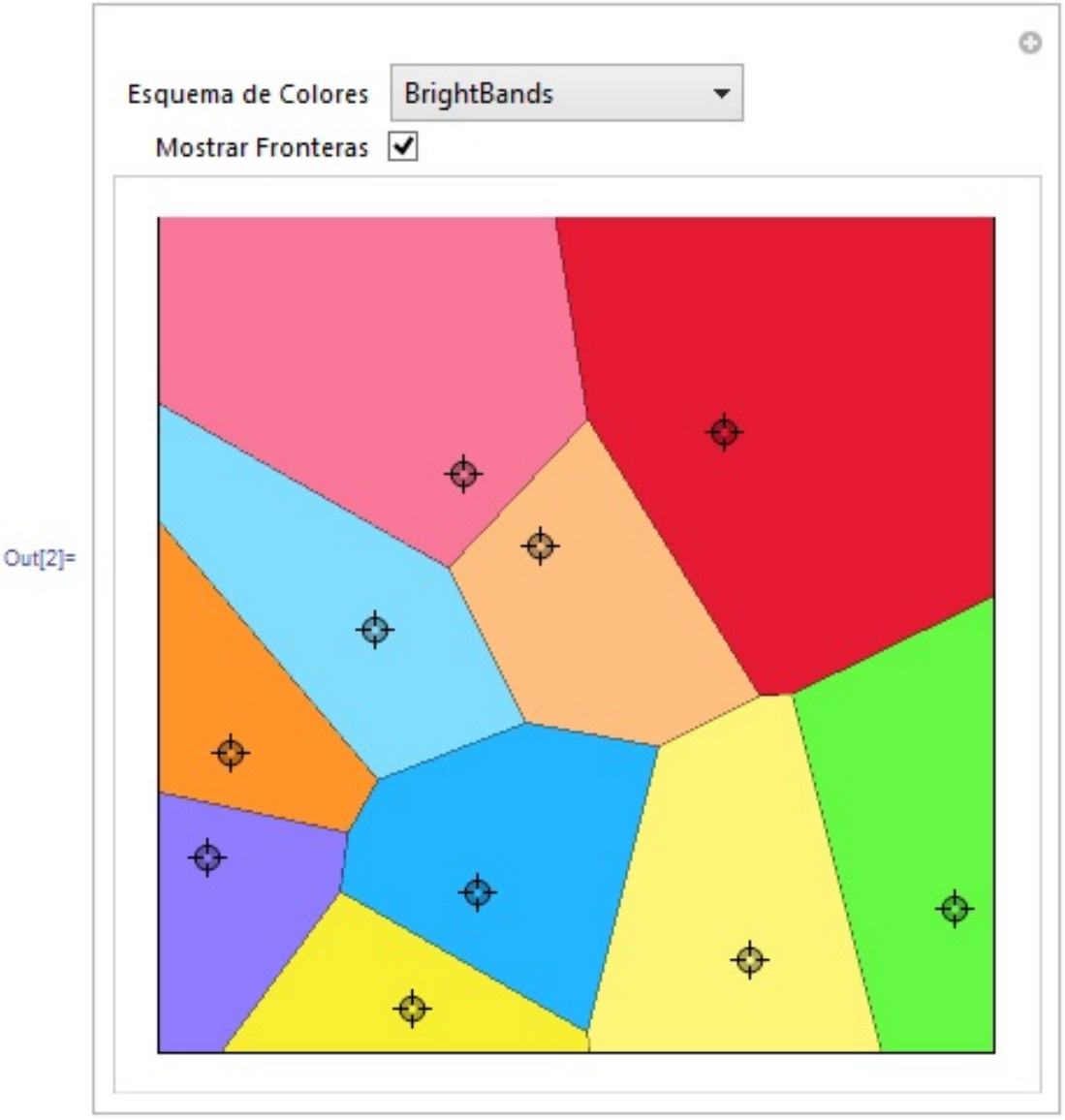


FIGURA B.1: Diagrama de Voronoi realizado con Mathematica.

Bibliografía

[1] **Shamos, M. I.,** *The early years of Computational Geometry, a personal memoir.,* Contemporary Mathematics, pg. 313-332, 1999.

[2] **Davies, J.,** *Freelance data visualization and Computer Science.,* recuperado de http://www.jasondavies.com/.

[3] **Gómez, J. R.,** *Diagramas de Voronoi.,* Matemática Aplicada, Universidad de Sevilla, pg. 8-12.

[4] **Aurenhammer, F., Klein, R.,** *Voronoi Diagrams.,* Cap. 5 en *Handbook of Computational Geometry.* (Ed. J.-R. Sack and J. Urrutia). Amsterdam, Netherlands: North-Holland, pg. 203, 2000.

[5] **Dirichlet, G. L.,** *Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen.,* J. reine angew. Math. 40, pg. 209-227, 1850.

[6] **Doval, H.,** *John Show y la epidemia de cólera en Londres en 1854.,* Revista Argentina de Cardiología, Vol. 71, No. 6, pg. 463-467, 2003.

[7] **Manzoor, S., et al.,** *Boundary Aligned Grid Generation and CVD-MPFA-Cell-centred Versus Cell-vertex on Unstructured Grids.,* ECMOR XIV-14th European conference on the mathematics of oil recovery, 2014.

[8] **Okabe, A., Boots, B., Sugihara, K., y Chiu, S. N.,** *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.,* 2^a Edición, Ed. John Wiley and Sons, pg. 6-12, 1999.

[9] **Frank, F. C., Kasper, J. S.,** *Complex alloy structures regarded as spherical packings. I. Definitions and basic principles.,* Acta Crystallogr. 11, pg. 184-190, 1958.

[10] **Brown, G.S.,** *Point density in stems per acre.,* New Zealand Forest Research Note 38, pg. 1-12, 1965.

[11] **Turek, Z., Hoofd, L., Batra, S., Rakusan, K.,** *The Effect of Realistic Geometry of Capillary Networks on Tissue PO_2 in Hypertrophied Rat Heart.,* Oxygen Transport to Tissue XIV. Advances in Experimental Medicine and Biology, Vol. 317, pg. 567-572, 1992.

[12] **Icke, V., van de Weygaert, R.,** *Fragmenting the Universe I. Statistics of two-dimensional Voronoi foams.,* Astron. Astrophys., Vol. 184, pg. 16-32, 1987.

[13] **Okabe, A., Boots, B., Sugihara, K., y Chiu, S. N.,** *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.,* 2^a Edición, Ed. John Wiley and Sons, pg. 221-224, 1999.

[14] **Haag, C.,** *Die Mundarten des oberen Neckar- und Donautales (Schwbisch-alemannisches Grenzgebiet: Baarmundarten).,* Hutzler, Reutlingen, 1898.

[15] **Bogue, D. J.,** *The Structure of the Metropolitan Community: A Study of Dominance and Sub-dominance.,* University of Michigan: Contributions of the Institute for Human Adjustment, Social Science Research Project, University of Florida Libraries, pg. 14-30.

[16] **Okabe, A., Boots, B., Sugihara, K., y Chiu, S. N.,** *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.,* 2^a Edición, Ed. John Wiley and Sons, pg. 119, 1999.

[17] **Kopec, R. J.,** *An alternative method for the construction of Thiessen Polygons.,* The Profesional Geographer, Vol. 5, 5^a Edición, pg. 24-26, 1963.

[18] **Shamos, M. I., Hoey, D.,** *Closest-point problems.,* 16th Annual Symposium on Foundations of Computer Science, Ed. IEEE, pg. 151-162, 1975.

[19] **Gruber, P. M.,** *Convex and Discrete Geometry.,* A Series of Comprehensive Studies in Mathematics, Vol 336, Ed. Springer, pg. 467, 2007.

[20] **Euler, L.,** *Solutio problematis ad geometriam situs pertinentis.,* originalmente publicado en *Commentarii academiae scientiarum Petropolitanae 8*, pg. 128-140, 1741.

[21] **Rosen, K. H.,** *Matemática discreta y sus aplicaciones.,* 5^a Edición, Ed. McGraw-Hill, pg. 503-581, 2004.

[22] **Skiena, S.,** *The Stony Brook Algorithm Repository.,* Stony Brook University, sitio oficial en http://www3.cs.stonybrook.edu/~algorith/implement/nauty/implement.shtml.

[23] **Brualdi, R. A.,** *Introductory Combinatorics.,* Ed. North-Holland, 1977.

[24] **Hsu, L., Lin, C.,** *Planar Graphs.,* Cap. 10 en *Graph Theory and Interconnection Network*, Ed. CRC Press, pg. 165, 2008.

[25] **Rosen, K. H.,** *Matemática discreta y sus aplicaciones.,* 5^a Edición, Ed. McGraw-Hill, pg. 568, 2004.

[26] **West, D. B.,** *Introduction to Graph Theory.,* 3^a Edición, Ed. Prentice Hall, 2007.

[27] **Okabe, A., Boots, B., Sugihara, K., y Chiu, S. N.,** *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.,* 2^a Edición, Ed. John Wiley and Sons, pg. 17, 1999.

[28] **Martinetz T., Schulten K.,** *Neural Networks.,* Topology representing networks, Vol. 7, pg. 507-522, 1994.

[29] **Dehne, F.,** *Computing digitized Voronoi diagrams on a systolic screen and applications to clustering.,* Lecture Notes in Computer Science, Vol. 401, pg. 14-24, 1989.

[30] **ArcGIS.,** *ArcGIS Resource Center.,* recuperado de http://help.arcgis.com/es/arcgisdesktop/10.0/help/index.html#//009t00000002000000, Copyright © 1995-2012 Esri. Todos los derechos reservados.

[31] **Toriwaki, J. I., Yokoi, S.,** *Voronoi and related neighbors on digitized 2-dimensional space with application of texture analysis.,* Computational Morphology, Ed. Toussaint, North-Holland, 1988.

[32] **Watanabe, T., Murashima, S.,** *A method to construct a Voronoi diagram on 2-D digitized space (1) computing time.,* Faculty of Engineering, Kagoshima, 1996.

[33] **Siersma, D., Tibar, M.,** *Topology of polynomial functions and monodromy dynamics.,* C. R. Acad. Sci. Paris, T. 327, Serie I, pg. 655-660, 1998.

[34] **Rodríguez, E. A.,** *Cubiertas convexas II.,* CINVESTAV-Tamaulipas, pg. 3-38, 2013.

[35] **De Berg, M., Cheong, O., Van Krevel, M., Overmars M.,** *Computational Geometry: Algorithms and Applications.,* Ed. Springer-Verlag, pg. 191-218, 2008.

[36] **Valenzuela, P. D.,** *Implementación de una biblioteca de triangulación de polígonos basada en el algoritmo Lepp Delaunay.,* Universidad de Chile, Departamento de Ciencias de la Computación, pg. 14, 2009.

[37] **Abellanas, M.,** *Envolvente Convexa, Triangulación de Delaunay y Diagrama de Voronoi: tres estructuras geométricas en una, con muchas aplicaciones.,* Art. en *Un paseo por la geometría*, Universidad Politécnica de Madrid, pg. 161-165, 2006/2007.

[38] **Rodríguez, E. A.,** *Triangulaciones de Delaunay.,* CINVESTAV-Tamaulipas, pg. 13-14, 2013.

[39] **Fortune, S.,** *A fast algorithm for Voronoi Diagrams.,* Algorithmica 2, pg. 153-174, 1987.

[40] **Brown, K. Q.,** *Voronoi diagrams from convex hulls.,* Information Processing Letters, Vol. 9, No. 5, pg. 223-228, 1979.

[41] **Edelsbrunner, H., Seidel, R.,** *Voronoi diagrams and arrangements.,* Discrete Comput. Geom. I, pg. 25-44, 1986.

[42] **Klein, R.,** *Abstract Voronoi diagrams and their applications (extendend abstract).,* Proc. Computational Geometry and its Applications (CG′88), Lecture Notes in Computer Science, Vol. 333, Ed. H. Noltemeier, pg. 148-157, 1988.

[43] **Fortune, S.,** *A sweepline algorithm for Voronoi diagrams.,* Algorithmica 2, pg. 153-174, 1987.

[44] **Rivero, F.,** *Diagrama de Voronoi.,* Cap. 4 en *Geometría Computacional*, Universidad de los Andes, pg. 60-64.

[45] **Rivero, F.,** *Diagrama de Voronoi.,* Cap. 4 en *Geometría Computacional*, Universidad de los Andes, pg. 48-49.

[46] **Pelegrín, B,** *Un algoritmo para determinar las medianas absolutas generales sobre una red tipo árbol.,* Trabajos de estadística y de investigación operativa, Vol. 33, No. 1, pg. 54-63, 1982.

[47] **O′Rourke, J.,** *Computational Geometry in C.,* 3^a Edición, Ed. Cambridge University Press, pg. 64, 1998.

[48] **Abellanas, M.,** *Diagramas de Voronoi.,* recuperado de http://www.dma.fi.upm.es/mabellanas/voronoi/voronoi/voronoi.html#3.3.4.

[49] **Valenzuela, P. D.,** *Implementación de una biblioteca de triangulación de polígonos basada en el algoritmo Lepp Delaunay.,* Universidad de Chile, Departamento de Ciencias de la Computación, pg. 18-28, 2009.

[50] **Guibas, L., Stolfi, J.,** *Primitives for the manipulation of general subdivisions and the computation of Voronoi.,* ACM Trans. Graph., Vol. 4, No. 2, pg. 74-123, 1985.

[51] **Chew, L.,** *Guaranteed-quality triangular meshes.,* Department of Computer Science, Technical Report, Cornell University, Report 89-983, 1989.

[52] **Posada, E. S.,** *Reducción y ajuste de mallas triangulares.,* Universidad Nacional de Colombia, pg. 6, 2013.

[53] **Thompson, J., Soni, B., Weatherill N.,** *Handbook of Grid Generation.,* Ed. CRC Press, 1998.

[54] **Fortune, S.,** *Voronoi diagrams and Delaunay triangulations in Computing in Euclidean Geometry.,* World Scientific, River Edge, NJ, pg. 193-233, 1992.

[55] **Frey, P., Borouchaki, H., George, P.,** *Delaunay tetrahedralization using an advancing-front approach.,* Proc. Fifth Internat. Meshing Roundtable, Sandia National Lab., pg. 87-106, 1996.

[56] **Miller, G., Talmor, D., Teng, S., Walkington, N., Wang, H.,** *Control volume meshes using sphere packing: generation, refinement and coarsening.,* Proc. Fifth Internat. Meshing Roundtable, Sandia National Lab., pg.47-62, 1996.

[57] **Ruppert, J.,** *A Delaunay refinement algorithm for quality 2-dimensional mesh generation.,* J. Algorithms, Vol. 18, pg. 548-585, 1995.

[58] **Du, Q., Faber, V., Gunzburger, M.,** *Centroidal Voronoi tessellations: applications and algorithms.,* SIAM Rev.41, pg. 637-676, 1999.

[59] **Zhao, Y., Liu, S., Zhang, Y.,** *Spatial Density Voronoi Diagram and Construction.,* Journal of Computers, Vol. 7, No. 8, pg. 2007-2014, 2012.

[60] **Wang, J., Wang, X.,** *Edge-Weighted Centroidal Voronoi Tessellations.,* Numer. Math. Theor. Meth. Appl., Vol. 3, No. 2, pg. 223-244, 2010.

[61] **Bottou, L., Vapnik, V.,** *Neural computation.,* Local learning algorithms, Vol. 4, No. 6, pg. 888-900, 1992.

[62] **Hedenfalk, I., et al.,** *Gene expression profiles in hereditary breast cancer.,* New England Journal of Medicine, Vol. 344, No. 8, pg. 539-548, 2001.

[63] **Kalos, M. H., Whitlock, P. A.,** *Monte Carlo methods.,* Ed. John Wiley & Sons, 2008.

[64] **Ju, L., Du, Q., Gunzburger, M.,** *Probabilistic methods for centroidal Voronoi tessellations and their parallel implementations.,* Parallel Computing, Vol. 28, pg. 1477-1500, 2001.

[65] **Lloyd, S.,** *Least square quantization in PCM.,* IEEE Transactions on Information Theory, Vol. 28, No. 2, pg. 129-137.

[66] **Du, Q., Gunzburger, M.,** *Grid generation and optimization based on centroidal Voronoi tessellations.,* Applied Mathematics and Computation 133, Ed. Elsevier, pg. 591-607, 2002.

[67] **Eiseman, P.,** *Adaptive grid generation.,* Computer Methods in Applied Mechanics and Engineering, Vol. 64, pg. 321-376, 1987.

[68] **Freitag, L., Jones, M., Plassmann, P.,** *A parallel algorithm for mesh smoothing.,* SIAM Journal on Scientific Computing, Vol. 20, pg. 2023-2040, 1999.

[69] **McRae, D., Laflin, K.,** *Dynamic grid adaptation and grid quality.,* Handbook of Grid Generation, Ed. CRC Press, pg. 34-1-34-33, 1998.

[70] **MeshLab,** *Sitio Oficial de MeshLab,* recuperado de http://meshlab.sourceforge.net/.

[71] **VCG,** *Sitio Oficial de la Librera VCG,* recuperado de http://vcg.isti.cnr.it/vcglib/.

[72] **Curless, B., Levoy, M.,** *A Volumetric Method for Building Complex Models from Range Images.,* SIGGRAPH, pg. 302-312, 1996.

[73] **Visual Computing Lab ISTI,** *Sitio Oficial del Laboratorio de Computación Visual ISTI,* recuperado de http://www.isti.cnr.it/.

[74] **Loop, C. T.,** *Smooth Subdivision Surfaces Based on Triangles.,* M.S. Mathematics thesis, The University of Utah, 1987.

[75] **McCool, M., Fiume, E.,** *Hierarchical Poisson Disk Sampling Distributions.,* University of Toronto, 1992.

[76] **Corsini, M., Cignoni, P., Scopigno, R.,** *Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes.,* IEEE Transaction on Visualization and Computer Graphics, Vol. 18, No. 6, pg. 914-924, 2012.

[77] **Lorensen, W. E., Cline, H. E.,** *Marching Cubes: A Hight-Resolution 3D Surface Construction Algorithm.,* Computer Graphics, Vol. 21, No. 4, pg. 163-169, 1987.

[78] **Taubin, G.,** *A Signal Processing Approach To Fair Surface Design.,* Proc. of SIGGRAPH '95, pg. 351-358, 1995.

[79] **Snow, J.,** *On the Mode of Communication of Cholera.,* Ed. John Churchill, 1855.

[80] **GeoGebra,** *Sitio Oficial de GeoGebra.,* recuperado de http://www.geogebra.org/.

[81] **Johnson, H.** *Historia del ataque japonés a Pearl Harbor.,* recuperado de http://goo.gl/G5veUr.

[82] **Turing, A. M.,** *The Chemical Basis of Morphogenesis.,* Philosophical Transactions of the Royal Society of London, Series B, Biological Sciences, Vol. 237, No. 641, pg. 37-72, 1952.