VISUALIZACIÓN DE SUPERFICIES EN 3D POR MEDIO DE DIAGRAMAS DE VORONOI

GERMAN EUTIMIO SORA YANQUÉN

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICAS
BUCARAMANGA
2007

VISUALIZACIÓN DE SUPERFICIES EN 3D POR MEDIO DE DIAGRAMAS DE VORONOI

GERMAN EUTIMIO SORA YANQUÉN

DIRECTOR:

Doctor MARLIO PAREDES GUTIÉRREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICAS
BUCARAMANGA
2007

TABLA DE CONTENIDO

AGRADECIMIENTOS

A Dios por permitirme conocer la ciencia sin apartarme de su camino.

A mis Padres quienes desde siempre se han sacrificado por mi bienestar.

A todos los Profesores quienes me han formado integralmente.

A mi Director por sus consejos y por darme siempre una posibilidad para seguir creciendo.

RESUMEN

TITULO:
VISUALIZACIÓN DE SUPERFICIES EN 3D POR MEDIO DE DIAGRAMAS DE VORONOI[*]

AUTOR:
SORA Yanquén, German Eutimio.[**]

PALABRAS CLAVES:
Polígono, Triangulación, Envolvente Convexa, Diagrama de Voronoi, Triangulación de Delaunay, Frente de Onda.

DESCRIPCIÓN

Este trabajo describe tres de las estructuras más importantes en la Geometría Computacional que son la Envolvente Convexa, el Diagrama de Voronoi y la Triangulación de Delaunay de una nube de puntos en el espacio. La mayor parte de los contenidos se basan en el texto de Geometría Computacional en C de Joseph O'Rourke A través del trabajo, se presentan de manera progresiva los algoritmos que permiten construir las estructuras mencionadas, hasta llegar a diseños óptimos en su tiempo de ejecución.

En el primer capítulo se plantea el Problema de la Galería de Arte de Klee con el fin de enmarcar la importancia de la triangulación; una vez resuelto dicho problema, se procede a encontrar el algoritmo que realice la mejor partición de un polígono; este debe dividir dicho polígono en el menor número de partes convexas en el menor tiempo posible. En el segundo capítulo se define la envolvente convexa de una nube de puntos y se presentan los algoritmos que permiten calcularla. En el tercer capítulo se define el Diagrama de Voronoi de una nube de puntos, su estrecha relación con la triangulaciones de Delaunay y se presenta un código en lenguaje C con el cual se puede calcular el diagrama de Voronoi en el plano, apoyándose en su triangulación de Delaunay, obtenida mediante el cálculo de la envolvente convexa de la proyección de dichos puntos sobre un paraboloide. El trabajo finaliza con la presentación de un algoritmo que permite visualizar el comportamiento de un frente de onda en determinada superficie.

[*] Monografía en la modalidad de Revisión Bibliográfica.
[**] Facultad de Ciencias, Escuela de Matemáticas, Marlio Paredes Gutiérrez

ABSTRACT

TITLE:
SURFACES VISUALIZATION IN 3D USING VORONOI DIAGRAMS*

AUTHOR:
SORA Yanquén, German Eutimio.**

KEY WORDS:
Polygon, Triangulation, Convex Hull, Voronoi Diagram, Delaunay Triangulation, Wave front.

DESCRIPTION:

This paper describes three of the most important structures in the Computational Geometry; these are the Convex Hull, the Voronoi Diagram and the Delaunay Triangulation of a set of points in the space. Most of the contents are based on the text of Computational Geometry in C of Joseph O'Rourke. Through this paper, algorithms that progressively allow to build the aforementioned structures are presented, until arriving to good designs by their run-time.

The first chapter begins making reference to the Art Gallery Problem of Klee with the purpose of establishing the importance of the triangulation; solved this problem, next challenge is to find some algorithm that gets the best partition of a polygon; this have to divide the polygon in the smallest number of convex parts in the smallest possible time. In the second chapter, the convex hull of a cloud of points is defined and the algorithms that allow to calculate it are presented. In the third chapter, the Voronoi diagram of a cloud of points is defined, as well as their narrow relationship with the Delaunay triangulations and a code in C is presented, which calculate the Voronoi diagram in the plane, by using its Delaunay triangulation; this is obtained by calculating the convex hull of the points projection on a paraboloide. Paper finishes with the presentation of an algorithm that allows to visualize the behavior of a wave front on some given surface.

* Monograph in modality of Bibliographical Revision.
** Faculty of Sciences, Mathematics School, Marlio Paredes Gutiérrez.

INTRODUCCIÓN

La invención del computador trajo a la industria, a la sociedad y a la ciencia la herramienta más potente en cuanto a la agilización de procedimientos mecánicos, la producción en cadena y el desarrollo de cálculos que anteriormente eran muy complicados y muchas veces considerados imposibles.

En la industria en particular, el estudio de yacimientos e hidrocarburos en muchas ocasiones se realiza mediante la manipulación de una cantidad enorme de datos y variables que serán trabajadas con procesos repetitivos extenuantes para cualquier persona, de modo que se hace necesario el diseño de modelos que se ajusten de la mejor manera a todas estas variables para luego, por medio de la máquina, obtener resultados y así poder sacar conclusiones.

La Geometría Computacional desde hace varios años ha proveído de diversos algoritmos geométricos de gran utilidad en campos como la Matemática Discreta, el Análisis Combinatorio, el Diseño de Algoritmos y Estructuras de Datos, la Técnica Aplicada en Robótica, CAD (Dibujo Asistido por Computador), SIG (Sistemas de Información Geográfica), entre otros, pero es difícil encontrar alguno que se ajuste exactamente a nuestras necesidades. Sin embargo, por medio de su estudio, es posible adaptar estos algoritmos a propósitos específicos tales como el modelamiento de superficies, la caracterización de espacios o la generación de frentes de onda.

Actualmente se realizan encuentros a nivel mundial relacionados con este campo, por ejemplo, en la Universidad de Sevilla -España-, se estudian a fondo temas de Geometría Computacional y Matemática Discreta, y se organiza cada 2 años el Encuentro de Geometría Computacional, celebrando el undécimo en el año 2005 en Barcelona.

En este trabajo se presentan tres estructuras geométricas que al final serán reunidas en un algoritmo que iteración tras iteración se apoya en ellas para el modelamiento de una perturbación en determinada superficie.

En el primer capítulo se presentan las primeras definiciones y algoritmos utilizados para la triangulación de polígonos. Debido a que son pocas las figuras planas que tienen bien definidas sus características, triangular dichas figuras permite su análisis en forma modulada, es decir, observando por separado cada una de sus partes pero sin dejar de entenderla como un todo.

En el segundo capítulo, se introducirá la envolvente convexa de una nube de puntos por medio de varias definiciones. Esta es simplemente el menor polígono convexo que “rodea” un conjunto arbitrario de puntos en el plano, o el poliedro convexo “más pequeño” que rodea una nube de puntos en el espacio. Más adelante, por medio de ella se podrá construir un algoritmo para el cálculo de otras estructuras.

En el tercer capítulo se presenta el diagrama de Voronoi de un conjunto de puntos en el plano como el lugar geométrico que ocupan los puntos en dicho plano que son equidistantes a dos o más puntos del conjunto inicial, siempre y cuando no haya otro punto con el cual estén aun mas cerca. De igual forma se define la triangulación de Delaunay de una nube de puntos como el grafo asociado al diagrama de Voronoi donde los puntos del conjunto dado pasan a ser los nodos del grafo y dos nodos estarán relacionados siempre y cuando posean puntos comunes en el diagrama, es decir que existan puntos en el plano que sean equidistantes a los nodos relacionados y dicha distancia sea menor que con cualquier otro nodo del conjunto. Con el apoyo de la relación biunívoca de estas dos últimas estructuras, la envolvente convexa en tres dimensiones y las triangulaciones vistas en el primer capítulo, se diseña un algoritmo con el cual se construye el diagrama de Voronoi.

El trabajo finaliza en el cuarto capítulo donde se plantea cómo por medio del algoritmo visto en el capítulo tres, se puede dividir una superficie en particiones apropiadas a fin de modelar el comportamiento del frente de onda generado en algún punto de dicha superficie, para lo cual se presentan algunos conceptos previos relacionados con el tratamiento de ondas cuando se propagan y atraviesan diferentes medios.

Capítulo 1

Particiones de Polígonos

1.1. INTRODUCCIÓN.

Uno de los problemas clásicos de la geometría computacional es el problema de la "Galería de Arte" de Klee, que consiste en averiguar el menor número de guardias necesarios para vigilar el área de una sala en un museo, cuyo plano se puede representar como un polígono y los guardias como puntos en él y debe cumplirse que para cualquier otra sala con igual cantidad de muros, la misma cantidad de guardias sea suficiente. Otro, consiste en escanear y reconocer caracteres haciendo comparaciones con una base de datos previamente establecida, lo cual es muy útil para la digitalización de textos. Uno más, es el reconocimiento de una superficie por medio de un robot; como en el caso anterior, el robot debe visualizar la superficie, compararla con su base de datos y encontrar la mejor división de la misma a fin de hacer interpolaciones, extrapolaciones y obtener un buen modelo de dicha superficie. Estos problemas pueden ser abordados mediante algoritmos geométricos pero con la dificultad que demandan un alto "costo computacional" y por eso es necesario simplificar dichos algoritmos hasta obtener un orden lineal.
Los algoritmos discutidos en el presente capítulo, evolucionarán buscando siempre una partición óptima para polígonos en el sentido que proporcione el menor número de partes en el menor tiempo posible.

1.2. REPRESENTACIÓN DE POLÍGONOS.

Definición 1.1.
Un *polígono* es una región del plano limitada por una colección finita de segmentos de recta formando una *curva cerrada simple.*

La palabra *curva* hace referencia al hecho de que los segmentos de recta estén conectados uno a continuación del otro. La palabra *cerrada* puede estar siendo mal utilizada, pero indica que el arreglo de estos segmentos es cíclico, es decir que el último se enlaza con el primero. La palabra *simple* significa que los segmentos no adyacentes no se intersecan.

Sin importar las apreciaciones que pueda tener, la definición anterior no es muy útil para fines computacionales, así que es mejor expresarla de la siguiente manera:

Definición 1.2.
Sean $v_0, v_1,\ldots, v_{n-1}$ un conjunto de puntos en el plano llamados *vértices* y $e_0 = v_0v_1$, $e_1 = v_1v_2,\ldots, e_i = v_iv_{i+1},\ldots, e_{n-1} = v_{n-1}v_0$,[1] una colección de n segmentos conocidos como lados y cuyos extremos son los puntos anteriores. Estos segmentos *delimitan* un *polígono* en el plano si y solo si:

i. $e_i \cap e_{i+1} = v_{i+1}$,[2] para todo $i = 0, 1,\ldots, n-1$.
ii. $e_i \cap e_j = \varnothing$, para todo $j \neq i+1$ o $j \neq i-1$.

Nota: Por el teorema de la "*Curva de Jordan*", la región anterior divide el plano en dos conjuntos, así que la palabra *delimitan* se refiere específicamente a que el polígono es la región compacta[3], incluyendo su frontera.
La demostración de dicho teorema se sale del alcance e interés de este documento de manera que no se profundizará en ello.
Los vértices de los polígonos serán numerados en forma antihoraria comenzando desde *0* hasta *n-1*.

Ejemplo 1.1.
La Figura 1.1 (a) queda por fuera de la Definición 1.2 ya que contradice el segundo ítem. La Figura 1.1 (b) tampoco puede ser considerada como un polígono ya que no cumple con el primer ítem, la intersección de dos de sus lados consecutivos es un punto interior de los mismos y no sus respectivos extremos. Por otra parte, las Figuras 1.1 (c) y 1.1 (d) satisfacen todas las características de un polígono. Note que la convexidad de dichas figuras no interfiere a la hora de considerarlas o no como polígonos. La convexidad es un concepto que será abordado en la Sección 1.3.2.

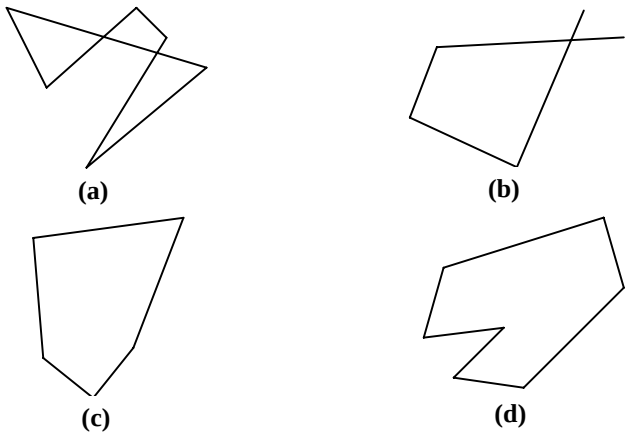


Figura 1.1. Ejemplos de Polígonos.

1 Para facilitar la escritura, será omitida la barra superior en la notación de segmento.
2 Los subíndices son trabajados en módulo n de manera que para $i = n-1$, $i+1 = n \equiv 0$.
3 Para una definición de compacidad ver APOSTOL, Tom M., Análisis Matemático, pág 71.

Con base en la Definición 1.2 se puede empezar a obtener resultados para los polígonos que a simple vista pueden parecer evidentes pero que demandan una demostración y serán de mucha importancia en resultados posteriores.

Lema 1.1.
Sea P un polígono, entonces existe un vértice v de P tal que para todo $p \in P$, $v_y \leq p_y$, donde v_y y p_y son respectivamente las componentes verticales de v y p.

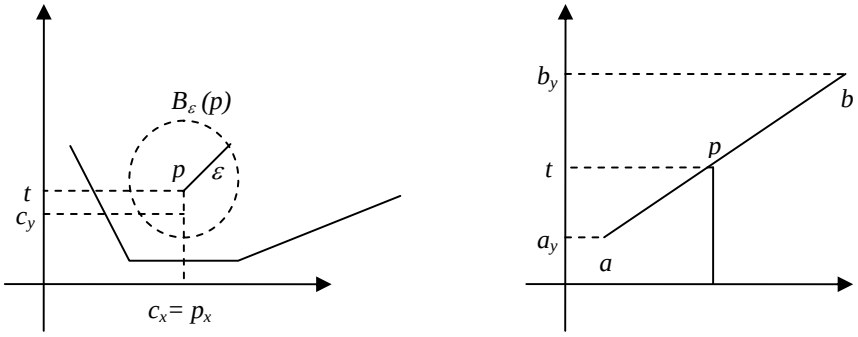


Figura 1.2. Existe un vértice que es cota inferior del Conjunto.

Demostración:
Haremos una demostración por contradicción:

Como P es compacto[4] y la función que a cada punto de P le asocia su componente vertical es continua[5], entonces la imagen que deja P sobre el eje vertical es compacta. Así, dicha imagen es cerrada y acotada[6], luego tiene un punto mínimo.

Sea t dicho punto, entonces para todo p de P, $t \leq p_y$.

Suponga que para todo p de P tal que $p_y = t$, se tiene que p está en el interior del polígono. De esta manera existe una bola abierta $B_\varepsilon(p)$ que está estrictamente contenida en P.

Sea c un punto de P tal que $c_x = p_x$ y $c_y = p_y - \varepsilon/2$; se puede comprobar que $c \in B$ y por ende a P, luego existe un punto de P con componente vertical inferior a t lo cual contradice el hecho de que t es la menor imagen del polígono bajo la función "componente vertical", luego la suposición hecha no es cierta y existe por lo menos un punto en la frontera del polígono con componente vertical igual a la del punto mínimo.

Suponga ahora que para todo p de δP tal que $p_y = t$, se tiene que p está en el interior de algún lado del polígono, es decir que no es un vértice. Así, la componente vertical de los dos vértices de dicho lado debe ser mayor que la componente vertical de p, lo cual contradice el hecho de que estos vértices y p conforman un lado ya que no es posible asociar ninguna función lineal ente sus las componentes horizontales y verticales de dicho lado.

Por tanto, existe por lo menos un vértice v del polígono tal que $v_y = t$.

□

4 Vea la nota de la Definición 1.2
5 El lector puede verificar esta afirmación teniendo en cuenta que se están tomando todas las condiciones usuales en R^2.
6 En R^n con la topología usual, ser compacto es equivalente a ser cerrado y acotado.

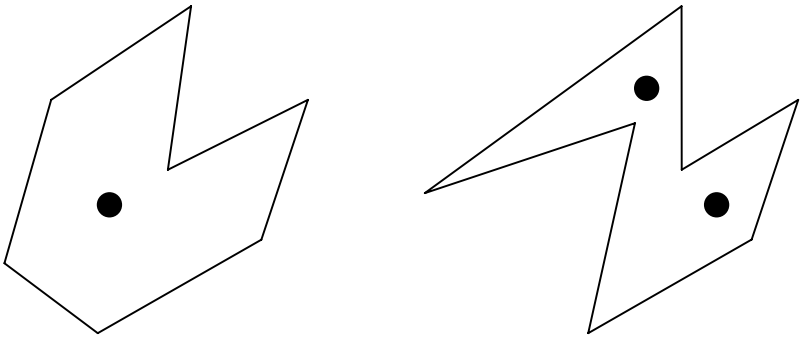
1.3. EL PROBLEMA DE LA GALERÍA DE ARTE Y LA TRIANGULACIÓN.

1.3.1. Planteamiento del Problema.

Suponga que la Figura 1.3 representa la vista superior de dos salas de un museo. ¿Cuál es el número de guardias necesarios para vigilar dichas salas?[7] Note que la palabra necesarios lleva inmediatamente a buscar el menor número de guardias posibles.

Definición 1.3.
Sean $S \subset \boldsymbol{R^2}$ y P_n un polígono. S ***cubre*** a P_n si y solo si para todo $p \in P_n$, existe un $s \in S$ tal que $sp \subset P_n$.

El polígono de la Figura 1.3 (a) posee *7* vértices y puede ser cubierto por un solo guardia. Así, es tentador pensar que para cualquier polígono de *7* vértices, la solución del problema de Klee es *1*, pero la Figura 1.3 (b) refuta inmediatamente esta suposición.



(a) **(b)**

Figura 1.3. Soluciones para $n = 7$.

Estos cambios ocurren por diversas razones -entre las cuales se encuentra la convexidad del polígono- y hacen que el problema se torne interesante.
El problema puede ser reformulado más exactamente de la siguiente manera:

Sea S un conjunto de puntos en el plano, P_n un polígono de n vértices y $g(P_n)$ el número de guardias necesarios para vigilar el área de P_n, entonces

$$g(P_n) = mín_s\{|\, S \,| : S \text{ cubre a } P_n\}, \qquad \textbf{(1.1)}$$

donde las barras horizontales indican el cardinal del conjunto.

7 Los guardias pueden ser considerados como puntos en el área de la sala y no pueden vigilar lugares a través de los muros.

Así,

$$G(n) = máx_P\{g(P_n)\} \qquad \textbf{(1.2)}$$

es el menor número de guardias que cubren cualquier polígono de *n* vértices.

Hasta el momento no se ha encontrado una fórmula que permita hallar exactamente el valor de *G(n)*, pero en 1975 Chvátal logró establecer un buen límite superior de $\lfloor n/3 \rfloor$ el cual será enunciado y demostrado en el Teorema 1.2 después de la definición de algunos conceptos.

1.3.2. Triangulación de Polígonos.

La mejor prueba del resultado de Chvátal no fue precisamente la suya, la cual realizó por inducción, sino que fue dada por Fisk en 1978 y se basa en un concepto que tomará gran importancia de aquí en adelante pues permite una de las mejores particiones de polígonos y es la triangulación. En palabras sencillas, triangular -como su nombre lo dice- es dividir un polígono en triángulos de manera que dichas partes no queden sobrepuestas. Las triangulaciones que se usarán serán por diagonales, es decir donde los lados de los triángulos son diagonales o lados del polígono. A continuación se irán definiendo estos conceptos.

Definición 1.4. **Diagonales de un Polígono.**
Sean *V*, *E* respectivamente el conjunto de vértices y lados de un polígono *P*. Se dice que $d = e_i e_j$ es una *diagonal* de *P* si y solo si *d* satisface las siguientes condiciones:

i. $e_i, e_j \in V$,
ii. $d - \{ e_i, e_j \} \cap \delta P = \varnothing$ y
iii. $d \subset P$.

Nota: El conjunto de diagonales de un polígono no es único. Esta es una propiedad que el lector fácilmente puede comprobar y que se muestra en el Ejemplo 1.2.

Definición 1.5. **Convexidad.**
Sea *v* un vértice de cierto polígono *P*. Se dice que *v* es *convexo* si y solo si el ángulo $\angle v^- v v^+$ es menor o igual que π.

Nota: v^- y v^+ representan respectivamente los vértices anterior y siguiente a *v* en el orden cíclico establecido en la nota de la Definición 1.2 y el ángulo debe ser barrido en el interior del polígono. Por otra parte, si el ángulo $\angle v^- v v^+$ es estrictamente menor que π, se dice que *v* es *estrictamente convexo*.
Las definiciones de *cóncavo* y *estrictamente cóncavo* son análogas a las anteriores pero con el ángulo “mayor o igual” y “mayor estricto” respectivamente.

¿SIEMPRE ES POSIBLE TRIANGULAR UN POLÍGONO?
La respuesta a esta pregunta se puede resolver paso a paso mediante los siguientes lemas y será definitiva en el Teorema 1.1.

Lema 1.2.
Todo polígono tiene al menos un vértice estrictamente convexo.

Demostración:
Sea v un vértice de P tal que v_y (la componente vertical de v) es menor que p_y para todo $p \in P$, cuya existencia la garantiza el Lema 1.1. Como v no es necesariamente único, escoja el que está mas a la derecha. Vea la Figura 1.4.
Suponga que v^- y v^+ están a la misma altura que v. Así, como v es el que está más a la derecha, v^+ debe estar a su izquierda y se tiene que $v^-v \cap vv^+ \neq v$, lo que contradice el primer ítem de la Definición 1.2 de polígono, de manera que la componente vertical de v^- o v^+ o la de ambos es mayor que la de v. Como v es una cota inferior del polígono, todos los puntos de P deben estar por encima o a la misma altura de v. Así existe una recta que contiene a v (la horizontal) tal que v^-, v^+ y el interior del polígono, están en el mismo semiplano delimitado por la recta. Debido a que sobre esta recta es posible construir un ángulo llano, es decir igual a π, entonces el ángulo $\angle v^-vv^+ < \pi$.

□

Una vez se tiene este vértice convexo, el Lema 1.3 garantiza el trazo de diagonales para todo polígono.

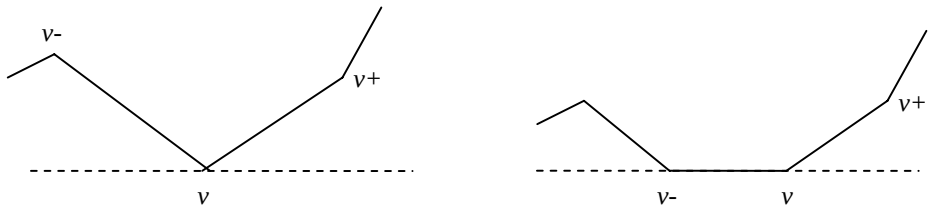


Figura 1.4. Vértice estrictamente convexo

Lema 1.3.
Todo polígono de cuatro o más vértices tiene al menos una diagonal.

Demostración:
Sea $V = \{v_0, v_1, \ldots, v_{n-1}\}$ el conjunto de vértices de cierto polígono P, donde n es mayor que 3.
Por el Lema 1.2 se sabe que existe por lo menos un vértice estrictamente convexo. Sea v_i dicho vértice. Si v_i^- y v_i^+ son los respectivos vértices anterior y siguiente de v_i y el segmento $v_i^-v_i^+$ es una diagonal, el lema queda demostrado.
Suponga que el segmento $v_i^-v_i^+$ no es una diagonal, entonces como v es convexo, el segmento no puede estar en el exterior de P así que $v_i^-v_i^+$ corta a alguno de sus lados. En este caso el triángulo $v_i^-v_iv_i^+$ contiene en su interior por lo menos un vértice v_j diferente de los tres ya mencionados, el cual garantiza su existencia por la hipótesis de que el polígono posee más de tres vértices.

Trace una recta paralela a $v_i^- v_i^+$ por cada vértice contenido en el triángulo $v_i^- v_i v_i^+$ y sea v_k el vértice de P en el interior del triángulo $v_i^- v_i v_i^+$ tal que la recta que lo contiene sea la más lejana a $v_i^- v_i^+$, entonces el segmento $v_i v_k$ es una diagonal de P. Vea la Figura 1.5.

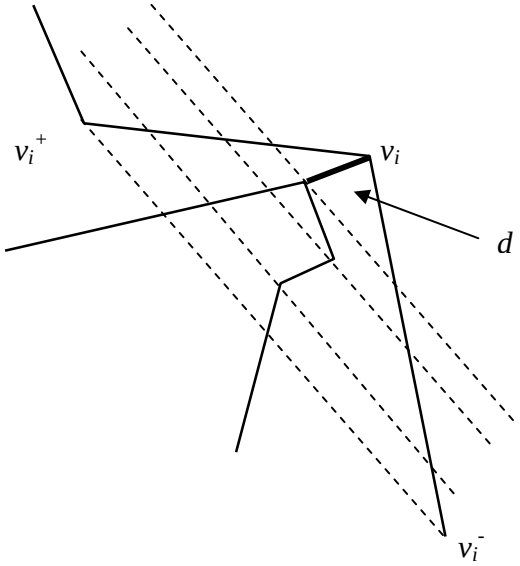


Figura 1.5. Existencia de las diagonales de un Polígono.

La respuesta definitiva a la pregunta anteriormente formulada es que efectivamente todo polígono puede ser triangulado por diagonales, lo cual queda explícito a continuación.

Teorema 1.1.
Todo polígono puede triangularse por diagonales.

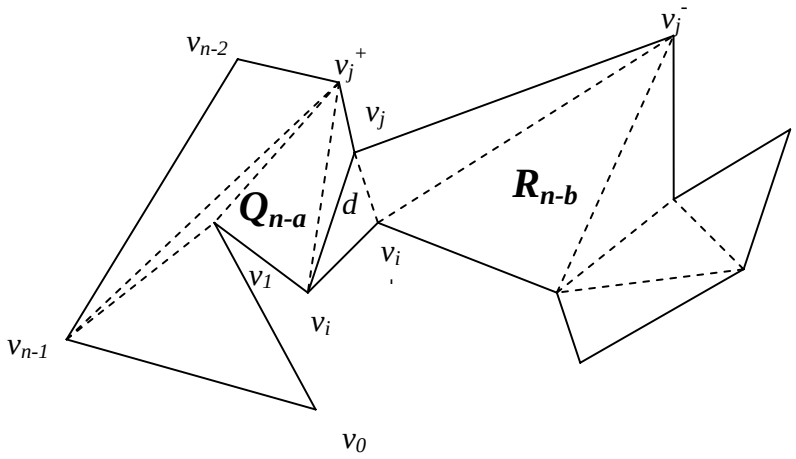


Figura 1.6. División de un polígono en dos para su triangulación.

Demostración:
Haremos una demostración aplicando el segundo principio de inducción matemática:
Sea P_n un polígono donde n representa el número de vértices y $V = \{v_0, v_1,\ldots, v_{n-1}\}$ el conjunto de sus vértices. Vea la Figura1.6.

i. Para $n = 3$: P_3 es un triángulo, de modo que el polígono ya está triangulado.
ii. Para $n \leq k$: Suponga que es posible triangular por diagonales a P_n.
iii. Para $n = k+1$: Sea $d = v_iv_j$, con $v_i, v_j \in V$, una diagonal de P_{k+1}, la cual garantiza su existencia por el Lema 1.3. Sean Q_{n-a}, R_{n-b} con $a+b = n-2$, los dos nuevos polígonos en los cuales queda dividido P_n y $V_a = \{v_0, v_1,\ldots, v_i, v_j, v_j^+,\ldots, v_{n-1}\}$, $V_b = \{v_j, v_i, v_i^+,\ldots, v_j^-\}$, sus respectivos conjuntos de vértices. Como Q_{n-a} y R_{n-b} tienen cada uno menos de $k+1$ vértices[8], por el segundo ítem son triangulables por diagonales, de manera que P_n también lo es.

Así, por el segundo principio de inducción matemática, se tiene que todo polígono P_n es triangulable por diagonales, para todo $n \in N_3$.

□

Una vez establecida la posibilidad de triangular polígonos, queda todo listo para el tricoloreado de los mismos.

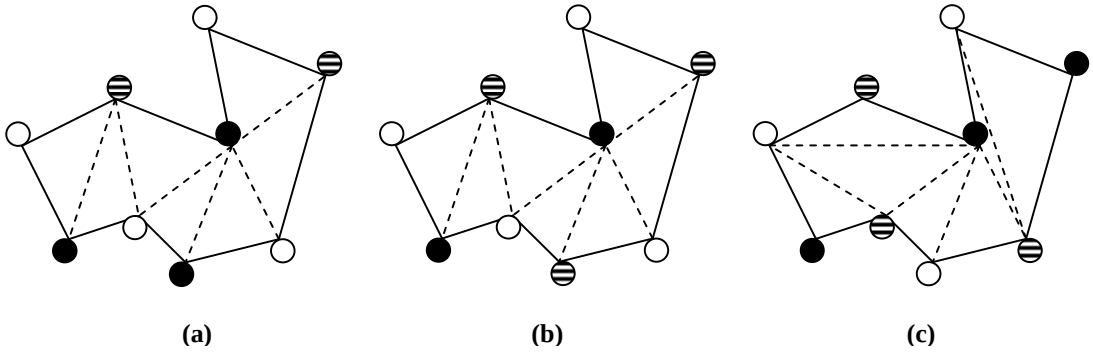
1.3.3. Tricoloreado y Delimitación del Problema de Klee.

Definición 1.6. **Tricoloreado.**
Sean V el conjunto de vértices de un polígono P, $L = E \cup D$, donde E es el conjunto de lados y D un conjunto de diagonales de dicho polígono y $C = \{c_1, c_2, c_3\}$ un conjunto de tres elementos llamados colores. La función $f: V \rightarrow C$ que a cada vértice le asigna un color, es un tricoloreado de P si y solo si para todo $v_i, v_j \in V$, $e \in L$, se tiene que $e = v_iv_j$ implica que $f(v_i) \neq f(v_j)$.

El concepto es mucho más sencillo de lo que parece, observemos el siguiente ejemplo para comprender un poco más la definición.

Ejemplo 1.2.
En la Figura 1.7 se muestran tres aparentes formas de tricolorear un polígono, la Figura (a) es un tricoloreado incorrecto ya que el color 1 está "conectado" por una diagonal, es decir que los extremos de dicha diagonal poseen el mismo color. En las Figuras (b) y (c) se observan dos buenos ejemplos de tricoloreado donde el número de colores varía según la ubicación de las diagonales.

8 Note que entre los dos extremos de la diagonal debe existir por lo menos un vértice pues de no ser así v_j sería el siguiente de v_i lo cual significaría que d es un lado de P_n y no una diagonal. Así, la cantidad de vértices en cada subdivisión es menor que la del polígono inicial.



(a) **(b)** **(c)**

Color 1.
Color 2.
Color 3.

Figura 1.7. Ejemplos de tricoloreado.

Volviendo al problema de la galería de arte, si el polígono de la Figura 1.3 representa una sala de un museo, el lector puede verificar que dicha sala puede ser vigilada por cualquier grupo de guardias del mismo color, ya que cada punto del polígono está contenido en algún triángulo y cada triángulo tiene un vértice de cada color.

Note que el tricoloreado no es precisamente la mejor manera de resolver el problema de Klee ya que el menor número de guardias varía según la triangulación realizada, por ejemplo en la Figura 1.7 (b) el menor número de guardias es dos (negro), mientras que en la Figura 1.7 (c) es 3 (cualquier color). Pero, aunque por medio de este método no se pueda establecer con precisión el número de guardias necesarios para cada valor de *n*, si se llega a una demostración sencilla del resultado de Chvátal.

El Teorema 1.2 establece una cota superior a las posibles soluciones del problema de Klee y se demuestra apoyándose en el tricoloreado y en un principio conocido como principio del palomar, el cual determina que dado un conjunto *A* y una partición *P(A)* del mismo, existe por lo menos un elemento P_i de la partición tal que $|P_i| \leq |A| / |P(A)|$.

Teorema 1.2.
$G(n) \leq \lfloor n/3 \rfloor$

Demostración:
Sea P_n un polígono de n vértices.
Triangule a P_n según garantiza el Teorema 1.1.
Tricoloree a P_n como se explica en la Definición 1.5.
Por el principio del palomar, se tiene que si se asocian n vértices a k imágenes, por lo menos una de las imágenes debe tener un número de preimágenes menor o igual a n/k, es decir que por lo menos uno de los tres colores debe ser usado no mas de $n/3$ veces. Además, como ningún color puede ser usado un número inexacto de veces, el límite pasa a ser $\lfloor n/3 \rfloor$, donde $\lfloor x \rfloor$ es el operador piso que aproxima al mayor entero menor o igual que x. De esta manera, para cada P_n existe un conjunto S de $\lfloor n/3 \rfloor$ vértices de cierto color que lo cubre, entonces $g(P_n)$ es menor o igual que $\lfloor n/3 \rfloor$, luego el máximo de los cardinales de dichos conjuntos también es menor que $\lfloor n/3 \rfloor$, es decir que $G(n)$ es menor o igual que $\lfloor n/3 \rfloor$.

□

1.4. POLÍGONOS MONÓTONOS.

En esta sección, se expondrá una partición mejor que la triangulación ya que es mucho más rápida, para ello, se requiere dividir primero la frontera del polígono en cadenas de vértices conocidas como *cadenas poligonales*.

Definición 1.7.
Sea $C \subset \delta P$. C es una *cadena poligonal* de P si y solo si C es conexo[9] y sus extremos son vértices de P.

La monotonía está definida con respecto a una línea:

Para las Definiciones 1.8 a 1.12, sean P un polígono, v un vértice de P, C una cadena poligonal de P, L una recta y L' cualquier recta perpendicular a L.

Definición 1.8.
C es *monótona* con respecto a L, si y solo si $L' \cap C$ contiene a lo sumo un componente, el cual es vacío, o un punto interior de un segmento, o un segmento.

Definición 1.9.
C es *estrictamente monótona* con respecto a L si y solo si toda línea L' perpendicular a L corta a C a lo sumo en un punto, i.e., $L' \cap C$ es o vacío o un punto.

9 El concepto de conexidad hace referencia a que la cadena no puede ser expresada como la unión disyunta de dos conjuntos abiertos.

Definición 1.10.
P es *monótono* con respecto a *L,* si y solo si δP puede ser dividida en dos cadenas *A* y *B* tales que cada cadena sea monótona con respecto a *L*.

Nota: Cabe aclarar que la palabra "dividir" significará que dichas cadenas solo tienen en común sus extremos y que con la unión de las mismas se obtiene la frontera del polígono.
Con esta definición de polígono monótono se puede ver que es posible ordenar los vértices con respecto al eje *y* primero por cadenas y luego mezclarlos, lo cual se puede hacer en tiempo lineal, para ello, deben tomarse como extremos de cada cadena los vértices máximos del polígono (el de mayor y el de menor coordenada vertical).

Definición 1.11.
v es *cúspide interior* de *P,* si y solo si *v* es estrictamente cóncavo y la coordenada vertical de sus vértices adyacentes v^- y v^+ es mayor o igual que la de *v* (para ambos), conocida como *cúspide interior descendente*, o menor o igual que la de *v* (para ambos), conocida como *cúspide interior ascendente*.

Nota: No es posible que la coordenada vertical sea la misma para los tres, por lo establecido en la Definición 1.5 para concavidad.

Definición 1.12.
C es cóncava, si y solo si para todo $v \in C$, *v* es cóncavo o llano.

A continuación se enunciará y demostrará un lema importante relacionado con los polígonos monótonos, que será la base para la construcción del algoritmo de división de un polígono en piezas monótonas con el que se concluirá este capítulo.

Lema 1.4.
Si *P* no tiene cúspides interiores, entonces *P* es monótono.

Demostración:
Sea *P* un polígono no monótono con respecto a una recta vertical *L*. Entonces sean C_1 y C_2 las dos cadenas en las cuales está dividido *P* y cuyos extremos son los vértices más alto y más bajo de *P* con respecto a *L*. Así, alguna (o ambas) de estas cadenas no es monótona con respecto a *L*. Sea C_1 dicha cadena, entonces debe existir una perpendicular *L'* que interseque a dos componentes no conectados de C_1. Sean *a* el punto más a la derecha del componente que está a la izquierda y *b* el punto más a la izquierda del componente que está a la derecha.
Para el segmento *ab* -sin tener en cuenta los extremos- se tienen dos posibilidades, que esté en el exterior o en el interior del polígono.

Caso 1: *ab* está en el exterior del polígono.
Si la subcadena contenida desde *a* hasta *b* está por encima de dichos puntos, entonces debe contener un vértice más alto (y más a la derecha si hay varios) y en este caso dicho vértice es una cúspide interior por ser cóncavo y tener a sus vértices adyacentes a igual o menor altura, luego *P* tiene cúspides interiores. Vea la Figura 1.8.

Si la subcadena contenida desde *a* hasta *b* está por debajo de dichos puntos, entonces debe contener un vértice más bajo y en este caso dicho vértice es una cúspide interior por ser cóncavo y tener a sus vértices adyacentes a igual o mayor altura, luego *P* tiene cúspides interiores.

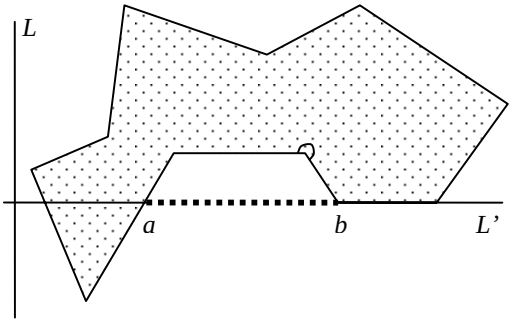


Figura 1.8. *ab* está en el exterior de *P*.

Caso 2: *ab* está en el interior del polígono.
Observando en detalle las diversas situaciones que se pueden presentar en este caso, se puede ver que los dos componentes a los cuales corta la perpendicular orientan el análisis conforme a su naturaleza.

- Si los componentes son lados o vértices, los puntos *a* y *b* serán vértices y si se quiere evitar que dichos vértices sean cúspides, una forma de hacerlo es orientar el lado siguiente en sentido contrario al establecido por la cadena encerrada por *a* y *b*, con esto se gana que los vértices consecutivos no cumplan la condición de estar ambos arriba o ambos abajo. Si el resto de la cadena se mantiene en este lado como en la Figura 1.9 (a), entonces ya sea el vértice más bajo o el más alto, se encuentra en la subcadena delimitada por *a* y *b*, lo cual contradice el hecho de que los puntos más alto y más bajo del polígono deben ser los extremos de cada una de las cadenas y no estar en su interior.

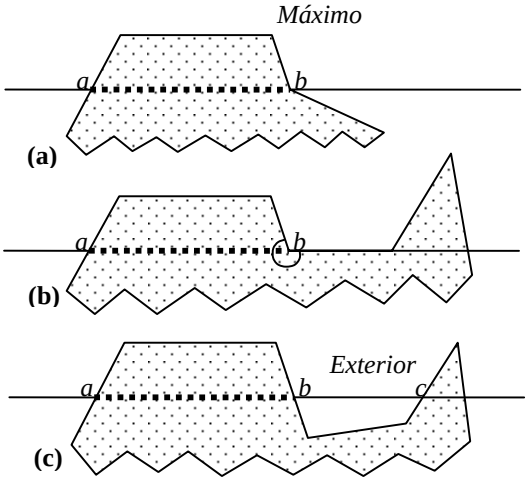


Figura 1.9. *ab* está en el interior de *P*.

Si se sigue la cadena y esta pasa al otro lado de la perpendicular como en la Figura 1.9 (c), entonces la corta y se cuenta ahora con un nuevo componente, donde el segmento de la perpendicular comprendido entre este y el componente anterior se encuentra en el exterior del polígono lo cual lleva al primer caso.

Por otra parte, si el lado siguiente al componente intersecado o el mismo componente en el caso que sea un lado, se orienta con el mismo sentido al de la subcadena, los vértices adyacentes ya sea de *a* (o de *b*), estarán ambos encima o ambos debajo de *a* (o de *b*) lo cual convierte a *a* (o a *b*) en una cúspide interior. Vea la Figura 1.9 (b).

- Si los componentes son puntos interiores de algunos de los lados del polígono, a cada uno de los vértices de dichos lados no les queda más remedio que estar en sentido contrario al anterior y se presenta nuevamente el caso de la contradicción de la hipótesis establecida para los extremos ilustrada en la Figura 1.9 (a), o en el caso que hayan mas de dos componentes se pasaría nuevamente al primer caso. Vea la Figura 1.9 (c).

De esta manera, se ha demostrado que si un polígono no es monótono, entonces tiene cúspides interiores, lo cual es la afirmación contra recíproca y lógicamente equivalente al enunciado inicial.

□

1.5. TRIANGULACIÓN DE POLÍGONOS MONÓTONOS.

Ahora están sentados los precedentes para la triangulación de polígonos monótonos en tiempo computacional lineal.

Algoritmo 1.1.**Triangulación de Polígonos Monótonos.**

1. Organice los vértices por la coordenada *y* de mayor a menor (en tiempo lineal).
2. Construya una *cadena cóncava* con los dos primeros vértices.
3. Sea *v* el tercer vértice.
4. **Mientras que**: *v* no sea el último.
   Haga:
   Caso 1: *v* está en la cadena opuesta a la *cadena cóncava*.
   - Trace la diagonal de *v* al segundo vértice de la *cadena cóncava* y remueva la *cima* de la *cadena cóncava*.
   - Si la *cadena cóncava* tiene un solo elemento, agréguele a *v* y avance *v*.

   Caso 2: *v* es adyacente a la base de la *cadena cóncava*.
   Caso 2a: v^+ es estrictamente convexo.
   - Trace la diagonal de *v* al penúltimo vértice de la cadena y remueva la *base* de la *cadena cóncava* y avance *v.*
   - Si la cadena tiene un solo elemento, agréguele *v* y avance *v*.

   Caso 2b: v^+ es cóncavo o plano.
   - Agréguele *v* a la cadena y avance *v*.

Ejemplo 1.3.
En la Figura 1.10 se puede ver la triangulación de un polígono monótono utilizando el Algoritmo 1.1. Para empezar, el polígono ha de ser dividido en dos cadenas monótonas a saber:
Cad 1: *7-8-9-10-11-12-13-14-0*.
Cad 2: *7-6-5-4-3-2-1-0*.

Note que los vértices del polígono con mayor y menor componente vertical pertenecen a ambas cadenas.

En este caso, los vértices que se seleccionan para comenzar la cadena son el *7* y el *6* y como el siguiente vértice en altura es el *8*, la diagonal puede ser libremente trazada por el hecho de que el vértice *8* está en la otra cadena y los únicos vértices sobre él son los incluidos en la *cadena cóncava*. Ahora se elimina la cima, que en este caso corresponde al *7* y que pasa a ser el *6*, y la cadena queda *6-8*.

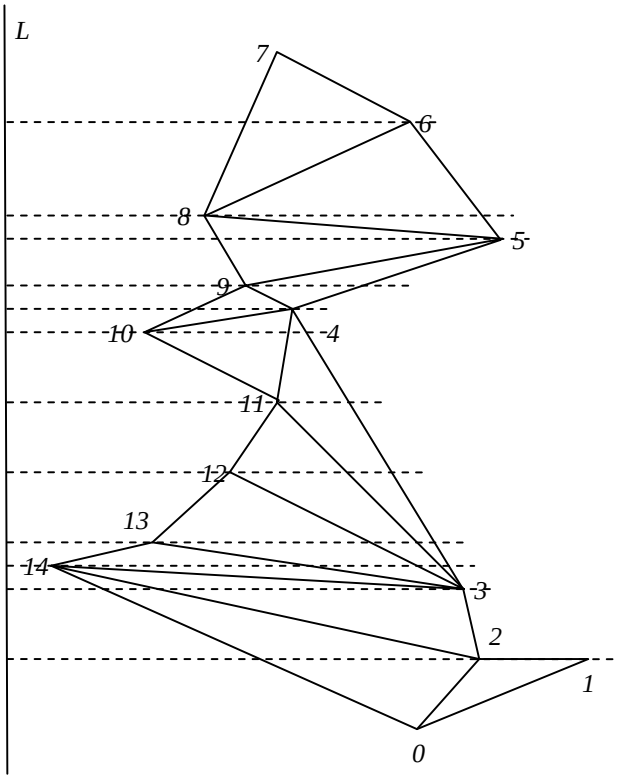


Figura 1.10. Triangulación de Polígonos Monótonos

Se procede de la misma manera hasta tener la cadena *4-11* para la cual solo se le agrega *12-13-14* sin eliminar a nadie, por ser *11, 12* y *13* cóncavos respectivamente. En este momento se procede a trazar diagonales y eliminar vértices pues el siguiente vértice que es el *3* se encuentra en la otra cadena y garantiza la visibilidad con el *11* por la existencia del segmento de *3* a *4* y el hecho de que el orden de alturas es *4-11-3* de esta manera se continúa hasta que la cadena llega a ser *14-3*, donde se toma como siguiente vértice al *2* y no al *1* por estar el *2* conectado con el *3* por medio de un lado o una diagonal. El procedimiento continúa de la misma manera hasta llegar al vértice más bajo que en este caso es el *0*.

Es natural que se pregunte si siempre es posible construir la *cadena cóncava* mencionada en el algoritmo anterior y si las diagonales que se van a trazar son siempre válidas, esto se puede lograr siempre que en cada iteración del algoritmo, el nuevo vértice esté en la otra cadena o si está en la misma, el vértice de la base de la *lista*[10] sea cóncavo. El siguiente lema lo garantiza:

10 En algunas ocasiones como esta, se le llamará *lista* a la *cadena cóncava* a fin de no redundar en su mención.

Lema 1.5.
Los vértices encima de *v* en el comienzo de una nueva iteración en el algoritmo de triangulación monótona:

1. Están todos en una misma cadena (aunque el vértice mas alto es miembro de ambas) y
2. La cadena es cóncava.

Demostración:
Suponga primero que no todos los vértices están sobre una misma cadena, con tal de contradecir la primera afirmación del lema. Sea *a* aquel que está en la cadena contraria de tal manera que *b*, el vértice que le sigue a *a* en altura, pertenece a la cadena inicial. Si *b* es el tope de la lista, entonces tanto *a* como *b* están en la misma cadena ya que el tope pertenece a ambas cadenas. De esta manera, *b* no es el tope de la cadena por lo que el análisis se ubica en el caso 1 del Algoritmo 1.1 así que *ab* es una diagonal del polígono y debería haber sido trazada en una iteración anterior donde se viera por primera vez que *a* pasaba a la cadena contraria. Así, en cada iteración todos los vértices deben estar sobre una misma cadena.
Ahora suponga que existe un vértice en la lista diferente del primero o el último, cuyo ángulo es convexo. De esta manera, dicho vértice debe tener uno que le sigue en menor altura y que pertenece a la lista conforme a la construcción de la misma, y esto ubica el análisis en el *caso 2a* del algoritmo, donde este vértice debería ser la base de la lista y debería haber sido eliminado al trazar la diagonal correspondiente a los vértices que le siguen en mayor y menor altura. Así, la cadena debe ser cóncava

□

Como ya se posee una herramienta para triangular polígonos monótonos en tiempo lineal, solo falta una que permita partir cualquier polígono en pedazos monótonos también en tiempo lineal. Antes de esto, hay otro tipo de particiones, no tan "económico" (computacionalmente) pero muy efectivo y que es además la base para la construcción del algoritmo para la triangulación de polígonos en tiempo lineal.

1.6. PARTICIONES MONÓTONAS.

1.6.1. Trapezoidalización.

Definición 1.13.
P es un trapezoide, si y solo si *P* es un polígono de *4* lados dos de ellos paralelos.

Nota: Los triángulos también se tomarán como trapezoides pero degenerados, relacionando a cualquiera de los lados con su vértice opuesto correspondiente como si este fuera un lado de longitud *0*.

Como su nombre lo indica, una trapezoidalización consiste en dividir un polígono en trapezoides, lo cual se logra trazando líneas paralelas que crucen por cada uno de sus vértices y permanezcan internas al polígono. Las figuras delimitadas por estos segmentos conforman los trapezoides de la partición.

Aquí se hará referencia a trapezoidalizaciones horizontales siguiendo con la rutina de concentrarse en las coordenadas verticales de cada vértice.
Los vértices sobre los cuales queda formado cada trapezoide son llamados *vértices de apoyo*.

La conexión importante entre la trapezoidalización y los polígonos monótonos es que si un vértice de apoyo está en el interior de la horizontal trazada, este necesariamente ¡es una cúspide interior!

Ahora simplemente se eliminan las cúspides interiores trazando segmentos que unan los vértices de apoyo de los trapezoides que involucren cúspides interiores. Como cada vértice de apoyo pertenece a dos trapezoides (menos los vértices de los extremos), se unirá con el vértice de apoyo inmediatamente superior si la cúspide es ascendente[11] e inmediatamente inferior si la cúspide es descendente[12].

Ejemplo 1.4.

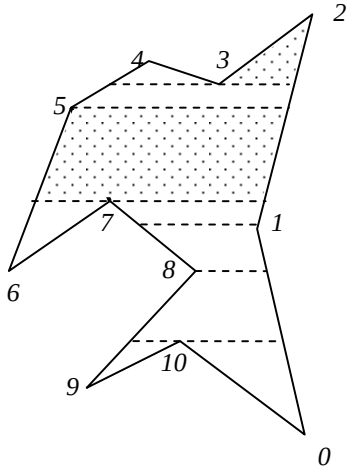


Figura 1.11. Trapezoidalización.

En la Figura 1.11 se ilustra la forma como se divide un polígono cualquiera en trapezoides (líneas punteadas). La sección del polígono delimitada por las horizontales que pasan por *5* y por *7* claramente forma un trapezoide, pero la delimitada por el vértice *2* y la horizontal que pasa por *3* es uno de esos casos degenerados que se mencionaban anteriormente, donde el vértice *2* es el lado degenerado de longitud *0*.

Ahora, ¿cómo trazar estas horizontales?
La técnica utilizada para esto es conocida como *Barrido Del Plano* y consiste en obtener los lados que intersecan a la horizontal correspondiente a cada vértice lo cual se logra en un tiempo de *O(log n)* por cada vértice para un tiempo total de *O(nlog n)* para el polígono completo.

1.6.1.1. Barrido del Plano.

Básicamente lo que plantea esta técnica es que dado un vértice *v*, calcule los lados a los cuales interseca la recta horizontal *L'* que pasa por *v*, que son aquellos cuya componente vertical de sus extremos esta una por encima y otra por debajo de *L'*, para luego encontrar los dos puntos de intersección cuya componente horizontal está más cerca a la de *v*. Una

11 $v \geq v^+$ y $v \geq v^-$.
12 $v \leq v^+$ y $v \leq v^-$.

vez obtenidos estos dos puntos, se desea saber si el segmento a lado y lado de *v* está en el interior del polígono, para lo cual los lados deben ser organizados en una estructura que es almacenada hasta el final cuando al cortar el primer lado, se entiende que se está en el interior, al cortar el segundo en el exterior y así sucesivamente hasta llegar a los dos lados requeridos.

Suponiendo que todos los vértices están a diferente altura, en la situación anterior se contemplan dos casos para la ubicación del vértice los cuales dan la pauta para la organización de los lados:

Sea *v* un vértice de cierto polígono, tal que los lados que lo comparten son *a* y *b* y los dos lados siguientes en distancia horizontal a *v* son *c* y *d* a izquierda y derecha respectivamente. Sea *L'* una recta horizontal que pasa por *v*.

Caso 1:
a está por encima de *L'* y *b* por debajo.
En este caso se conserva el arreglo que se lleva de lados del polígono eliminando al lado que está sobre la recta, es decir *a*.

Caso 2:
Tanto *a* como *b* están por encima o por debajo de *L'*.
En este caso se conserva el arreglo que se lleva de lados del polígono eliminando tanto a *a* como a *b*.

Ejemplo 1.5.
La Figura 1.12 muestra como después de la trapezoidalización del polígono es posible dividirlo en piezas monótonas "cancelando" sus cúspides interiores al dejar los vértices adyacentes al vértice cúspide, uno por debajo y otro sobre la horizontal, ya que por el Lema 1.4 si un polígono carece de cúspides entonces es monótono. Los detalles de la organización de los lados en el barrido del plano quedan explícitos en la tabla 1.1.

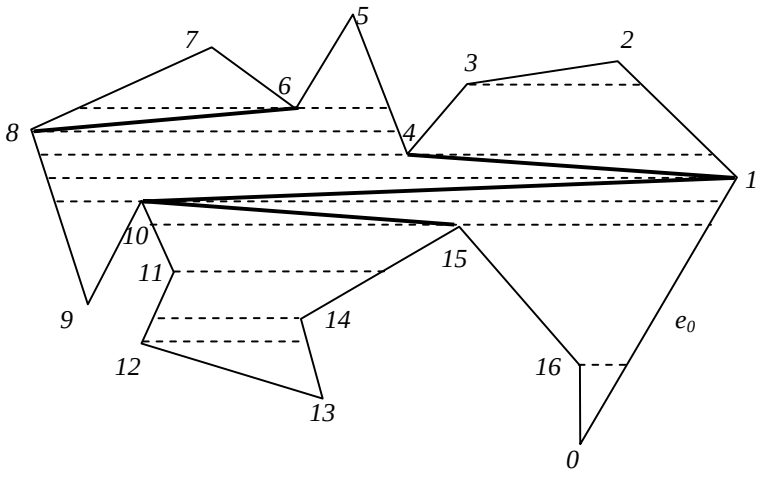


Figura 1.12. Partición de un polígono en piezas monótonas.

La partición anterior se convierte entonces en el medio por el cual se consigue que un polígono arbitrario quede dividido en subpolígonos monótonos, los cuales pueden ser triangulados por el Algoritmo 1.1.

Vértice	Lados intersecados	Lados más cercanos	Observaciones
5			Se eliminaron los dos lados adyacentes al vértice.
7	*e5, e4*	*e5*	Se eliminaron los dos lados adyacentes al vértice.
2	*e7, e6, e5, e4*	*e4*	Se eliminaron los dos lados adyacentes al vértice.
3	*e7, e6, e5, e4, e3, e1*	*e4, e3, e1*	Se eliminó el lado *e2*.
6	*e7, e4, e3, e1*	*e7, e4*	Se eliminaron los dos lados adyacentes al vértice.
8	*e8, e4, e3, e1*	*e8, e4*	Se eliminó el lado *e7*.
4	*e8, e1*	*e8, e1*	Se eliminaron los dos lados adyacentes al vértice.
1	*e8, e0*	*e8, e0*	Se eliminó el lado *e1*.
10	*e8, e0*	*e8, e0*	Se eliminaron los dos lados adyacentes al vértice.
15	*e8, e9, e10, e0*	*e10, e0*	Se eliminaron los dos lados adyacentes al vértice.
11	*e8, e9, e11, e14, e15, e0*	*e9, e11, e14*	Se eliminó el lado *e10*.
9	*e11, e14, e15, e0*	*e11*	Se eliminaron los dos lados adyacentes al vértice.
14	*e11, e13*	*e11, e13, e15*	Se eliminó el lado *e14*.
12	*e16, e13, e15, e0*	*e12, e13*	Se eliminó el lado *e11*.
16	*e16, e13, e16, e0*	*e13, e16, e0*	Se eliminó el lado *e15*.
13	*e16, e0*	*e16*	Se eliminaron los dos lados adyacentes al vértice.
0			Se eliminaron los dos lados adyacentes al vértice.

Tabla 1.1. Barrido del Plano.

El algoritmo resultante para la triangulación es el siguiente:

Algoritmo 1.2. **Triangulación de polígonos.**
1. Organice los vértices por la coordenada vertical.
2. Ejecute el barrido del plano para construir la trapezoidalización.
3. Divida en polígonos monótonos eliminando las cúspides interiores (como se explicó en el Ejemplo 1.5).
4. Triangule cada polígono monótono usando el Algoritmo 1.1.

Ejemplo 1.6.
Observe que la horizontal solo se traza en ambos costados de cada vértice hasta encontrar los lados más cercanos.
Habría sido ideal que el orden fuera lineal, por ello es necesaria la partición en partes convexas que a su vez son monótonas.

Antes de la partición lineal, Tarjan y VanWik (1988) obtuvieron algoritmos de *O(n loglog n)*[13] lo cual produjo gran actividad en la comunidad científica apareciendo algoritmos diversos entre los que se destacan algunos de *O(n log*n)*[14], donde *log*n* se refiere al número de veces que se debe aplicar el logaritmo hasta obtener un valor menor o igual a *1*.

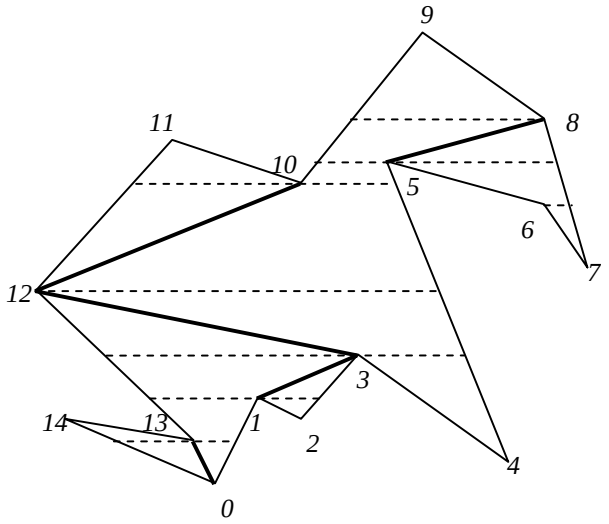


Figura 1.13. Trapezoidalización cancelando solo cúspides.

Ejemplo 1.7.
*Log*65536 = 4*, ya que:

1. *Log 65536 = 16 > 1*
2. *Log 16 = 4 > 1*
3. *Log 4 = 2 > 1*
4. *Log 2 = 1 ≤ 1*

Chazelle (1991)[15] mejora todos los algoritmos anteriores obteniendo un *O(n)*.
Básicamente lo que hace es subdividir el polígono en cadenas, tomando exclusivamente grupos de vértices que permitan ser organizados en un tiempo lineal para después mezclarlos en un tiempo también lineal.
Además, Chazelle acota el número de partes convexas en función del número de vértices cóncavos apoyándose en particiones por segmentos y por diagonales.
En las *particiones por segmentos*, toma los vértices cóncavos y los biseca y en las *particiones por diagonales* simplemente los une.
Aquí, no se expondrá dicho algoritmo en detalle y quedará solamente el algoritmo detallado anteriormente con un orden de *nlog n*, que es mayor que el lineal, pero aceptable. Así se da paso a una de las estructuras más importantes de la Geometría computacional por cuanto acota el análisis geométrico de diversas figuras y que es el tema del siguiente capítulo.

13 Observe que *nloglog n < nlog n*.
14 Observe que *nlog* n << nlog n* para *n* suficientemente grande.
15 CHAZELLE, B., Triangulating a simple polygon in Linear Time, 1991.

Capítulo 2

Envolvente Convexa

2.1. INTRODUCCIÓN.

Imagine que se desea encontrar la menor longitud para un tubo, de manera que una hilera de personas -no equidistantes- pueda sujetarse de él. De la misma forma, piense en cuál es el menor tamaño y forma que debería tener una galería de arte para que un solo guardia pueda vigilar todas las pinturas y esculturas en ella. Piense también, en la forma que debería tener lo visualizado por un radar acuático para ilustrar una bancada de peces en un modelo de 3D. Todos estos interrogantes podrían ser comprendidos desde un concepto que más adelante se definirá formalmente que es la *envolvente convexa*.
Los problemas anteriores pueden ser clasificados según el espacio en el cual se desenvuelven como lo son $\boldsymbol{R}$, $\boldsymbol{R^2}$ y $\boldsymbol{R^3}$ respectivamente y de la misma forma clasificar la envolvente correspondiente a cada uno de ellos.
Algunos investigadores destacados han avanzado en algoritmos computacionales que consiguen dicha envolvente, los cuales se verán en el presente capítulo.
La convexidad es una propiedad de los objetos que facilita la manipulación de los mismos y por ende la profundización en su conocimiento. Este hecho resalta la importancia de la envolvente convexa y motiva su análisis.

2.2. DEFINICIÓN DE CONCEPTOS.

Para poder relacionarse con el concepto de *Envolvente Convexa*, primero es necesario hacer una definición formal de convexidad que servirá de apoyo en el Lema 2.1 para demostrar que la envolvente cumple con dicha propiedad.

Sea S un subconjunto de $\boldsymbol{R^n}$.

Definición 2.1. **Convexidad.**
S es convexo si y solo si para todo par de puntos x, y de S, se tiene que el segmento xy está contenido en S.

Con esta definición queda claro que cualquier región en el plano con una abolladura es no convexa y que en particular, un polígono con un vértice cóncavo es no convexo.

Proposición 2.1.
Todo polígono con uno o más vértices cóncavos es no convexo.

Demostración:
Sea *P* un polígono, *v* un vértice cóncavo de dicho polígono y *a* y *b* los vértices anterior y siguiente de *v*. Vea la Figura 2.1.
Como los vértices están en la frontera del polígono, existe una bola B_ε (*a*) dividida en dos sectores circulares para los cuales uno está en el interior y otro en el exterior del polígono. Los puntos en el interior del segmento *ab* que están en el interior de la bola pertenecen al sector exterior, pues de no ser así el vértice *v* sería convexo, es decir que para los puntos *a* y *b* del polígono el segmento *ab* no está completamente contenido en el polígono y por ello *P* no es convexo.

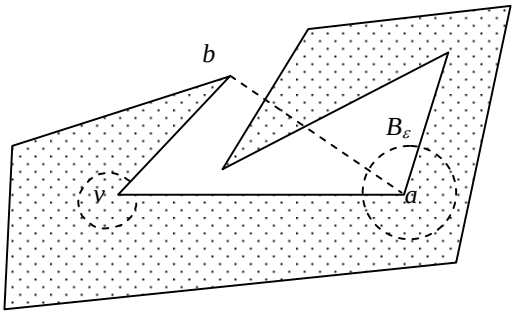


Figura 2.1. Convexidad de un Polígono.

□

Para la construcción de una definición para la envolvente es útil conocer las combinaciones convexas de un conjunto de puntos, siendo el segmento la combinación a tratar con menor dimensión.

Definición 2.2.
El segmento *xy* es el conjunto de todos los puntos de la forma $\alpha x + \beta y$ con $\alpha \geq 0$, $\beta \geq 0$ y $\alpha + \beta = 1$.

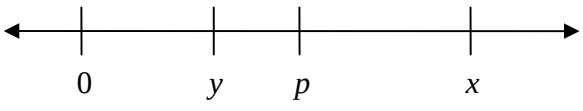


Figura 2.2. Visión algebraica de un segmento.

Cualquier real *p* en el segmento *yx* es igual a la suma del extremo menor del segmento, en este caso *y*, con un porcentaje del mismo, el cual es un número real menor que *1*, asi:

$$
\begin{aligned}
p &= y + \alpha\,(yx) \\
&= y + \alpha\, \mathbf{d}(y, x)^{16} \\
&= y + \alpha\,|y - x| \\
&= y + \alpha\,(x - y) \\
&= y + \alpha x - \alpha y \\
&= \alpha x + y - \alpha y \\
&= \alpha x + (1 - \alpha) y \\
&= \alpha x + \beta y, \text{ donde } \beta = (1 - \alpha), \text{ es decir que } \alpha + \beta = 1
\end{aligned}
$$

16 ***d (y, x)*** es la distancia de *x* a *y* bajo la métrica usual en ***R***

Definición 2.3.
Una *combinación convexa* de los puntos $x_1,..., x_k$ es una suma de la forma $\alpha_1 x_1 +...+ \alpha_k x_k$, con $\alpha_i \geq 0$ para $i \in [1: k]$ y $\alpha_1 +...+ \alpha_k = 1$.

Nota: De esta manera, un segmento consiste de todas las combinaciones convexas de sus puntos extremos y (generalizando) todo polígono convexo consiste de todas las combinaciones convexas de sus vértices.

A partir de las definiciones anteriores es posible formalizar el concepto de envolvente convexa con las siguientes definiciones:

Definición 2.4.
La envolvente convexa de *S* es el conjunto de todas las combinaciones convexas de los puntos de *S*.

Notación:
La envolvente convexa se denota comúnmente con la abreviación *Conv(S)* o con el símbolo $\mathcal{H}(S)$.[17]

Ejemplo 2.1.

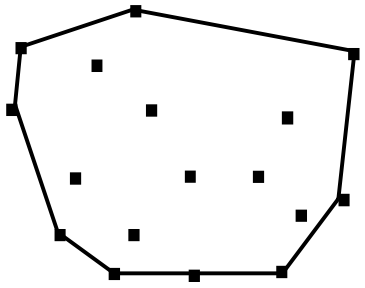


Figura 2.3. Envolvente convexa de una nube de puntos en **R2**.

La Figura 2.3 muestra la envolvente convexa de un conjunto de puntos en el plano. La cual como se puede observar, está formada por la menor región poligonal que contiene al conjunto.

Note que lo que está remarcado en la figura mencionada no es la envolvente como tal sino su frontera.

Lema 2.1.
La envolvente de un conjunto de puntos es un conjunto convexo.

Demostración:
Sean $a, b \in \mathcal{H}(S)$ dos puntos arbitrarios.
El segmento *ab* es por definición el conjunto de puntos de la forma $\alpha a + \beta b$, $\alpha \geq 0$ con $\beta \geq 0$ y $\alpha + \beta = 1$, pero como cada una de estas combinaciones puede expresarse como

$$\alpha a + \beta b + \sum_{x} (0 * x), \text{ donde } x \in S - \{a, b\}$$

que es una combinación convexa de los puntos de *S* con todos los escalares no negativos y donde la suma de los mismos es *1*, entonces $ab \subseteq \mathcal{H}(S)$, i.e., $\mathcal{H}(S)$ es convexo.

□

17 La $\mathcal{H}$ se debe a la palabra en inglés "*hull*" que significa *cascarón o envolvente*.

Definición 2.5.
$\mathcal{H}(S)$ en d dimensiones es el conjunto de todas las combinaciones convexas de $d + 1$ (o menos) puntos de S.

Esta definición tiene la ventaja que solo necesita $d + 1$ puntos para construir las combinaciones. Por ejemplo, para conjuntos en el plano solo basta construir combinaciones de tres puntos, es decir de todos los triángulos formados por los puntos del conjunto.

Definición 2.6.
$\mathcal{H}(S)$ es la intersección de todos los conjuntos convexos que contienen a S.

$$\mathcal{H}(S) = \cap K, \text{ donde } S \subseteq K \text{ y } K \text{ es convexo.}$$

Nota: Esta definición no depende de la noción de combinación convexa, pero por otra parte la noción de "todo conjunto convexo" no es clara.

Definición 2.7.
$\mathcal{H}(S)$ es la intersección de todos los semiespacios que contienen a S.

$$\mathcal{H}(S) = \cap \pi, \text{ donde } \pi \text{ es una semirecta, un semiplano, un semiespacio,...}$$

Nota: Dado un punto p en $\boldsymbol{R}$, un semiespacio en $\boldsymbol{R}$ es un subconjunto conformado por todos los puntos que pertenecen a $\boldsymbol{R}$, tales que para cualquier par de dichos puntos, el segmento que los une no contiene a p (semirecta). Dada una recta l en $\boldsymbol{R^2}$, un semiespacio en $\boldsymbol{R^2}$ es un subconjunto conformado por todas las parejas ordenadas tales que para cualquier par de ellas, el segmento que las une no corta a la recta l (semiplano). Dado un plano π en $\boldsymbol{R^3}$, un semiespacio en $\boldsymbol{R^3}$ es un subconjunto conformado por todas las triplas tales que para cualquier par de ellas, el segmento que las une no corta al plano π (semiespacio). De aquí en adelante se les conoce como *semiespacios*.

Definición 2.8.
$\mathcal{H}(S)$ en el plano es el polígono convexo P más pequeño[18] que encierra a S.

Definición 2.9.
$\mathcal{H}(S)$ en el plano es la unión de todos los triángulos determinados por puntos en S.

$$\mathcal{H}(S) = \cup T, \text{ donde } T \text{ es un triángulo con vértices en } S.$$

A diferencia de las anteriores, esta última definición sugiere un algoritmo. Básicamente propone que para calcular la envolvente de un conjunto solo basta con hacer la unión de todos los triángulos que se puedan conseguir tomando vértices en S.

18 El más pequeño en el sentido que no hay otro P' tal que $P \supset P' \supseteq S$.

2.3. ALGORITMOS BIDIMENSIONALES.

2.3.1. Delimitación de los algoritmos.

De aquí en adelante se hará referencia a la envolvente como a la frontera de la misma, a menos que se especifique lo contrario.

En este punto surgen preguntas como ¿qué salidas se desean para los algoritmos?, ¿qué constituye construir la frontera de la envolvente?
Los algoritmos que se presentarán son para conjuntos finitos de puntos en dos dimensiones.

De la misma forma que en el Capítulo 1 se pretendía conseguir un algoritmo que dividiera un polígono en la menor cantidad de piezas y en el menor tiempo posible, en este capítulo se quiere conseguir uno que encuentre la envolvente convexa de un conjunto de puntos en $\boldsymbol{R^2}$ en el menor tiempo posible.

Los algoritmos irán evolucionando mejorando cada vez su orden, pero a diferencia del capítulo anterior, aquí solo se llegará a un *O(nlogn).*

Para contestar la primera pregunta planteada, se definirá la salida del algoritmo como los puntos extremos, los cuales corresponden a los vértices de la envolvente. No se considerarán los puntos en el interior de un segmento de la envolvente como extremos, ya que esto conllevaría a realizar cálculos innecesarios.

2.3.2. Puntos Interiores.

Definición 2.10.
Un punto $l \in S$ es *interior*, si y solo si existen tres puntos i, j, k en S tales que los determinantes

$$\begin{vmatrix} i_x & i_y & 1 \\ j_x & j_y & 1 \\ l_x & l_y & 1 \end{vmatrix}, \begin{vmatrix} j_x & j_y & 1 \\ k_x & k_y & 1 \\ l_x & l_y & 1 \end{vmatrix} \text{ y } \begin{vmatrix} k_x & k_y & 1 \\ i_x & i_y & 1 \\ l_x & l_y & 1 \end{vmatrix}$$

son siempre positivos.

Inicialmente se pensó en diseñar un algoritmo que retornara primero los puntos *extremos*, pero debido a que su definición no sugiere ningún procedimiento analítico, se definen los *puntos interiores* de la envolvente, ya que después de tenerlos será muy sencillo saber cuales son los puntos extremos (los restantes).

Algoritmo 2.1. **Puntos Interiores (Interior Points).**
Sean $S \subset \boldsymbol{R}^2$ y $i, j, k, l \in [\, 0 : n \,]$, donde $n = |\, S \,|$
Para cada i , **haga**:
Para cada $j \neq i$, **haga**:
Para cada $k \neq i \neq j$, **haga**:
Para cada $l \neq i \neq j \neq k$, **haga**:
Si p_l está a la izquierda o sobre (p_i,p_j),
p_l está a la izquierda o sobre (p_j,p_k) y
p_l está a la izquierda o sobre (p_k,p_i),
Entonces
p_l es no extremo.

Orden:
Como hay cuatro secuencias anidadas, cada una de $O(n)$, el algoritmo completo es de $O(n^4)$. En otras palabras, la prueba de los extremos "cuesta" n por cada uno de los n^3 triángulos.

Nota: La Definición 2.10 y el Algoritmo 2.1 establecen implícitamente que un punto a "está a la izquierda" del segmento dirigido bc, si y solo si la suma de las componentes de la resultante del producto vectorial $ab \times ac$ es mayor o igual que cero.

Ejemplo 2.2.

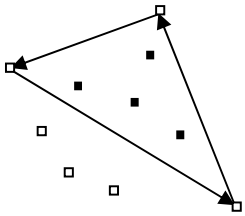


Figura 2.4. Puntos Interiores.

■ : Son puntos interiores.

Note que en la Figura 2.2. los lados del triángulo son líneas dirigidas con el fin de que tenga sentido mencionar que se está a la izquierda de una de ellas. También se garantiza que los puntos interiores no son vértices del triángulo al descartarlos en la secuencia de l (En la última secuencia $l \neq i \neq j \neq k$).

Teniendo en cuenta que un punto es extremo si y solo si existe una recta cuya intersección con el conjunto es igual a dicho punto, el siguiente lema garantiza la ejecución del algoritmo anterior.

Lema 2.2.
Un punto no es extremo si y solo si está en el interior de algún triángulo (cerrado), cuyos vértices son puntos del conjunto.

Demostración:
La base de este lema es simplemente la Definición 2.9
Sea p un punto que está en el interior de un triángulo formado por puntos de S.
Como $\mathcal{H}(S) = \cup\, T$, donde T es un triángulo con vértices en S, entonces p está en el interior de $\mathcal{H}(S)$. Así, existe una bola abierta $B_\varepsilon(\, p) \subset \mathcal{H}(S)$ y de esta manera, cualquier segmento que contenga a p también contiene infinitos puntos de $\mathcal{H}(S)$ -los que están en la bola-. Luego p no es un punto extremo. □

Nota: Para comprobar que si un punto es extremo o no, es necesario que este no sea un vértice del triángulo con el cual se verificará su contenencia, ya que en este caso el punto de análisis no estaría en el interior del triángulo.

2.3.3. Lados Extremos

EL siguiente algoritmo mejora el orden de ejecución a un $O(n^3)$ calculando los lados de la envolvente en vez de calcular sus vértices.

Algoritmo 2.2. **Lados extremos (Extreme Edges).**

Sea S un conjunto de puntos en $\boldsymbol{R}^2 i \in [\ 1 : n\ ]$, donde $n = |\ S\ |$
Para cada i , **haga**:
  Para cada $j \neq i$, **haga**:
    Para cada $k \neq i \neq j$, **haga**:
      Si
        p_k no está a la izquierda o sobre (p_i,p_j),
        Entonces:
          (p_i,p_j) es no extremo.

Orden:
En esta ocasión hay tres secuencias anidadas cada una de $O(n)$, por lo cual el orden total es $O(n^3)$. Es decir que por cada uno de los n^2 pares de puntos, debe comprobarse si cada uno de los $n-2$ puntos (lo cual es de $O(n)$) está o no a la izquierda del segmento formado por cada par de puntos.

Ejemplo 2.3.

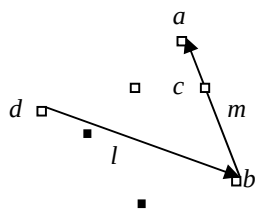


Figura 2.5. Lados Extremos.

■ : Está a la derecha de la línea dirigida l por lo cual dicha línea **no** es un lado extremo.

En la Figura 2.5, m es un lado extremo, pero en sentido contrario no lo es, por lo cual solo será tomado como lado extremo en la iteración donde satisfaga las condiciones del algoritmo.

Observe que tanto ac como bc podrían ser lados extremos pero no se tomarán así; el punto c queda descartado por no estar estrictamente a la izquierda de m. Con esta restricción, el Algoritmo 2.1 solo será utilizado para conjuntos de puntos no colineales.

2.3.4. Envoltura de Regalo.

A continuación se presenta un algoritmo que acelera el orden de ejecución del algoritmo 2.2 a un $O(n^2)$ utilizando una técnica muy natural que consiste en ir envolviendo la nube de puntos empezando por el primer lado extremo (dirigido) encontrado en el algoritmo 2.2 y cubriendo con segmentos de recta angularmente y en sentido antihorario, hasta retornar al punto final del lado dirigido ya mencionado.

Algoritmo 2.3. **Envoltura de regalo (Gift Wrapping).**

Encuentre el punto más bajo (la coordenada *y* más pequeña)
Sea i_0 su índice.
Haga $i = i_0$
Repita
Para cada $j \neq i$ **haga**:
Calcule el ángulo antihorario θ que forma cada segmento i_0p, donde i_0 es el vértice final del lado envolvente encontrado en la primera iteración del Algoritmo 2.2. y *p* es cada uno de los demás puntos del polígono.
Sea *k* el índice del punto con más pequeño ángulo θ.
Retorne (p_i, p_k) como un lado de la envolvente.
Haga $i = k$.
Hasta $i = i_0$.

Orden:
Este algoritmo es de $O(n^2)$ ya que ejecuta un $O(n)$ por coda lado de la envolvente.

Ejemplo 2.4.
—— : Lado encontrado por el Algoritmo 2.2.

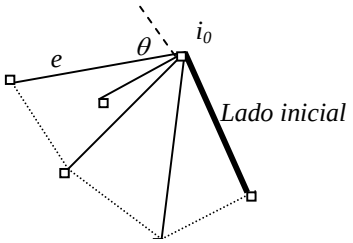


Figura 2.6. Envoltura de Regalo.

En la Figura 2.6, *e* es el lado con menor ángulo θ con respecto al lado encontrado.

La clave para que este algoritmo disminuya el tiempo de ejecución es el trabajo por ángulos ya que si para *e* se revisara si no hay puntos estrictamente a la derecha, se regresaría al $O(n^3)$ del algoritmo anterior.

2.3.5. Envolvente Rápida.

El siguiente algoritmo logra un orden de ejecución de $O(n \log n)$ llegando casi al prometido al comienzo de esta sección. Casi porque solo se consigue este orden en el mejor de los casos, mientras que en el peor se mantiene el $O(n^2)$ del algoritmo anterior.

La idea es tomar los cuatro puntos cardinales para formar un cuadrilátero que encierra cierta cantidad de puntos en la nube. La totalidad de los puntos quedan encerrados en el rectángulo formado por los dos segmentos horizontales que contienen respectivamente el punto más alto y más bajo y por los dos segmentos verticales que contienen los puntos más a la derecha y mas a la izquierda respectivamente. Este rectángulo puede tomarse como una primera aproximación de la envolvente, la cual es válida para ciertos objetivos específicos.

Al tomar simultáneamente al rectángulo y al cuadrilátero, se tiene una subdivisión de la región contenedora de la nube de puntos que consta del cuadrilátero anteriormente mencionado y cuatro triángulos rectángulos ubicados uno a cada lado del cuadrilátero.

Tomando a uno de estos triángulos se puede formar uno nuevo con los vértices del lado que coincide con el cuadrilátero y el punto en el interior del triángulo rectángulo que esté más lejos de dicho segmento.

De la misma forma se puede formar un nuevo triángulo sobre cada uno de los lados adyacentes al nuevo vértice y repetir el procedimiento hasta cubrir la nube completa.
Esto quedará más claro después de observar el Algoritmo 2.4 y el Ejemplo 2.4.

Algoritmo 2.4. **Envolvente rápida (QuickHull).**

Defina la función *QuickHull(a,b,S)*
Si
S={ *a, b* }
Entonces
Retorne (*a, b*).
Si no
Haga *c* = Índice del punto con distancia máxima a *ab*.
A= Puntos a la derecha de (*a, c*).
B= Puntos a la derecha de (*c, b*).
Retorne *QuickHull(a, c, A)* concadenado con *QuickHull(c, b, B).*

Orden:
Sea *T(n)* la complejidad de tiempo de la función *QuickHull(S)* con $|S| = n$.
Sea $|A| = \alpha$ y $|B| = \beta$.
Entonces $T(n) = O(n) + T(\alpha) + T(\beta)$, es decir, el tiempo que tarda en encontrar el punto más lejano sumado con el que tarde en aplicar *QuickHull(S)* tanto al subconjunto *A* como al subconjunto *B*.

Continuando con el análisis del tiempo de ejecución del algoritmo, se tienen dos casos; en el mejor de ellos, se divide el conjunto *S* en dos subconjuntos con igual cantidad de elementos y en el peor, la mayoría de los elementos de *S* quedan recargados a uno de sus subconjuntos:

En el mejor caso, el algoritmo trabaja con dos divisiones balanceadas de puntos:

$\alpha = \beta = n/2 \Rightarrow$ $T(n) = O(n) + 2T(n/2)$, lo cual es
$T(n) = O(n \log n)$.

En el peor caso, la división es lo más sesgada posible, por lo cual:

$\alpha = 1$ y $\beta = n-1 \Rightarrow$ $T(n) = O(n) + T(n - 1)$
$T(n) = c_0 n + T(n - 1)$
$T(n) = c_0 n + c_1 (n – 1) + T(n - 2)$
$T(n) = ...$
$T(n) = c_0 n + c_1 (n – 1) + c_2 (n – 2) + ... + c_{n-1} = O(n^2)$.

QuickHull es generalmente rápido, sin embargo alcanza un orden cuadrático en su peor caso.

Ejemplo 2.5.

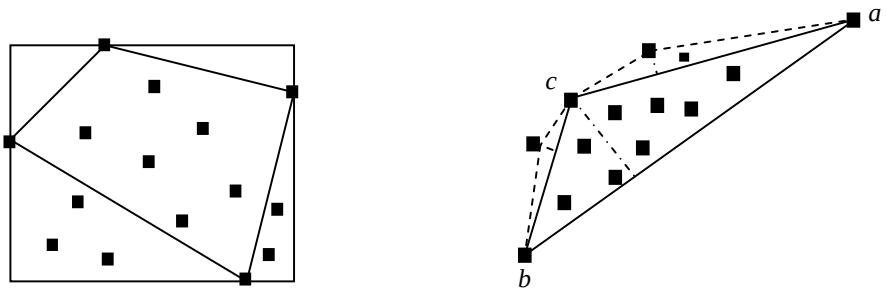


Figura 2.7. QuickHull.

La Figura 2.5 muestra los dos pasos claves del Algoritmo 2.4: El primero, que debe hacerse aunque no esté explícito en el pseudocódigo y que consiste en dividir la región que encierra la nube en cinco regiones y el segundo que aplica la función *QuickHull* a cada una de las cuatro regiones triangulares.

2.3.6. Escaneo de Graham.

Con este algoritmo se llega al final de los algoritmos presentados en este texto para calcular la envolvente convexa de una nube de puntos en dos dimensiones. Primero se presentará una versión intuitiva del proceso sugerido por Graham y luego una versión mejorada de dicho proceso.

La idea del algoritmo es simplemente escoger un punto interior de la nube y organizar el resto angularmente con respecto a él; después, cubrir la nube en sentido antihorario e ir construyendo un arreglo conformado por los puntos que se vayan encontrando, guiándose siempre por la organización angular previamente establecida y teniendo en cuenta que siempre se debe girar a la izquierda (en caso contrario deberá saltarse el último punto). Para esto, deberán definirse previamente dos funciones a saber *push(S, p)* y *pop(p)*, las cuales respectivamente agregan y saltan un punto de la organización angular de la nube en la construcción del arreglo. Al final, los puntos contenidos en el arreglo serán quienes determinen la frontera de la envolvente.

La necesidad de realizar dos versiones del algoritmo se debe a que en la primera se presenta el problema que si el punto inicial del arreglo (después del centro de organización angular) está en el interior de la envolvente, el resultado del algoritmo será la envolvente del conjunto pero con una abolladura en dicho punto. El segundo problema es que en ocasiones se pueden encontrar dos o más puntos en la misma posición angular con respecto al punto inicial, lo cual queda corregido en la segunda versión simplemente organizándolos según la distancia a la que estén del punto inicial empezando por el más cercano.

Algoritmo 2.5. **Escaneo de Graham versión A.**

1. Encuentre un punto interior x. Llámelo p_0.
2. Ordene los otros puntos angularmente y en sentido antihorario en torno a x. Llámelos $p_1,..., p_{n-1}$.
3. Sea el arreglo $S = \{p_{t-1},p_t\} = \{p_1,p_2\}$, donde t indica el tope.
4. **Haga** $i=3$,
5. **Mientras qu**e $i \leq n$, **haga**:
   Si
   p_i está a la izquierda de (p_{t-1},p_t)
   Entonces,
   push(S,i) e incremente i.
   Si no *pop(S)*.
6. **Haga** $i = 1$,
7. **Mientras que** $i = 1$, **haga:**
   Si
   p_i está a la izquierda de (p_{t-1},p_t)
   Entonces,
   push(S,i) e incremente i.
   Si no *pop(S)*.

Orden:
A lo sumo se quita y se pone un punto en cada iteración lo cual equivale a $O(2n) \equiv O(n)$. Sin embargo, la organización angular de los vértices cuesta $O(n \log n)$, por lo que en general, el algoritmo toma un tiempo de $O(n \log n)$.

Ejemplo 2.6.
Las funciones *push(S, p)* y *pop(p)* adicionan y saltan un punto respectivamente en el conteo angular antihorario.

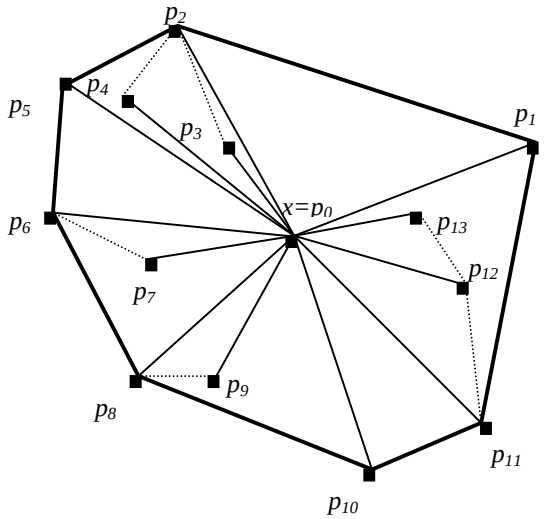


Figura 2.8. Graham versión A.

En la Figura 2.8 se puede observar que cuando se desea pasar a p_4 es necesario girar a la derecha por lo cual p_3 es saltado en el arreglo. Lo mismo le ocurre a cada punto que se encuentre en el interior de la envolvente. Después de haber saltado a p_{12} y a p_{13} se cierra la el arreglo agregando a p_1.

Los problemas ocurren cuando p_1 no pertenece a la frontera de la envolvente y cuando se encuentran puntos colineales. Por ello es necesario perfeccionar el algoritmo para obtener la siguiente versión.

Algoritmo 2.5. **Escaneo de Graham versión B.**

1. Encuentre el punto que esté a la derecha de entre los más bajos y llámelo p_0.
2. Organice los demás puntos angularmente con respecto a p_0.
   Para puntos colineales enumere primero los más cercanos a p_0 y llámelos $p_1, p_2,..., p_{n-1}$.
3. Sea el arreglo $S = \{p_{t-1}, p_t\} = \{p_{n-1}, p_0\}$, donde t indica el tope.
4. **Haga** $i = 1$.
5. **Mientras que** $i < n$, **haga**:
   Si
   p_i está a la izquierda de (p_{t-1},p_t)
   Entonces,
   push(S, i) e incremente *i*.
   Si no *pop(S)*.

Ejemplo 2.7.

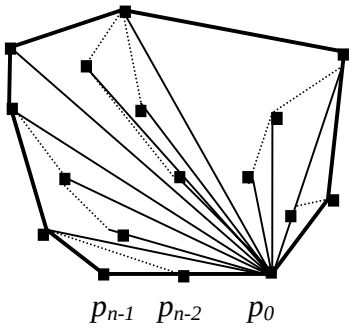


Figura 2.9. Graham versión B.

El problema de garantizar que el primer lado seleccionado pertenezca a la envolvente se soluciona escogiendo como inicio de la colección al punto que esté más a la derecha de entre los puntos con menor componente horizontal.

Si hay puntos colineales, simplemente se organizan según su cercanía al punto de referencia (p_0).

2.4. ENVOLVENTE CONVEXA EN TRES DIMENSIONES.

Si se concentra la atención en el plano, es sencillo comprender las figuras que resultan en el proceso de hallar la envolvente convexa de un conjunto de puntos, los polígonos.
Sin embargo, las figuras que se encuentra al tratar de construir la envolvente en un espacio tridimensional son un poco más complejas. Definir estas figuras y tener claridad en las características que se desean en cada una de ellas, es una tarea que facilita la comprensión de los algoritmos envolventes y deja sentadas las bases para el diseño y construcción de los mismos.

2.4.1. Poliedros.

"Un poliedro es la generalización natural de un polígono bidimensional a tres dimensiones: es una región del espacio cuya frontera comprende un número finito de caras poligonales planas, donde cualquier par de ellas o son disyuntas o cortan en lados y vértices"[19]

19 O'ROURKE ,Joseph. Geometría Computacional en C. p 113.

La frontera de un poliedro comprende tres tipos de objetos geométricos los cuales serán llamados componentes:

1. **Puntos**: Vértices de dimensión 0,
2. **Segmentos**: Lados o aristas de dimensión 1 y
3. **Polígonos**: Caras de dimensión dos.

Como todo polígono puede ser dividido en partes convexas, se puede afirmar que las caras son polígonos limitados y convexos y para dicho propósito se dirá que las caras pueden ser *coplanares*[20].

El concepto de poliedro puede ser comprendido bajo las siguientes condiciones, las cuales serán esclarecidas por medio de ejemplos para así presentar formalmente a los poliedros en la Definición 2.10.

Se dice que una figura compuesta de vértices, lados y caras es un *poliedro* si y solo si:

i. Los componentes se intersecan "*apropiadamente*",

En esta condición, la palabra "*apropiadamente*" quiere decir que para cada par de caras se deben dar solo uno de los tres casos siguientes:

1. Las caras son disyuntas.
2. Las caras tienen un solo vértice común.
3. Las caras tienen dos vértices y el lado que los une en común.

A continuación se presentan ejemplos de figuras que no cumplen la primera condición.

Ejemplo 2.8.

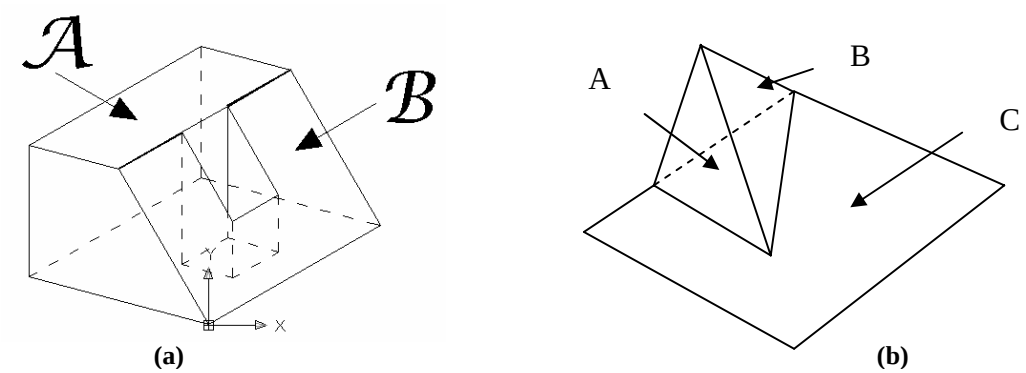


Figura 2.10. Ejemplos de figuras no poliédricas por intersección inapropiada de sus caras.

En la Figura 2.10 (a) las caras *A* y *B* se intersecan en dos lados de la cara *B*. en la Figura 2.10 (b) las caras *A* y *C* se intersecan en un lado de *A* pero en ningún componente contenido en *C*, por lo cual ninguna de estas dos figuras puede ser considerada como un poliedro.

20 Dos figuras son coplanares cuando están contenidas en un mismo plano.

ii. La topología local es "*apropiada*"

En esta condición, la palabra "*apropiada*" quiere decir que las vecindades de cada punto en la superficie son *homeomorfas*[21] a un disco abierto bidimensional. Esto es conocido como una *2-variedad*.

Otra manera de verlo es triangulando cada cara. La colección de lados opuestos a cada vértice debe ser una trayectoria poligonal simple y cerrada. Esta trayectoria es conocida como la *conexión* de dicho vértice. En consecuencia de esto, cada lado es compartido exactamente por dos caras, lo cual es muy importante para resultados posteriores.
Se dice que dos caras son *adyacentes* cuando comparten un lado del poliedro.

Los siguientes ejemplos son figuras que no cumplen la segunda condición.

Ejemplo 2.9.

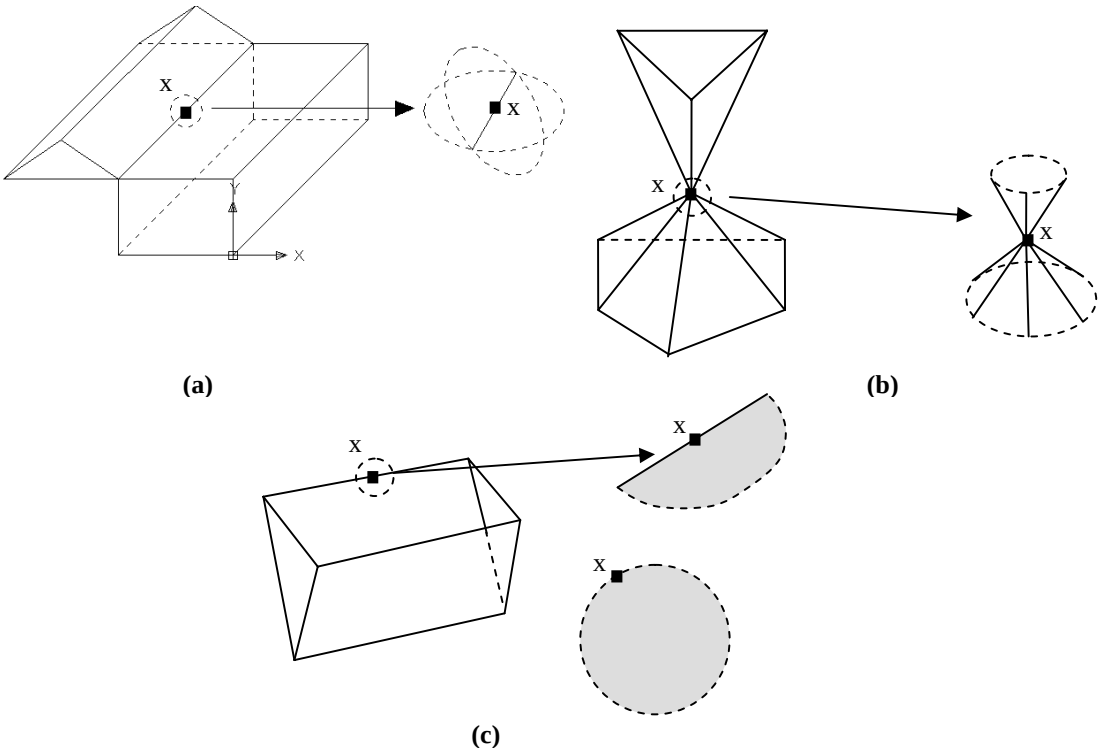


Figura 2.11. Ejemplos de figuras no poliédricas por su topología local.

Para todas las vecindades anteriores resulta imposible hacer modificaciones sin cortes ni pegamentos que las conviertan en un disco bidimensional. En la última en particular, es imposible que el punto *x* quede ubicado sobre la frontera del disco ya que dicho disco es abierto.

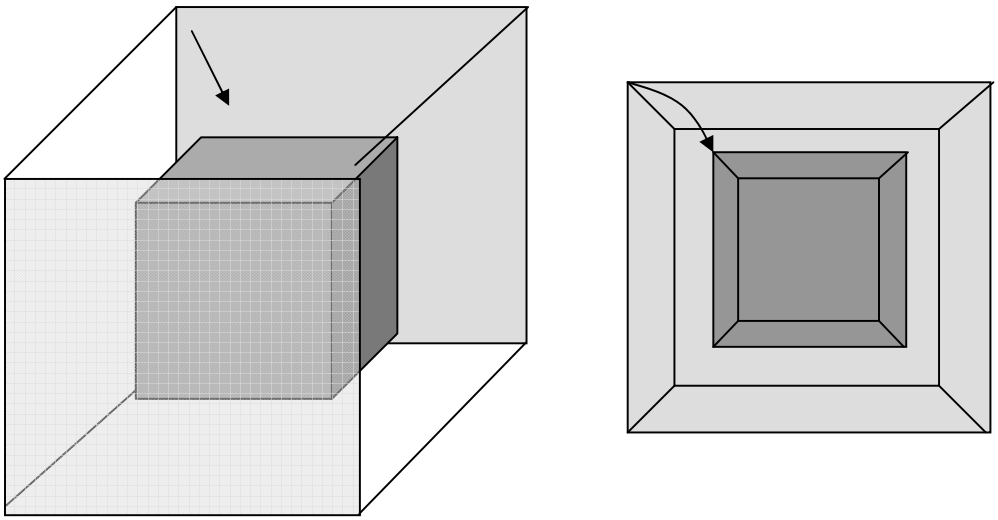
21 Se dice que dos objetos son homeomorfos si existe una función biyectiva y bicontinua definida de uno en el otro.

iii. La topología global es *"apropiada"*.

En la tercera condición, se quiere que la superficie sea cerrada, conexa y limitada, de modo que la palabra apropiada se refiere a el grafo inducido por los vértices y los lados de la figura (donde los nodos son los vértices y los arcos los lados) esté conectado, en otras palabras que se pueda caminar por los arcos de cualquier punto a otro en él.

Ejemplo 2.10.

En este ejemplo se tiene una figura formada por cubo con otro cubo flotando en su interior por lo cual no es un poliedro, ya que no hay ningún arco en el grafo inducido (es decir ningún lado de la figura), que conecte los nodos correspondientes al cubo interior con los nodos correspondientes al cubo exterior. Observe la Figura 2.12 (b) donde se muestra el grafo inducido.



(a) **(b)**

Figura 2.12. Ejemplos de figuras no poliédricas por su topología global.

Definición 2.10.
Se dice que una figura compuesta de vértices, lados y caras es un *poliedro* si y solo si:

i. Para un par de caras su intersección es o vacía o un vértice o un lado junto con sus respectivos vértices.
ii. La intersección de una bola abierta con centro en cada punto en la superficie y la superficie, es *homeomorfas* a un disco abierto bidimensional.
iii. La superficie de la figura es cerrada, conexa y acotada.

Nota:
¿Es posible que un toro sea un poliedro?

Las figuras que contienen “hoyos” que las atraviesen, lo cual es conocido como su *género*, también son consideradas poliedros, pero en este trabajo solo se van a considerar aquellas de *género 0*.

2.4.2. Politopos.

Definición 2.11.
Se dice que un poliedro es un politopo si y solo si es convexo.

Nota:
Para la convexidad se toman los puntos sobre la superficie.

Los politopos pueden ser caracterizados localmente con cualquiera de las siguientes condiciones:

- Los ángulos diedros son convexos, es decir menores o iguales a π.

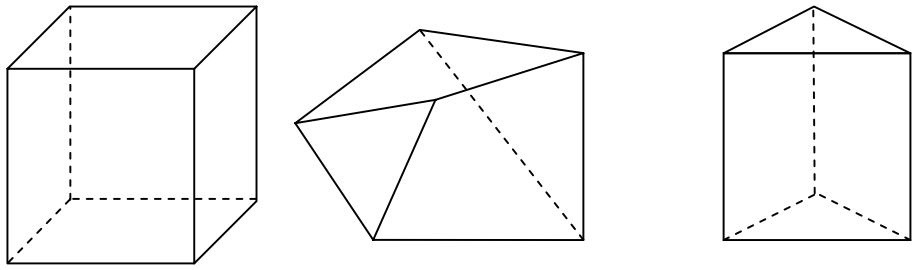
Estos son los ángulos internos formados por los planos que contienen las caras incidentes.

- La suma de los ángulos que rodean cada vértice es $\leq 2\pi$.

2.4.3. POLITOPOS REGULARES

Definición 2.12.
Se dice que un politopo es regular si todas sus caras son polígonos regulares congruentes y el número de caras incidentes es el mismo para cada vértice.

En la Figura 2.13 (a) se tiene un ejemplo de un politopo regular, mientras que las otras no lo son:



(a) **(b)** **(c)**

Figura 2.13. Ejemplos de figuras no poliédricas por su topología local.

Lo curioso es que solo hay cinco politopos que cumplen estas condiciones, conocidos como los sólidos platónicos. Esto se debe a que solo pueden ser construidos cinco polígonos regulares congruentes que puedan ser caras de un polítopo, ya que con el resto de ellos no es posible unir tres o más caras en un mismo vértice sin que se sobrepongan.

Teorema 2.1.
Solo hay cinco politopos regulares.

Demostración:
Sea *p* el número de vértices por cara en un polítopo, el cual debe ser mayor que *2* por la definición de polígono.

Cada cara será entonces un *p-gono*, es decir un polígono de *p* vértices.
La magnitud de cada ángulo de dichas caras es de $\pi(1-2/p)$.[22]
Sea *v* el número de caras con un vértice común, el cual debe ser mayor que *2* puesto que dichas caras deben compartir en forma cíclica las aristas del vértice que comparten y no es posible que la intersección de dos planos sea dos rectas diferentes en un espacio euclídeo.
La suma de los ángulos que rodean cada vértice es menor que 2π, luego

$$v\pi(1-2/p) < 2\pi$$
$$1-2/p < 2/v$$
$$pv < 2v+2p$$
$$pv-2v-2p+4 < 4$$
$$\mathbf{(p-2)(v-2) < 4}$$

Como *v* y *p* son enteros mayores que *2*, resulta fácil encontrar todas las posibilidades. Las cuales son expuestas en la tabla 2.1 y son los siguientes politopos:

p	**v**	**(p-2)(v-2)**	**V**	**E**	**F**	**Nombre**	**Descripción**
3	3	1	4	6	4	Tetraedro	3 triángulos en cada vértice
4	3	2	8	12	6	Cubo	3 cuadrados en cada vértice
3	4	2	6	12	8	Octaedro	4 triángulos en cada vértice
5	3	3	20	30	12	Dodecaedro	3 pentágonos en cada vértice
3	5	3	12	30	20	Icosaedro	3 triángulos en cada vértice

Tabla 2.1. Fórmula de Euler.

□

Para construir cualquiera de estos politopos será necesario calcular la cantidad de lados y caras para un cierto número de vértices. La fórmula que relaciona dichas variables fue demostrada por *Leonard Euler* en 1785 y lleva su nombre. Esta asegura que para cualquier poliedro, la suma de la cantidad de caras y la cantidad de vértices es igual al número de lados disminuido en dos. Esto puede ser comprobado fácilmente para los politopos regulares con la tabla 2.1 pero se demostrará en general para cualquier politopo:

Teorema 2.2.
Para un poliedro con $V = n$, *E y F* vértices, lados y caras respectivamente, se tiene que $V-E+F=2$ y calcular tanto a *F* como a *E* es un proceso de orden lineal.

22 O´ROURKE, pág. 14.

Demostración:
Sea *p* un politopo.
Aplaste sobre un plano el grafo inducido por el poliedro.
Elija una cara arbitraria *F* de *p* y remuévala, quedando un hoyo en la superficie y estírelo hasta que cubra todo el grafo.
Suponga que la grafo resultante es un árbol de *V* vértices y *E* lados.
Como un árbol delimita solo una cara y además la cantidad de vértices en un árbol es $V = E + 1$, entonces se tiene que $V – E + F = (E + 1) – E + 1 = 2$ lo cual demuestra la afirmación para árboles.
Ahora apliquemos el segundo principio de inducción matemática para demostrar que en la fórmula de Euler *E* puede tomar cualquier valor natural mayor o igual que *6*:

i. Si $E = 6$, entonces la solución es el tetraedro donde $F = 4$ y $V = 4$.

ii. Suponga que la fórmula de Euler se cumple para todo grafo de *E-1* lados.

iii. Sea *G* un grafo de *V* vértices, *E* lados y *F* caras.

Si *G* es un árbol se ha terminado, si no lo es entonces debe tener algún ciclo.
Sea *e* un lado en dicho ciclo y $G' = G - e$ el grafo resultante al suprimir el lado *e*; este grafo representa *V* vértices, *E-1* lados y, como cada lado es compartido por *2* caras, *F-1* caras.

Por ser un grafo de *E-1* lados, cumple la fórmula:

$$V - (E - 1) + (F - 1) = 2$$
$$\Rightarrow \quad V – E + 1 + F – 1 = 2$$
$$\Rightarrow \quad V – E + F = 2$$

Como el hecho de que se cumpla la fórmula para un grafo de *E-1* lados implica que también debe cumplirse para uno de *E* lados, la hipótesis de inducción comprueba la fórmula de Euler.

Ahora se demostrará el orden lineal:
Triangule el politopo de manera que los vértices de los triángulos coincidan con los del mismo. Sea $V = n$. De esta manera cada cara tiene *3* lados para un total de *3F* lados, pero como cada lado es compartido por *2* caras entonces,

$3F = 2E$ y como $V – E + F = 2$
$\Rightarrow V-E+2E/3=2$
$\Rightarrow V-2=E/3$
$\Rightarrow E=3V-6 < 3V = 3n = O(n)$ *y*
$\Rightarrow F=2E/3 = 2V-4 < 2V = 2n = O(n)$

□

2.5. ALGORITMOS TRIDIMENSIONALES.

En esta ocasión solo se expondrá en detalle el algoritmo incremental el cual es el más utilizado para el cálculo de la envolvente no solo en tres dimensiones sino también en dimensiones superiores por su sencillez. Este algoritmo es de $O(n^2)$ y es el mejor orden después del algoritmo de divide y vencerás que aunque es de $O(n \log n)$ su implementación es muy complicada.

2.5.1 Algoritmo Incremental.

La idea es muy sencilla, simplemente comience por definir la envolvente convexa de una nube de puntos en $\boldsymbol{R^3}$ como el tetraedro formado por cuatro de estos puntos. A continuación, agregue un nuevo punto teniendo en cuenta cuales son las caras visibles para dicho punto. La visibilidad será definida en términos del signo que se obtenga al calcular el volumen del tetraedro formado por cada una de las caras del tetraedro inicial y el punto a incluir. Si el volumen es negativo, el punto será omitido, pero si es positivo será necesario encontrar los lados de las caras visibles para los cuales el plano que contiene tanto al lado como al punto a incluir, es tangente a la superficie que se haya obtenido hasta el momento como envolvente. Una vez obtenidos estos lados, se procede a eliminar los vértices y lados que no resultaron "extremos" y a agregar en esta nueva iteración al punto encontrado como nuevo vértice en la envolvente y a las caras correspondientes a cada plano tangente, como caras de la misma. Este proceso se sigue repitiendo hasta agregar o eliminar a cada punto en la nube y en la iteración final se tendrá la envolvente convexa del conjunto completo.

Algoritmo 2.6. **Método Incremental**

1. Construya un tetraedro con cuatro puntos arbitrarios de la nube.
   Llámelo $H_4 = (p_0'. p_1'. p_2'. p_3')$
2. **Para** $i=4,\ldots, n-1$, **haga:**
   Para cada cara f de H_i, **haga:**
   Calcule el volumen de tetraedro formado por f y p_i.
   Marque a f como *visible* si y solo si el volumen es menor que 0.
   Si: No hay caras visibles, **entonces:**
   Descarte a p_i (Pues está en el interior de H_i).
   Si no:
   Para cada borde de H_i, **haga:**
   Construya la cara determinada por e y p_i.
   Para cada cara visible f, **haga:**
   Borre a f.
   Redefina a H_i.

Orden:
Los cálculos realizados en el interior de cada una de las secuencias están conformados por operaciones de orden lineal por lo visto en el Teorema 2.2, de manera que como a lo sumo hay dos secuencias anidadas cada una de $O(n)$, el orden total del algoritmo será $O(n^2)$.

Ejemplo 2.11.
En la Figura 2.14 se puede observar una iteración del Algoritmo 2.6 en la que se comprueba la visibilidad de un punto *p* y luego se traza el cono correspondiente a dicho punto y sus caras visibles.

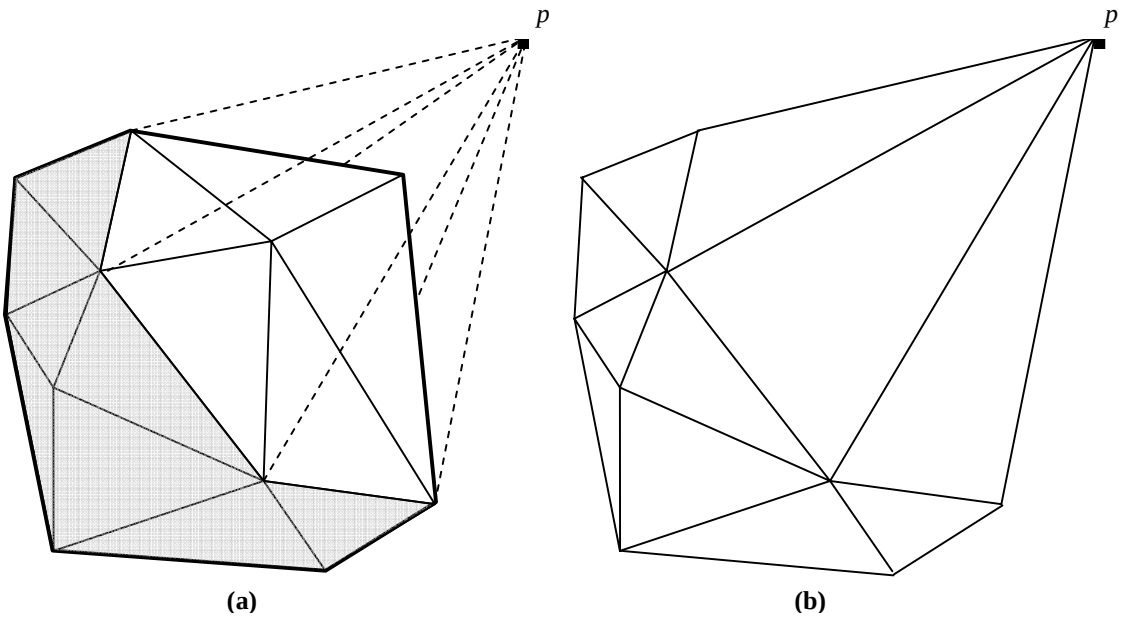


Figura 2.14. Algoritmo Incremental para tres dimensiones.

Con este ejemplo se da fin a este capítulo para pasar al estudio de la segunda estructura más importante de la geometría computacional que son los diagramas de Voronoi a los cuales se les dará una interesante aplicación en el Capítulo 4.

2.6. OBSERVACIONES.

- El algoritmo óptimo para encontrar la envolvente convexa de un conjunto de puntos en el plano es el algoritmo de Graham.
- Para ciertos fines, el algoritmo de envolvente Rápida resulta realmente eficiente.
- Siempre que se tenga la envolvente convexa, se tienen también dos cosas importantes: Un acotamiento del conjunto y todas las propiedades de los conjuntos convexos.
- Debido a la gran variedad de definiciones para la envolvente, queda abierta la propuesta a nuevos algoritmos quizá mejores.
- No todas las figuras tridimensionales son útiles para la construcción de la envolvente convexa.
- Dado el número de vértices de un poliedro, es posible encontrar el número de caras y lados del mismo en tiempo lineal.
- Solo existen cinco poliedros regulares.
- El algoritmo Incremental es un buen referente para calcular la envolvente convexa de una nube de puntos en tres dimensiones.

Capítulo 3

Diagramas de Voronoi

3.1. INTRODUCCIÓN.

En diversas áreas del conocimiento muchas veces resulta útil conseguir una variable implícita en los problemas que de alguna manera proporcione la información pertinente en cuanto a la cercanía o lejanía de un conjunto de objetos. Entre las estructuras más importantes en el campo de la geometría computacional se encuentran los *Diagramas de Voronoi*, los cuales logran de manera formidable realizar la caracterización anterior. Esta estructura está directamente asociada con una segunda conocida como *Triangulación de Delaunay* la cual es una de las mejores triangulaciones para figuras planas.
En el presente capítulo se definirá el diagrama de Voronoi y la triangulación de Delaunay, se expondrán sus propiedades y se explorarán algunas de las aplicaciones más comunes para dichas estructuras.

3.2. PROBLEMAS PRELIMINARES.

3.2.1. Torres de Vigilancia.

Suponga que en un bosque se han distribuido torres en diversos puntos en su interior a fin de poder vigilar el área completa del bosque ubicando un guardia en cada torre. ¿Cuál es el área del bosque que le corresponde vigilar a cada guardia ubicado en su respectiva torre, teniendo en cuenta que los árboles que le correspondan sean los más cercanos a él?

3.2.2. Torres en llamas.

En el caso anterior, suponga que por algún motivo las torres se encienden en llamas de manera simultánea. ¿Cuál es la zona del bosque en donde se extingue el fuego por causa de encontrarse con el fuego producido por otra torre?

3.2.3. Caracterización de objetos desconocidos por comparación al objeto conocido más cercano.

Dado un sistema cartesiano de coordenadas estrictamente enteras, ¿cuál es la mejor posición que se le puede asociar a los puntos cuyas coordenadas no son números enteros? En otras palabras, ¿cuál es la pareja *entera* que le corresponde por cercanía a cada una de las parejas *reales* en el plano cartesiano?

3.2.4. Ubicación estratégica.

En cierto cultivo se han distribuido semillas en diversos puntos conocidos de la superficie. Suponga que desea ubicar una nueva semilla de determinada planta en dicho cultivo de manera que quede lo más alejada posible de las semillas anteriores. ¿Cuál es la ubicación en la superficie que satisface dicha condición?

Todos estos interrogantes y muchos más, pueden ser abordados mediante una estructura geométrica que se construye precisamente basándose en las nociones de distancia y cercanía con la métrica usual en el plano, teniendo en cuenta que esta última noción no siempre se acomoda a las expectativas, ya que en muchas ocasiones ya sea por disposiciones geográficas, disponibilidad de comunicación, transporte, requerimientos políticos, entre otros, resulta mucho más "cerca" elegir un camino de mayor trayectoria. En la siguiente sección se define dicha estructura.

3.3. DIAGRAMAS DE VORONOI.

Definición 3.1.
Sea $P = \{p_0, p_1, ..., p_{n-1}\}$ un conjunto (o nube) de puntos llamados "*sitios*".
Se conoce como "*Región de Voronoi*" al conjunto:

$$V(p_i) = \{x : | p_i - x | \leq | p_j - x |, \forall j \neq i \},$$

y como "*Diagrama de Voronoi*" $\mathcal{V}(P)$ al conjunto de todos los puntos que tienen más de un vecino más cercano, es decir:

$$\mathcal{V}(P) = \cup [V(p_i) \cap V(p_j)], \; i \neq j.$$

Cada una de las líneas que resulta de intersecar dos Regiones de Voronoi no disjuntas se conoce como *lado de Voronoi*.
Los puntos que resultan de la intersección de tres Regiones de Voronoi se conocen como *Vértices de Voromoi*.

Ejemplo 3.1.
El diagrama de Voronoi para un conjunto de solamente dos sitios es simplemente la recta perpendicular al punto medio del segmento de recta que une los dos sitios, es decir la mediatriz del segmento que los une.
Las Regiones de Voronoi correspondientes son las dos regiones en las que queda dividido el plano. Note que estas regiones no son acotadas. Observe la Figura 3.1. (a).
El diagrama de Voronoi para tres sitios se determina trazando las mediatrices de los segmentos formados por cada par de puntos. Recuerde que estas rectas se cortan en un punto conocido como *circuncentro*. Las Regiones de Voronoi son las tres regiones en las que queda dividido el plano. Observe la Figura 3.1. (b).

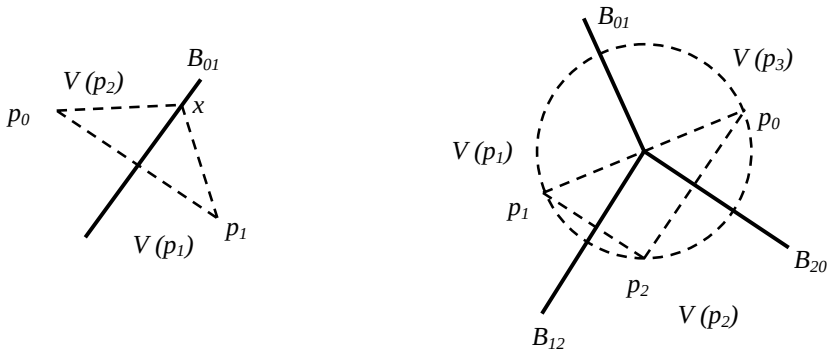


Figura 3.1. Diagrama de Voronoi para dos y tres sitios

De aquí en adelante el diagrama de Voronoi será obtenido de la misma forma que se hizo para tres sitios, pero se presumirá siempre que para cualesquiera cuatro puntos, estos no son *cocirculares*[23], ya que esto cancelaría un *lado de Voronoi* correspondiente a una de las diagonales de dicho rectángulo inscrito en el círculo que los contiene. Vea la Figura 3.2.

Observe que se obvian algunas líneas como por ejemplo la que separa a p_1 de p_3 en la Figura 3.2. (b) ya que esta los puntos que están sobre ella están más cerca de los sitios p_0 y p_2 de manera que queda por fuera de las Regiones de Voronoi de dichos puntos

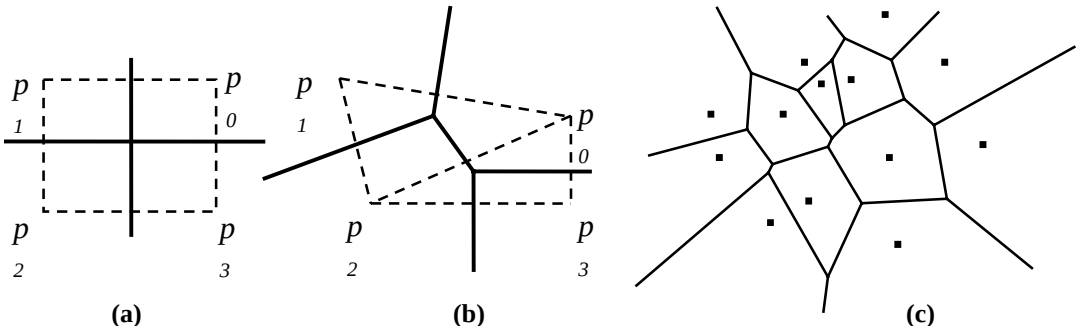


Figura 3.2. Diagrama de Voronoi para cuatro sitios cocirculares, cuatro no cocirculares y trece sitios

23 Es decir que no existe una circunferencia que los contiene.

3.4. TRIANGULACIÓN DE DELAUNAY.

Asociado al diagrama anterior es posible definir un grafo dual[24] trazando un segmento que una todos aquellos sitios cuyas Regiones de Voronoi compartan alguno de sus lados.

Definición 3.2.
Sea *S* un conjunto de *n* sitios en el plano. Se define como triangulación de Delaunay de *S* a la unión de las diagonales con puntos de *S* como vértices, tales que la intersección de las Regiones de Voronoi asociadas a dichos puntos es no vacía:

$\mathcal{D}(S) = \{d(p_i, p_j), V(p_i) \cap V(p_j) \neq \varnothing\}$, donde $d(p_i, p_j)$ es la diagonal de p_i a p_j.

Cada una de estas diagonales es conocida como *diagonal de Delaunay.*

Ejemplo 3.2.
En la Figura 3.3 se puede observar la triangulación de Delaunay correspondiente al diagrama de Voronoi del Ejemplo 3.1 para trece sitios.

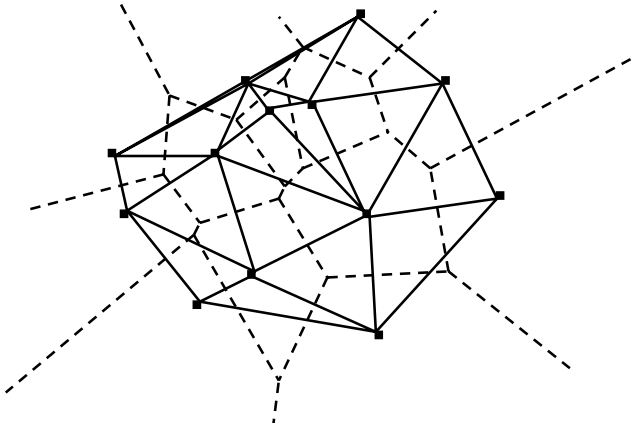


Figura 3.3. Triangulación de Delaunay para trece sitios.

Gracias a esta relación y como las caras y lados de un grafo pueden ser halladas linealmente[25], el proceso de construir el diagrama de Voronoi es de orden lineal.

24 Para profundizar en el concepto de grafo ver HARARY, Frank. *Graph Theory,* Addison-Wesley, 1972.
25 Ver el Teorema 2.2 en la página 40

3.5. PROPIEDADES.

3.5.1.Propiedades de las Triangulaciones de Delaunay.

1. $\mathcal{D}(P)$ es el grafo dual de $\mathcal{V}(P)$ trazando en línea recta las relaciones ente sus nodos.
2. **Teorema de Delaunay**: $\mathcal{D}(P)$ es una triangulación si no hay cuatro puntos de *P cocirculares*.
3. Cada triángulo de $\mathcal{D}(P)$ corresponde a un vértice de $\mathcal{V}(P)$.
4. Cada lado de $\mathcal{D}(P)$ corresponde a un lado de $\mathcal{V}(P)$.
5. Cada nodo de $\mathcal{D}(P)$ corresponde a una región de $\mathcal{V}(P)$.
6. La frontera de $\mathcal{D}(P)$ es la envolvente convexa de los sitios.
7. El interior de cada triángulo de $\mathcal{D}(P)$ no contiene sitios.

3.5.2.Propiedades de los Diagramas de Voronoi.

1. Cada Región de Voronoi $V(p)$ es convexa.
2. $V(p_i)$ es no acotado si y solo si p_i está en la envolvente convexa de los sitios.
3. Si v es un vértice de Voronoi en la intersección de $V(p_i)$, $V(p_j)$ y $V(p_k)$, entonces v es el centro de la circunferencia que los contiene ($C(v)$).
4. $C(v)$ es el *circuncírculo*[26] del triángulo de Delaunay correspondiente a v.
5. El interior de $C(v)$ no contiene sitios.
6. Si p_j es el vecino más cercano de p_i, entonces (p_i, p_j) es un lado de $\mathcal{D}(P)$.
7. (p_i, p_j) es un lado de $\mathcal{D}(P)$ si y solo si existe algún círculo que contenga a p_i y p_j en su frontera y que no contenga a otros sitios.

Teorema 3.1.
$ab \in \mathcal{D}(P)$ si y solo si, existe un círculo que contenga a a y b en su frontera y que no contenga a otros sitios.

Demostración:
Primero se demostrará de izquierda a derecha, es decir que si $ab \in \mathcal{D}(P)$, entonces existe un círculo a través de a y b que no contiene otros sitios:

Como ab está en la triangulación de Delaunay, entonces por definición $V(a) \cap V(b) \neq \varnothing$.
Sea x un punto en el interior del lado de Voronoi entre a y b. Construya un círculo con centro en x y como radio la distancia de x a a o a b (que es la misma).
Suponga que existe un sitio c contenido en el círculo, entonces $x \in V(c)$ y así tendría que estar en su interior o ser un vértice de Voronoi.
Si está en su interior, no debería estar en ninguna otra región de Voronoi así que se tendría una contradicción y si es un vértice de Voronoi no estaría en el interior de un Lado de Voronoi y se tendría otra contradicción.

26 Es decir la circunferencia circunscrita en el triángulo.

En conclusión, la hipótesis realizada es falsa, es decir que no hay otro punto en el interior del círculo construido. Vea la Figura 3.4. (a).

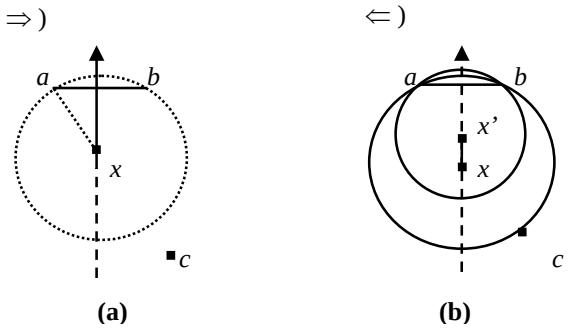


Figura 3.4. Teorema de Delaunay.

Ahora la demostración de la afirmación recíproca:

Suponga que existe un círculo a través de *a* y *b* que no contiene otros sitios de la nube.
Sea *x* el centro de este círculo. Así, la distancia de *x* a *a* es igual a la distancia de *x* a *b*. Luego *x* está en un Lado de Voronoi.
Sea *x'* el punto más cercano a *x* sobre la mediatriz de *ab* tal que la circunferencia con centro en *x'* pasa a través de *a, b* y *c*, siendo *c* otro punto de la nube.
El segmento *x x'* está contenido en un lado de Voronoi ya que contiene infinitos puntos del plano que equidistan de *a* y *b* y no hay otro punto de la nube que esté a menor distancia de ellos. La existencia de este Lado de Voronoi garantiza por definición que *ab* es una Diagonal de Delaunay, es decir que $ab \in \mathcal{D}(P)$.

□

3.6. ALGORITMOS

En esta ocasión, solo se expondrá un algoritmo que permite el cálculo del diagrama de Voronoi hallando primero la envolvente convexa de la imagen de los sitios bajo la función $f(x,y) = x^2 + y^2$, y luego de triangular cada una de las caras del paraboloide; se deduce que la proyección de estos triángulos sobre el plano no es nada más que la triangulación de Delaunay de los sitios, lo cual permite calcular de manera directa (lineal) el diagrama de Voronoi de la nube de puntos.

La exposición de este algoritmo, al igual que su implementación en *lenguaje C* serán presentados después de la mención de algunos algoritmos que se han desarrollado para el cálculo de diagramas de Voronoi y triangulaciones de Delaunay.

1. Algoritmo Incremental. $O(n^2)$.
   GREEN, P. y SIBSON, R. (1997).
2. Divida y Conquiste. $O(n \log n)$.
   SHAMOS, M. y HOEY, D. (1975).
3. Algoritmo de Fortune. $O(n \log n)$.
   O'ROURKE, J. (1995).

De los anteriores algoritmos se trabajará con el algoritmo incremental aún después de existir el algoritmo de Divida y Conquiste, ya que aunque proporciona un mejor orden, el primero es mucho más sencillo de implementar. A pesar de ello, cabe notar que el algoritmo de Fortune, es el más aceptado gracias a que reúne la sencillez del algoritmo Incremental con el orden del algoritmo de Divida y Conquiste.

3.7. IMPLEMENTACIÓN DEL ALGORITMO INCREMENTAL PARA HALLAR EL DIAGRAMA DE VORONOI DE UNA NUBE DE PUNTOS EN EL PLANO.

3.7.1. Visualización de la técnica en una dimensión.

Dado un punto *a* en el eje real, halle su imagen mediante la función:

$$f(x) = z = x^2 \qquad (3.1)$$

Como $dz/dx = 2x$, entonces la pendiente de la recta tangente a la parábola anterior en el punto *a* es *2a*.

De esta manera, la ecuación de la recta tangente a la parábola en el punto *a* es:

$$z - a^2 = 2a\,(x - a)$$

$$z = 2ax - 2a^2 + a^2$$

$$z = 2ax - a^2$$

la cual al ser trasladada una distancia r^2 hacia arriba corresponde a la siguiente ecuación:

$$z = 2ax - a^2 + r^2 \qquad (3.2)$$

y corta a la parábola en dos puntos, los cuales pueden ser hallados al igualar la Ecuación (3.1) y la Ecuación (3.2):

$$x^2 = 2ax - a^2 + r^2$$

$$x^2 - 2ax + a^2 + = r^2$$

$$(x - a)^2 = r^2$$

$$x = a \pm r \qquad (3.3)$$

Como se puede ver en la Ecuación (3.3), la Ecuación cuadrática (3.2) tiene exactamente dos soluciones reales que son precisamente los extremos del intervalo con centro en *a* y radio *r*.

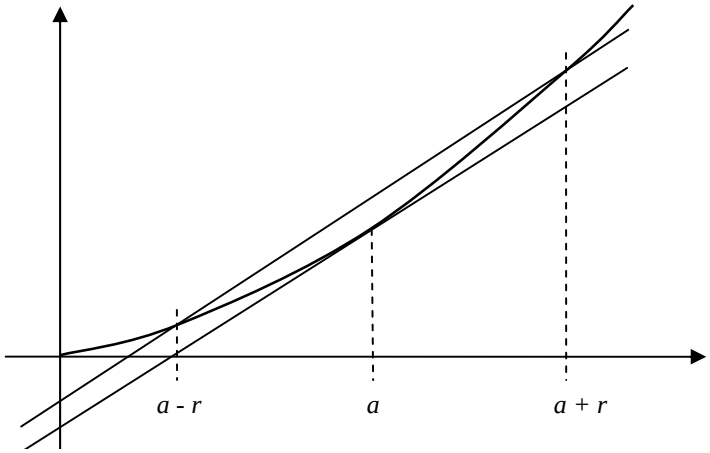


Figura 3.5. Proyección de una secante de la parábola sobre el plano.

Aplicando el mismo proceso en dos dimensiones se puede definir las características de la proyección de la envolvente convexa de una nube de puntos en tres dimensiones sobre el plano.

3.7.2. Visualización de la técnica en dos dimensiones.

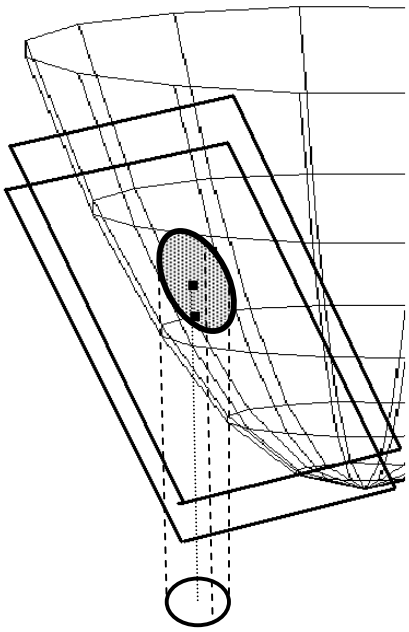


Figura 3.6. Proyección de un plano secante de un paraboloide sobre el plano.

Dado un punto (*a,b*) en el plano real, halle su imagen mediante la función:

$$f(x,y)= z = x^2 + y^2 \qquad \textbf{(3.4)}$$

Como $\partial z/\partial x=2x$ y $\partial z/\partial y=2y$, entonces la ecuación del plano tangente al paraboloide en el punto (*a,b*) es:

$$z = 2ax + 2by – (a^2 + b^2)$$
$$z = 2ax - a^2$$

la cual al ser trasladada una distancia r^2 hacia arriba corresponde a la siguiente ecuación:

$$z = 2ax + 2by – (a^2 + b^2) + r^2 \qquad \textbf{(3.5)}$$

y corta al paraboloide en una traza cuya proyección sobre el plano *x-y* puede ser hallada al igualar la Ecuación (3.4) y la Ecuación (3.5):

$$x^2 + y^2 = 2ax + 2by – (a^2 + b^2) + r^2$$
$$(x^2 - 2ax + a^2) + (y^2 - 2by + b^2) = r^2$$
$$(x - a)^2 + (y - b)^2 = r^2 \qquad \textbf{(3.6)}$$

La Ecuación (3.6) es la circunferencia con centro en (*a*,*b*) y radio *r*. En la Figura 3.6 se puede ver la circunferencia que se proyecta.

3.7.3. Proyección de las caras de la envolvente convexa sobre el plano.

Una vez se ha calculado la envolvente convexa en dos dimensiones mediante el algoritmo incremental de orden cuadrático, cada una de sus caras (polígonos) puede ser triangulada mediante el algoritmo de Chazelle[27] en un orden lineal o utilizando la trapezoidalización en un $O(n \log n)$ y trasladar hacia abajo cada uno de los planos en los que se encuentran los triángulos hasta obtener un punto en el que sean tangentes dichas traslaciones, que no son mas que el circuncentro del triángulo formado sobre el plano *x-y* por los tres vértices del triángulo sobre la cara de la envolvente. Esta circunferencia está libre de otros puntos ya que en caso de haber alguno en su interior, su imagen sobre el paraboloide estaría claramente por fuera de la envolvente encontrada inicialmente lo cual no es posible. De esta manera, como el interior del círculo no contiene otros sitios, entonces el centro del círculo es un vértice de Voronoi. En la Figura 3.7 se ilustra una proyección de cierto triángulo en alguna cara de la envolvente.

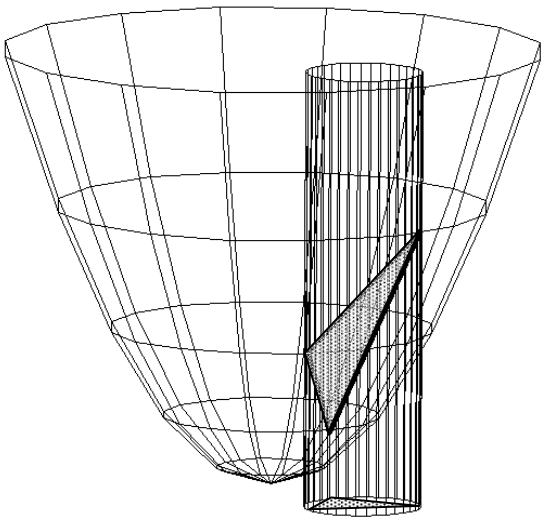


Figura 3.7. La triangulación de la envolvente convexa proyecta sobre el plano la triangulación de delaunay.

A continuación se presenta el código fuente en lenguaje C del algoritmo descrito. Este código es propuesto por O'Rourke como ejercicio y contiene además de los puntos correspondientes al diagrama de Voronoi, la gráfica correspondiente del mismo y de la triangulación de Delaunay.

27 CHAZELLE, B., *Triangulating a simple polygon in Linear Time*, 1991.

3.7.4.Implementación del algoritmo.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
 /* request auto detection */
 int gdriver = DETECT, gmode, errorcode;

 int x[200] , y[200] , z[200];       //Entrada de puntos (x,y,z), donde z=x^2+y^2
 int n;                              //N£mero de datos
 int i,j,k,m;                        //Indices para los puntos a comparar
 int xn,yn,zn;                       //Salida(vector) normal a i->j->k
 int flag;                           //Es cierto, si m est  por encima de i->j->k
 float A,B,C,D;                      //variables temporales
 float cx,cy;                        //Coordenadas de los V,rtices de Voronoi

 struct lado_voronoi
 {
  float x0,y0,xf,yf;
  int marca;
 }
 linea[200,200];
 free(linea);

 x[0]=-6;y[0]=4;
 x[1]=-4;y[1]=7;
 x[2]=-5;y[2]=1;
 x[3]=1;y[3]=7;
 x[4]=-1;y[4]=2;
 x[5]=-3;y[5]=-2;
 x[6]=4;y[6]=5;
 x[7]=3;y[7]=2;
 x[8]=2;y[8]=-4;
 x[9]=7;y[9]=2;
 x[10]=5;y[10]=-2;
 x[11]=-3;y[11]=4;


 //Entre los puntos y compute z=x^2+y^2
 printf("Entre el n£mero de puntos \n\n");
 scanf("%d",&n);
 for(i=0;i<n;i++)
    for(j=i+1;j<n;j++)
    {
     linea[i,j].marca=linea[i,j].x0=linea[i,j].y0=linea[i,j].xf=linea[i,j].yf=3;
    }

 for (i=0;i<n;i++)
     {
      if(i==0)
        printf("\nEntre las coordenadas x y y del primer punto \n\n");
      else
        printf("\nEntre las coordenadas x y y del siguiente punto \n\n");
      scanf("%d %d",&x[i],&y[i]);
      z[i]=x[i]*x[i]+y[i]*y[i];
      printf("\n%d\t%d\t%d",x[i],y[i],z[i]);
     }
     getch();
/* Inicialice el modo grafico */
   initgraph(&gdriver, &gmode, "C:\\TC\\BGI");

/* lea el resultado de la inicialización */
   errorcode = graphresult();
```

```
   if (errorcode != grOk)  /* un error ocurri¢ */
   {
      printf("Error Gr fico: %s\n", grapherrormsg(errorcode));
      printf("Presione cualquier tecla para continuar...");
      getch();
      exit(1);             /* regresa con error en el código */
   }

//Para cada tripla (i,j,k)

     printf("\n\nSitio1\tSitio2\tSitio3\tx-centro\ty-centro");
     outtextxy(300,279,"(0,0)");*/
     for(i=0;i<n-2;i++)
       for(j=i+1;j<n;j++)
          for(k=i+1;k<n;k++)
             if(j!=k)
              {
               //Compute la normal al tri ngulo ijk
               xn=(y[j]-y[i])*(z[k]-z[i])-(y[k]-y[i])*(z[j]-z[i]);
               yn=(x[k]-x[i])*(z[j]-z[i])-(x[j]-x[i])*(z[k]-z[i]);
               zn=(x[j]-x[i])*(y[k]-y[i])-(x[k]-x[i])*(y[j]-y[i]);

               //Examine solo las caras inferiores del paraboloide:zn<0
               flag=(zn<0);
               if(flag)
               //Para todos los dem s puntos m
                 for(m=0;m<n;m++)
                    //Revise si m est  encima del plano ijk
                 flag=flag&&((x[m]-x[i])*xn+(y[m]-y[i])*yn+(z[m]-z[i])*zn<=0);
               if(flag)
                 {
                  D=2*(y[i]*(x[k]-x[j])+y[j]*(x[i]-x[k])+y[k]*(x[j]-x[i]));
                  A=x[i]*x[i]+y[i]*y[i];
                  B=x[j]*x[j]+y[j]*y[j];
                  C=x[k]*x[k]+y[k]*y[k];
                  cx=((y[j]-y[k])*A+(y[k]-y[i])*B+(y[i]-y[j])*C)/D;
                  cy=((x[k]-x[j])*A+(x[i]-x[k])*B+(x[j]-x[i])*C)/D;
                  printf("\n\n%d\t%d\t%d\t%f\t%f\n",i,j,k,cx,cy);

//OJO: Las coordenadas m ximax son (639,479)

                  if(i<j)
                    {
                     if(linea[i,j].marca==1)
                     {
                      linea[i,j].xf=cx;
                      linea[i,j].yf=cy;
                      linea[i,j].marca=2;
                     }

                     else
                     {
                      linea[i,j].x0=cx;
                      linea[i,j].y0=cy;
                      linea[i,j].xf=0;
                      linea[i,j].yf=0;
                      linea[i,j].marca=1;
                     }
                    }
                  else
                    {
                     if(linea[j,i].marca==1)
                     {
                      linea[j,i].xf=cx;
                      linea[j,i].yf=cy;
                      linea[j,i].marca=2;
                     }
```

```
   else
    {
     linea[j,i].x0=cx;
     linea[j,i].y0=cy;
     linea[j,i].xf=0;
     linea[j,i].yf=0;
     linea[j,i].marca=1;
    }
  }
if(j<k)
  {
   if(linea[j,k].marca==1)
    {
     linea[j,k].xf=cx;
     linea[j,k].yf=cy;
     linea[j,k].marca=2;
    }
   else
    {
     linea[j,k].x0=cx;
     linea[j,k].y0=cy;
     linea[j,k].xf=0;
     linea[j,k].yf=0;
     linea[j,k].marca=1;
    }
  }
else
  {
   if(linea[k,j].marca==1)
    {
     linea[k,j].xf=cx;
     linea[k,j].yf=cy;
     linea[k,j].marca=2;
    }
   else
    {
     linea[k,j].x0=cx;
     linea[k,j].y0=cy;
     linea[k,j].xf=0;
     linea[k,j].yf=0;
     linea[k,j].marca=1;
    }
  }
if(i<k)
  {
   if(linea[i,k].marca==1)
    {
     linea[i,k].xf=cx;
     linea[i,k].yf=cy;
     linea[i,k].marca=2;
    }
   else
    {
     linea[i,k].x0=cx;
     linea[i,k].y0=cy;
     linea[i,k].xf=0;
     linea[i,k].yf=0;
     linea[i,k].marca=1;
    }
  }
else
  {
   if(linea[k,i].marca==1)
    {
     linea[k,i].xf=cx;
     linea[k,i].yf=cy;
     linea[k,i].marca=2;
    }
```

```
                    else
                     {
                      linea[k,i].x0=cx;
                      linea[k,i].y0=cy;
                      linea[k,i].xf=0;
                      linea[k,i].yf=0;
                      linea[k,i].marca=1;
                     }
                   }
                  getch();
                 }
             }

    for(i=0;i<n;i++)
     for(j=i+1;j<n;j++)
      if(linea[i,j].marca==1)
      {
       setlinestyle(DASHED_LINE,1,1);
       setcolor(4);
       printf("\n%d %d %d %f %f %f   %f",i,j,linea[i,j].marca,linea[i,j].
       x0,linea[i,j].y0,linea[i,j].xf,linea[i,j].yf);
       getch();
      }
    getch();
    closegraph();
    return 0;
}
```

Capítulo 4

Voronoi y los Frentes de Onda

4.1. INTRODUCCIÓN.

Los frentes de onda cumplen un papel importante en el estudio de los suelos y de todo lo referente a la teoría de propagación de ondas. Si se puede subdividir una superficie de manera que se generen regiones en donde la mayoría de sus elementos tengan las mismas propiedades, entonces los frentes de onda podrán ser estudiados en alguna de dichas regiones, para luego deducir su comportamiento en las intersecciones de las mismas, consideradas como *interfaces* que por medio de ciertos principios, proporcionaran las condiciones iniciales para estimar las características del frente en la siguiente región, hasta obtener un mapeo completo de la superficie.

4.2. CONCEPTOS PRELIMINARES.

4.2.1.Frente de Onda.

Se puede considerar un frente de onda como el conjunto de puntos que en cierto instante de tiempo se ven afectados por la misma perturbación, generada por medios artificiales como explosiones con dinamita o naturales como los sismos, entre otros.

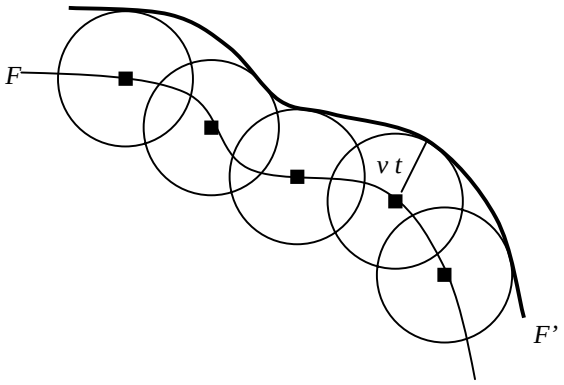
4.2.2.Principio de Huygens.[28]

"*Todo punto en un frente de onda puede ser considerado como una fuente puntual secundaria de otra frente de onda esférica"*.

Partiendo de un frente de onda *F* en determinado instante, este principio proporciona un método geométrico para deducir la forma y posición del mencionado frente en un instante de tiempo posterior (*F'*).

28 <http://www.sc.ehu.es/sbweb/fisica/ondas/snell/snell.htm>, 2007

Teniendo inicialmente un frente de onda, se puede construir otro frente secundario sobre él y desde allí trazar circunferencias de radio *vt* siendo *v* la velocidad de propagación de la onda en el punto del frente secundario y *t* el instante de tiempo en el que se pretende conocer las características del frente inicial. Vea la Figura 4.1; dicho frente será la envolvente de todas las circunferencias.



Si el medio de propagación conserva las mismas propiedades en todos los puntos y direcciones, se dice que es homogéneo e isótropo y el radio tendrá el mismo valor en todos los puntos.

Figura. 4.1. Principio de Huygens.

4.2.3.Ley de Snell.[29]

¿Cuál sería la forma después de un tiempo *t*, de un frente de onda que se acerca a la superficie de separación de dos medios con velocidades de propagación v_1 y v_2 respectivamente?

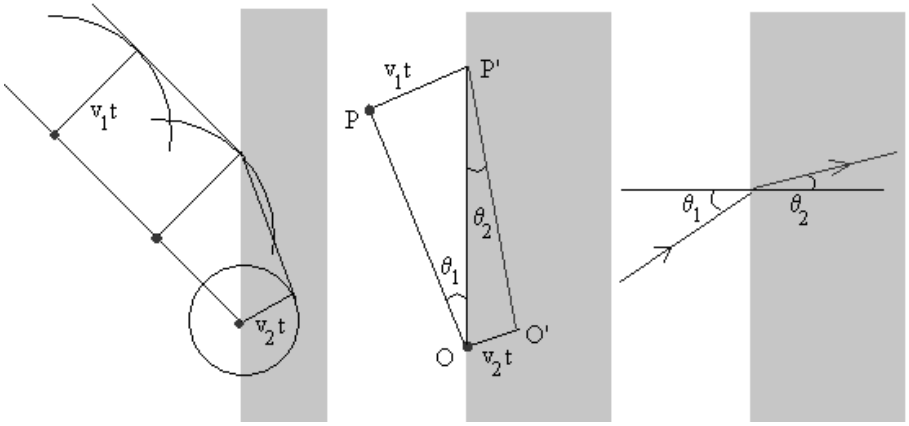


Figura 4.2. Ley de Snell.

Cuando un frente de onda incide en dos medios diferentes, el frente secundario produce ondas (representadas con circunferencias) que se propagan en todas las direcciones con una velocidad v_1 y v_2 en el primero y segundo medio respectivamente. La envolvente de dichas circunferencias proporciona la forma del frente en el instante t, que resulta ser una línea "quebrada", en comparación con la onda original.

29 <http://www.sc.ehu.es/sbweb/fisica/ondas/snell/snell.htm>, 2007

Usando propiedades trigonométricas se obtiene que la relación entre las velocidades de propagación y los ángulos θ_1 y θ_2 que forman los respectivos frentes con la superficie de separación resulta ser la siguiente:

$$\frac{v_1}{sen\theta_1} = \frac{v_2}{sen\theta_2} \qquad \textbf{(4.1)}$$

Cuando $\theta_1 < \theta_2$ y $v_1 < v_2$, a medida que θ_2 crezca, se acercará al límite $\theta_2 = \pi/2$, a partir del cual el frente no se transmite de un medio al otro sino que se refleja al medio inicial.

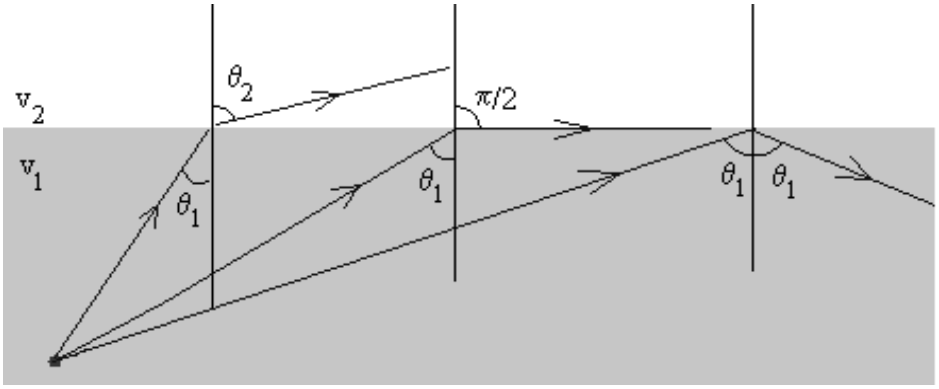


Figura 4.3. Ángulo límite de refracción.

4.2.4.Principio de Fermat.

"*La trayectoria de un rayo entre dos puntos, es aquella en la cual el tiempo de tránsito es un extremo (ya sea mínimo o máximo) en relación a todas las posibles trayectorias suficientemente próximas*".

4.3. CONSTRUCCIÓN DE FRENTES DE ONDA.

El problema de construir frentes de onda es ahora una aplicación del Principio de Huygens en la cual, el frente de onda en un instante de tiempo es calculado a partir del frente en el instante anterior aplicando el trazado de rayos cortos y cuando en el frente obtenido la distancia entre los extremos de los rayos es muy grande, basta solo con agregar una nueva fuente puntual como se muestra en la Figura 4.4. Este método es conocido como *WFC* del inglés "*Wavefront Construction*"[30]

30 COMAN, Radu y GAJEWSKI, Dirk, *Wavefront Construction Method with Spherical Interpolation*, 2000

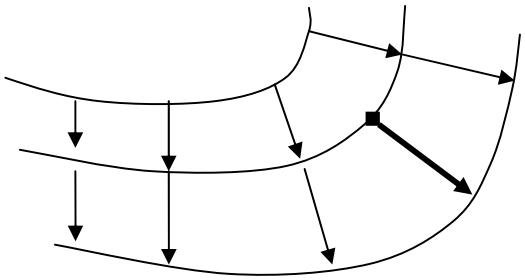


Figura. 4.4. Wavefront Construction

4.3.1. Problemas clásicos en la Propagación de Frentes de Onda.

1. Representación del frente por triangulación.
2. Reflexión de los rayos al entrar en una interfaz.
3. Localización de puntos en el frente.

El problema es sencillamente encontrar el frente de onda sobre cierta superficie para cierto tiempo.
La idea general del algoritmo parece ser sencilla pero sus detalles no son tan evidentes. Se presentan inconvenientes como ¿cuál es la mejor manera para la representación de los sitios, lados, regiones, rayos,…?, ¿cómo hallar de manera genérica los ángulos de refracción?, ¿cómo representar el frente cuando dos rayos se cruzan o se reflejan?

4.3.2. Método de construcción del frente de onda sobre una malla de Voronoi.

DATOS DE ENTRADA

Para comenzar, se supone que se conoce el diagrama de Voronoi de la superficie, en el cual se maneja la suposición que los puntos de la superficie conservan en forma homogénea las características de la región de Voronoi a la cual pertenecen y de esta manera, la velocidad de la onda será igual para todos los puntos en una misma región.

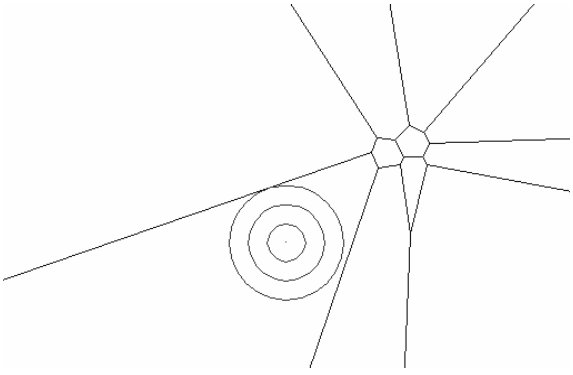


Figura. 4.5. Frente de onda en una misma región de Voronoi.

Como un ejemplo ilustrativo, en el diagrama de la Figura 4.5, el frente conserva su forma mientras se mantiene en la una misma región.

Cuando los rayos comienzan a intersecar con los lados de la región es necesario saber con cual lado se intersecó y cuál es el ángulo de refracción. Para saber el respectivo lado para cada rayo, primero se subdivide la región hallando los ángulos que forman los vértices con la horizontal (ordenados en forma antihoraria) en el punto inicial de cada rayo, que en la primera iteración es igual para todos los rayos. Vea la Figura 4.6.

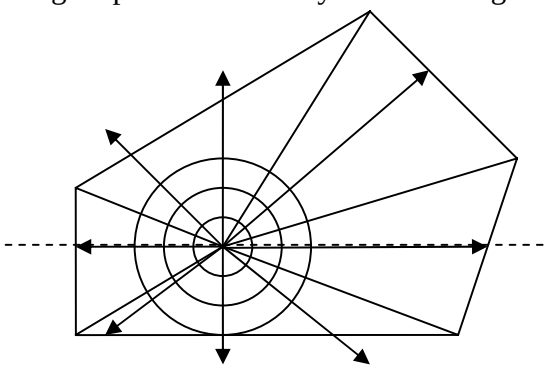


Figura. 4.6. Incidencia de los rayos del frente de onda.

Para cada iteración en la construcción del frente, se debe saber si el rayo se mantiene en la misma región o ha pasado a la siguiente. Para esto, se tienen dos posibilidades:

1. Preguntar si el punto inicial y el punto final del rayo se encuentran al mismo costado del lado.
   Nota: Es recomendable conservar el costado del punto inicial del rayo, para compararlo en todas las iteraciones posteriores hasta cambiar de región.

2. Calcular la distancia entre el punto de intersección y el punto inicial hasta que el rayo supere esta distancia.

De cualquier forma, al aplicar la ley de Snell se consiguen los diferentes ángulos de refracción cada vez que uno de los rayos atraviesa de un medio a otro.

El algoritmo resultante de este análisis es el siguiente:

Algoritmo 4.1. Construcción de frentes de onda sobre mallas de Voronoi.

Dado un conjunto de puntos en el espacio con coordenadas y características del suelo. (Para saber su ubicación en el espacio y la velocidad con la que se desplaza el frente de onda en cada región)

1. Interpole los puntos para obtener un modelo de la superficie.
2. Genere un pulso en cualquier punto de la superficie.
3. Calcule el diagrama de Voronoi de la superficie.
4. Calcule el frente de onda teniendo en cuenta las características del suelo proporcionadas por cada dato para todos los puntos que compartan determinada región de Voronoi.

En la Figura 4.7 se puede observar como se vería el frente de onda después de ciertas iteraciones en el Algoritmo 4.1.

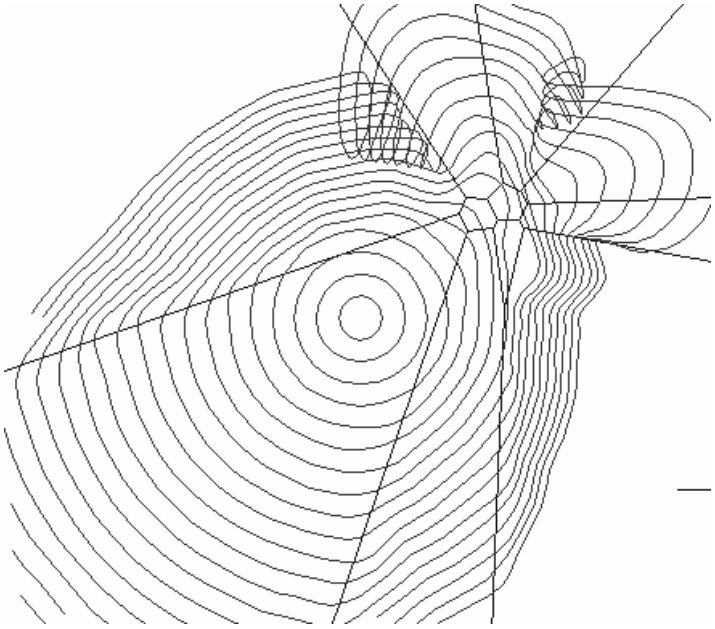


Figura. 4.7. Incidencia de los rayos del frente de onda.

Esto es solo un bosquejo inicial de lo que debe ser el programa que genere el frente de onda ya que para su completa implementación es necesario aplicar algunas teorías sobre el trazado de rayos de onda que se salen del alcance y objetivos de este documento.

BIBLIOGRAFIA

- APOSTOL, Tom M., *Análisis Matemático*, Reverté, 1989.
- CHAZELLE, B., *Triangulating a simple polygon in Linear Time*, 1991.
- CHVÁTAL, V., *A Computational Theorem in Plane Geometry*, 1975.
- COMAN, Radu y GAJEWSKI, Dirk, *Wavefront Construction Method with Spherical Interpolation*, 62nd EAGE Meeting, Glasgow, 2000
- FISK, S., *A Short Proof of Chvátal Watchman Theorem*, 1978.
- GREEN, P. y SIBSON, R., *Computing Dirichlet tesselations in the plane*, 1997
- HARARY, Frank. *Graph Theory,* Addison-Wesley. USA, 1972, Third printing.
- O'ROURKE, Joseph. *Computational Geometry in C*. Cambridge University Press, 1993
- PREPARATA, F. P. y Shamos, M. I.. *Computational Geometry: an Introduction,* Springer-Verlag, New York, 1985.
- SHAMOS, M. y HOEY, D., *Closest point problems,* 1975
- TARJAN, R. E. y VANWYK, J. C., *An O(log log n) Time Algorithm for Triangulating a simple polygon,* 1988.
- <http://www.dma.fi.upm.es/docencia/trabajosfindecarrera/programas/geometriacomputacional/>, 2005
- <http://www.sc.ehu.es/sbweb/fisica/ondas/snell/snell.htm>, 2007