Construyendo Mapas Topográficos

Viviana Carrizales Luna

Resumen

Este estudio presenta un análisis exhaustivo de los algoritmos para la construcción de triangulaciones de Delaunay, junto con la implementación de un programa innovador para la interpolación de alturas. Se investiga en detalle la eficacia y la precisión de los métodos de triangulación de Delaunay en el procesamiento de conjuntos de datos irregulares. El estudio también examina la viabilidad del algoritmo del vecino más cercano para la interpolación de valores de altura en datos geoespaciales dispersos. Para representar las alturas de los datos interpolados, se utilizó una escala de colores que se asignó a diferentes intervalos de alturas. Se optó por una escala de colores gradiente que va desde tonos más amarillos, representando alturas más bajas, hasta tonos más azulados, que indican alturas más altas. Esto permitió una representación visual efectiva de las variaciones en las alturas interpoladas en el área de estudio, facilitando la identificación rápida de patrones y tendencias geoespaciales.

Abstract

This study presents a mathematical analysis of algorithms for constructing Delaunay triangulations, along with the implementation of a program for height interpolation, taking advantage of the effectiveness and precision of Delaunay triangulation methods in processing irregular data sets are . The study also investigates the feasibility of the nearest neighbor algorithm for interpolating height values in dispersed geospatial data. To represent the heights of the interpolated data, a color scale was used, assigned to different height intervals. A gradient color scale was chosen, ranging from more yellowish tones representing lower heights to bluer tones indicating higher heights. This allowed for an effective visual representation of variations in interpolated heights in the study area, facilitating the rapid identification of geospatial patterns and trends.

Palabras Clave:

Diagrama, estructura, triángulo, interpolación, computo, listas, programación, rango, color.

1. Introducción

Resulta imposible hablar de las triangulaciones Delaunay y los diagrama Voronoi sin primero considerar la rama de estudio a la cual pertenecen. La geometría computacional. Si bien es una rama de las ciencias de la computación, sus fundamentos yacen en las matemáticas, pues se enfoca en el estudio de algoritmos que pueden ser construidos mediante propiedades geométricas.

1.0 Por qué geometría computacional

Con la finalidad de explicar su importancia en las ciencias computacionales, se presentará un problema y la forma en la que la implementación de esta rama disminuye en gran medida el tiempo de cómputo para resolverlo. Cabe señalar que no se demostrarán los teoremas utilizados solo se expondrán las ideas utilizadas.

¿Cómo podemos detectar la colisión de dos conejos? (este problema es especialmente recurrente en la programación de videojuegos)

Primero se comienza triangulando ambos conejos (esto es un problema de geometría computacional por sí solo), y proyectaremos cada triángulo de cada conejo a los 3 ejes, (x, y, z). Si existe un empalme de estas proyecciones en los 3 ejes habrá una colisión.

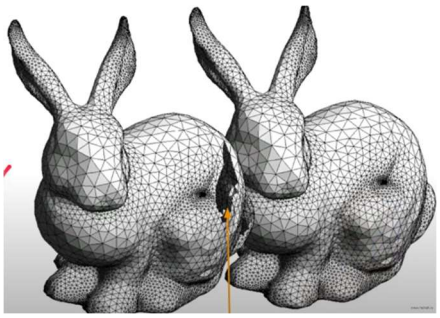


Figura 1. Triangulación de conejos. (Screenshot de Shah, M. 2023)

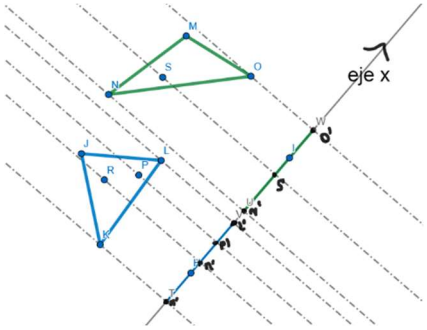


Figura 2. Proyección de triángulos a ejes x,y,z del software Geogebra.

Si bien este proceso es preciso y correcto, el costo de computación sería muy alto. Debido a la gran cantidad de triángulos que cada conejo presenta. Es por eso que se busca una forma de aplicar el mismo método para una cantidad menor de triángulos, en este caso se buscará el casco convexo de cada conejo.

Definición 1.21: El casco convexo de un conjunto de puntos en el espacio euclidiano, es la figura convexa más pequeña que los contiene.

Nota: Existen cascos convexos en más de 3 dimensiones.

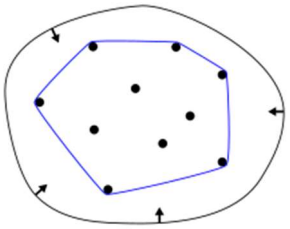


Figura 3. Casco convexo en 2d. (Sears, M & Alruwaythi, O & Goodrum, P. 2018)

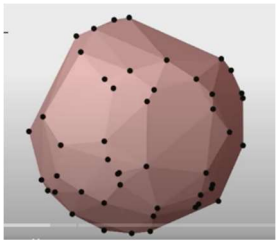


Figura 4. Casco convexo en 3d. (Screenshot de Kindermann, P. 2021)

Al utilizar el casco convexo de cada conejo, se puede determinar si existirá una colisión mediante el mismo método, sin perder mucha precisión.

Sin embargo, ahora se tiene que resolver un problema intermediario. Cómo se computa el casco convexo para una cantidad P de vértices en el plano. El empleo de algoritmos para construir figuras para disminuir el tiempo de cómputo de un problema es la base de la geometría computacional.

2. Marco Teórico

2.0 Primeras definiciones

Definición 2.01: Un simplex es la generalización de un triángulo para d dimensiones. Representa el politopo más sencillo en cada dimensión.

Observación: para d dimensiones, el simplex tendrá d+1 vértices.

Definición 2.02: Una faceta es un elemento de n-1 dimensiones de un politopo de n dimensiones

Definición 2.03: Una triangulación de un conjunto de puntos P es un complejo simplicial que cubre el casco convexo de P.

2.1 Triangulaciones de Delaunay

Definición 2.1: Dado un conjunto de puntos P, un triángulo con vértices $a,b,c \in P$ cumple la condición Delaunay si $\forall\ p \in P$, p no está dentro del circuncírculo de Δabc. Una triangulación es Delaunay si todos los triángulos que la confirman cumplen la condición Delaunay.

Primeras propiedades

A continuación, se enlistarán algunas propiedades que hacen a la triangulación Delaunay sujeto de estudio. En esta sección no serán exploradas ni demostradas, pero en otras secciones algunas se explorarán a más a detalle.

-No es posible agregar otro arista sin obtener cruces de aristas.

-La unión de todos los simpleces en la triangulación es el casco convexo de los puntos.

-Un par de puntos es una arista de la triangulación ⇔ existe un circuncírculo de otro triángulo perteneciente a la triangulación que lo intersecta.

-El árbol generador mínimo es una subgráfica de las triangulaciones de Delaunay.

Nota: Aunque no será utilizado en esta exploración, es de gran importancia al ser un concepto fundamental de otra rama de estudio. Teoría de grafos.

-Maximiza el ángulo mínimo de cada triángulo (evitando "triángulos delgados")

2.2 Diagramas Voronoi

Definición 2.21 Dado un conjunto de P puntos de n puntos en el plano:

Llamemos Vor(P)= subdivisión del plano en celdas de Voronoi, aristas y vértices y a una celda de Voronoi $V(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \; \forall \; q \in P\}$

$$V(p) \in P$$

-El diagrama de Voronoi y la triangulación Delaunay son duales. (correspondientes)

Para transformar el diagrama de Voronoi a la triangulación Delaunay basta con conectar los vértices de las células adyacentes.

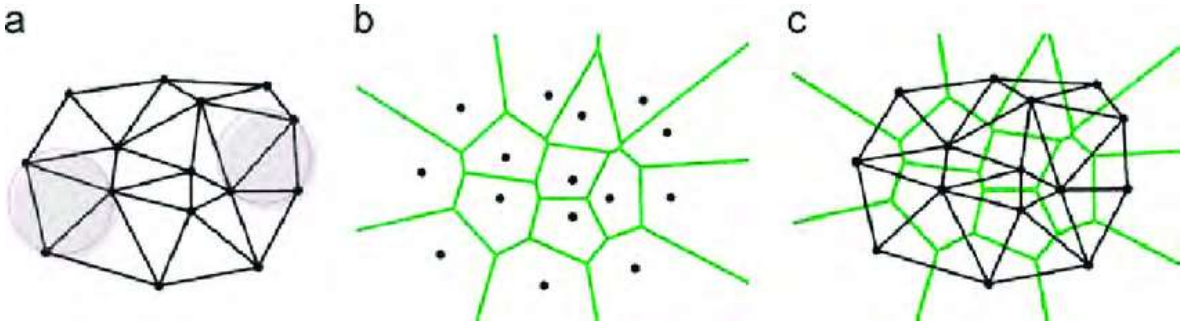


Figura 6. Dualidad diagrama Voronoi y triangulación Delaunay (Karimipour, Farid & Ghandehari, Mehran & Ledoux, H, 2013).

La razón de esta dualidad está directamente relacionada con la construcción del diagrama de Voronoi. Se observa que cada arista del diagrama es la mediatriz de la unión de los vértices.

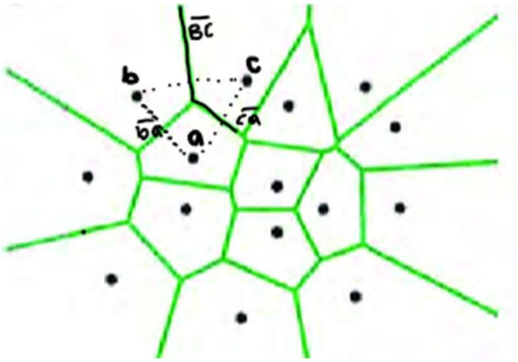


Figura 5. Demostración de Dualidad (Karimipour, Farid & Ghandehari, Mehran & Ledoux, H, 2013).

Ahora se construye la triangulación de la forma previamente descrita. Se supone que existen los vértices *a, b, c* t.q ∋ otro vértice *p* dentro del circuncírculo del Δabc, se observa que *p* estará dentro de algunas de las células con vértice *a,b o c* debido a que la intersección de los aristas del diagrama de Voronoi *bc, ca y ba* es el circuncírculo de Δabc!, por lo tanto, la triangulación es Delaunay.

Para transformar de la triangulación Delaunay al diagrama de Voronoi se utiliza un procedimiento análogo y su demostración utiliza las mismas propiedades e ideas. Esta dualidad es de importancia pues podemos computar la triangulación Delaunay mediante el diagrama Voronoi o viceversa.

2.3 Algoritmos para construcción:

Una vez que se ha definido una triangulación Delaunay, se expondrán dos algoritmos que son utilizados para su computación.

1) Flipping

Definición 3.1 Sea una triangulación *T* de un conjunto *P* de puntos en el plano, una arista es "*cambiable*" si pertenece a dos triángulos cuya unión es un cuadrilátero convexo C. Llamaremos a la operación "*cambio*" a la acción de tomar un diagonal de C y remplazarla por otra.

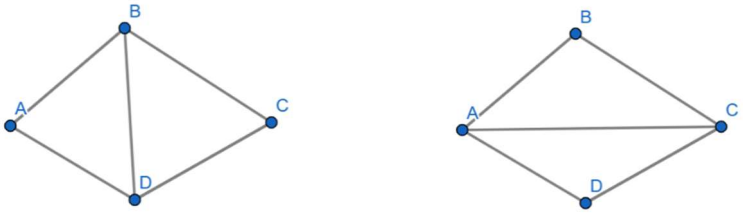


Figura 8 y 9. Cambio de la arista BD a AC del software Geogebra.

Uno de los algoritmos utilizados para la construcción de las triangulaciones de Delaunay consiste en considerar una triangulación cualquiera de P y hacer *cambios* de aristas hasta obtener la triangulación de Delaunay. Para demostrar que este proceso es finito, es necesario considerar otra triangulación.

Definición 3.2: Una triangulación T' es óptima si no existe ninguna triangulación T con ángulos menores mayores. Es decir $A(T') > A(T)$ donde $A(T)=(\alpha_1, \alpha_2, \dots, \alpha_{3m})$, m la cantidad de triángulos en T y $\alpha_1 \leq \alpha_2 \leq \dots, \leq \alpha_{3m}$

Lemma: Una triangulación es optima $\Leftrightarrow$ es Delaunay

Definición 3.3: Sea T una triangulación. Una arista e de T es *ilegal* si el mínimo ángulo entre los dos triángulos continuo a e aumenta cuando lo cambiamos.

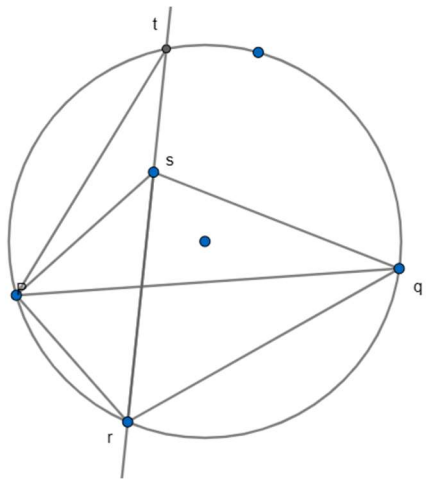
Observación: Sea e una arista ilegal de T y T' = giro $(T, e) \Rightarrow A(T')>A(T)$.

Lemma Sean $\Delta pqr, \Delta pqs \in T$ y $p, q, r \in circunferencia\ \Omega \Rightarrow$ *la arista* pq *es ilegal* $\Leftrightarrow s \in interior\ de\ \Omega$

Demostración:

Primero se nota que si p, q, r, s no están en una posición convexa entonces s está en el interior de Δpqr por lo tanto pq no es una arista !

Por lo tanto, se tiene que p, q, r, s están en posición convexa. Se va a demostrar que
$$\forall \alpha' \ en\ T' \ \exists\ \alpha\ en\ T \ \ t.q\ \alpha < \alpha'$$



Siendo T' la triangulación que contiene la arista sr y T la triangulación que contiene la arista pq

Se observa que para $\measuredangle spr$ en T', $\measuredangle spq, \measuredangle qpr$ son menores en T, y para $\measuredangle sqr$ en T' $\measuredangle sqp, \measuredangle pqr$ son menores en T.

Figura 10. Triangulación *legal* del software Geogebra.

Para $\measuredangle prs$, basta considerar $\measuredangle pqs$ pues se sabe que $\measuredangle prs = \measuredangle pqt > \measuredangle pqs$, análogamente para $\measuredangle srq$ se considera $\measuredangle spq$, por lo tanto la arista pq es ilegal.

Nota: Si $s \in \Omega$, ambas aristas, pq y rs son legales

Se observa que para cada arista ilegal en la triangulación *pq,* podemos cambiarla a *rs* y obtener una triangulación legal que además es óptima. Al ser legal para cada cuadrilátero convexo *pqrs*, *s* no está en el interior del circuncírculo del Δpqr por lo tanto se puede hacer este proceso hasta obtener una triangulación legal.

Algoritmo: Consideremos T una triangulación de P. Mientras $\exists$ una arista ilegal e, se cambia e. Se observa que esto es un proceso finito pues los ángulos menores aumentan por cada cambio y debido a que hay una cantidad finita de triangulaciones, $\exists$ una con $A(T)$ mayor, es decir con los ángulos menores mayores.

Por lo tanto, es posible una triangulación Delaunay a partir de cambios de aristas.

2)Lifiting

Otro de los métodos utilizados para computar la triangulación Delaunay de un conjunto P de puntos, es considerarse un paraboloide por encima del plano *P1* y proyectar cada punto.

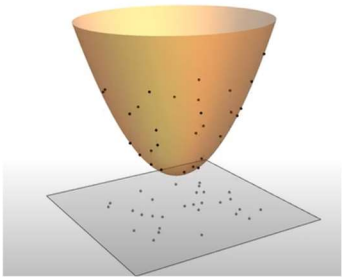


Figura 11. Computing Delaunay complex: Lifting to a paraboloid . (Screenshot de Draganov, O. 2021)

Después se computa el casco convexo de los puntos en 3d y se proyecta hacia el plano *P1*.

Nota: Las facetas del casco convexo en 3d, deben de ser simpleces para poder obtener una triangulación.

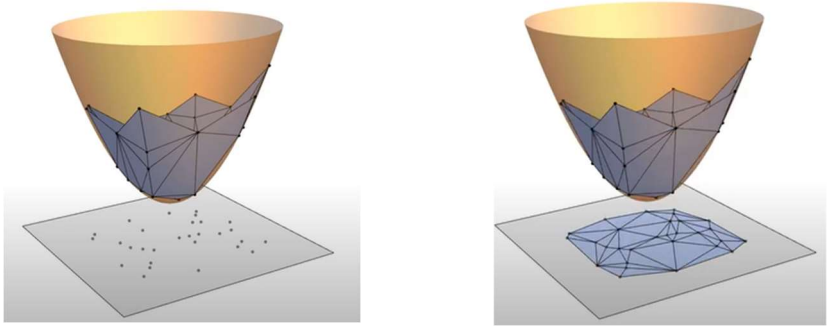


Figura 12 y Figura 13. Computing Delaunay complex: Lifting to a paraboloid . (Screenshot de Draganov, O. 2021)

Demostración: Primero se observa que si cortamos el paraboloide con cualquier plano *P2* y se proyecta cada punto donde se intersectan, se obtiene un círculo. Esto es sencillo de ver en una animación, pero se demostrará algebraicamente.

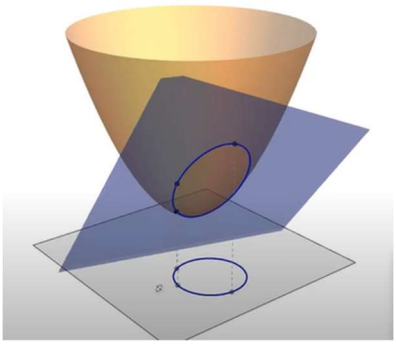


Figura 14. Computing Delaunay complex: Lifting to a paraboloid . (Screenshot de Draganov, O. 2021)

Sea $z = x^2 + y^2$ la ecuación de el paraboloide y $lx + my + nz = d$ la ecuación del plano P.

Por la intersección se obtiene que $z = \frac{d-lx-my}{n} = x^2 + y^2 \Rightarrow$

$$x^2 + y^2 = ax + by + c, \qquad a, b, c \in \mathbb{R} \Rightarrow$$

$$x^2 - ax + \frac{a^2}{4} + y^2 - by + \frac{b^2}{4} = c + \frac{a^2}{4} + \frac{b^2}{4} \Rightarrow$$

$$\left(x - \frac{a}{2}\right)^2 + \left(y - \frac{b}{2}\right)^2 = c + \frac{a^2}{4} + \frac{b^2}{4}$$

Se observa que esta ecuación representa a cada punto una vez que lo proyectamos a el plano *P1* ya que si un punto tenía coordenadas (x,y,z) al proyectarlas al plano, tendrá coordenadas (x,y,0). Como $a, b, c \in \mathbb{R} \Rightarrow \left(x - \frac{a}{2}\right)^2 + \left(y - \frac{b}{2}\right)^2 = c + \frac{a^2}{4} + \frac{b^2}{4}$ representa la ecuación de un circulo en el plano *P1.*

Se puede ver que un punto p estará dentro del circuncírculo abc en el plano *P1* $\Leftrightarrow$ su proyección p' está por debajo del plano *P2* definido por *a'b'c'.*

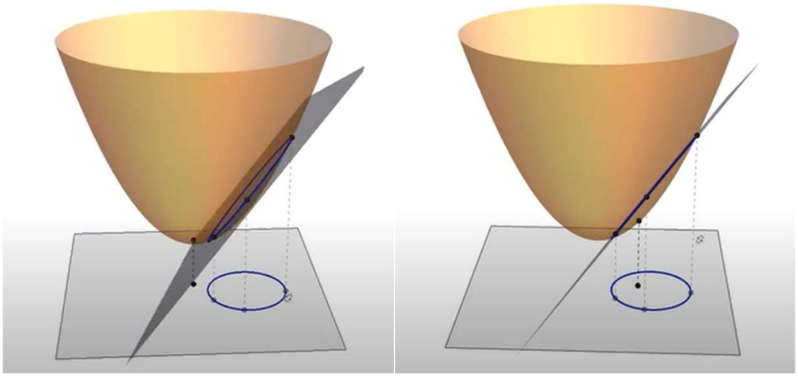


Figura 15 y Figura 16. Computing Delaunay complex: Lifting to a paraboloid . (Screenshot de Draganov, O. 2021)

Debido a que *a'b'c'* definen una cara del casco convexo de todos los puntos proyectados a el paraboloide, $\nexists p \in P' t.q\ p' está\ por\ debajo\ del\ plano\ definido\ por\ a'b'c'$

Por lo tanto, al proyectar cada punto de vuelta a *P1,* obtenemos una triangulación Delaunay.

Computar la Triangulación Delaunay de esta forma es útil debido a que existen algoritmos para computar el casco convexo inferior de manera muy eficiente.

3 Aplicaciones

a. Interpolación de alturas

Supongamos que se busca construir un mapa topográfico de un país, medir la altura de cada punto es imposible, por lo que se medirá la altura de algunos puntos y se hará una interpolación para el resto.

Dado un conjunto P de puntos cuyas alturas se conoce, se supone que se busca conocer la altura de un punto $q \notin P$, para hacer esto primero se proyectan todos los elementos de P a un plano de manera ortogonal obteniendo el conjunto P'. Se triangula P'. Se interpola de la siguiente manera. Si q pertenece a una arista de la triangulación, la altura de q será el promedio de los dos vértices extremos de la arista, si q no pertenece a un arista entonces está dentro de un triángulo, por lo que tendrá como altura el promedio de los tres vértices que crean el triángulo. Se observa que hay distintas formas de triangular los puntos. Por ejemplo:

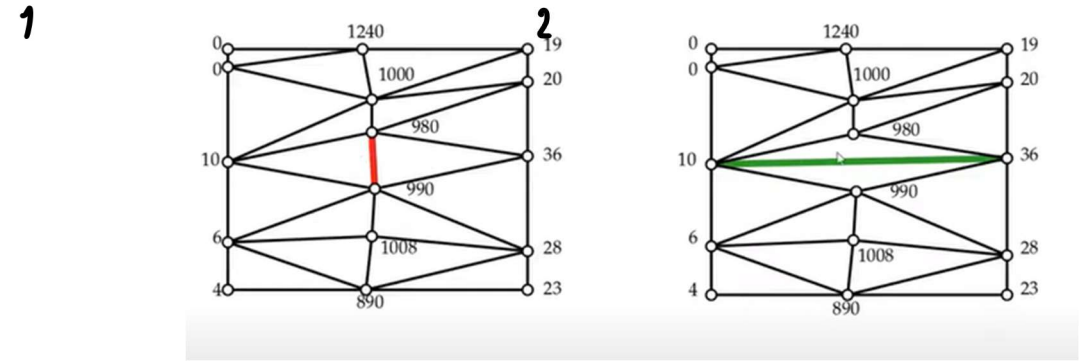


Figura 17. Triangulaciones distintas con cambio de arista. (Screenshot de Kinderman, P, 2021)

Al cambiar la arista roja por la arista verde se obtiene una nueva triangulación. Sea q la intersección entre la arista verde y la arista roja. Si se utiliza la triangulación 2 se obtendrá que la altura del punto q es 23, si se utiliza la triangulación 1, se obtendrá como altura 985. Se observa que entre más largo sea la arista más imprecisa será la altura de q, ya que entre más distancia hay entre dos puntos, mayor es la probabilidad de que cambie el relieve de manera abrupta. Por lo tanto, se quiere evitar triángulos "delgados" o con ángulos pequeños. Esto es una propiedad de las triangulaciones Delaunay y es por eso que son utilizadas en la interpolación de alturas.

Nota: Si se aumenta la cantidad de puntos en P, se obtendrá mayor precisión en la interpolación pues tendremos una mayor cantidad de triángulos, por lo que es especialmente importante tener algoritmos eficaces para construir las triangulaciones Delaunay para una gran cantidad de puntos.

Un método de interpolación que utiliza ideas similares es aplicado en los diagramas Voronoi. Se le conoce como algoritmo para la interpolación del vecino natural.

A continuación, los pasos que sigue el algoritmo:

Considerando el conjunto de puntos P, se construye el diagrama Voronoi de estos puntos. Se agrega el punto q y se construye el diagrama Voronoi del conjunto de puntos resultante de la unión de P y q. Sea Q a la celda que contiene el punto q. Para calcular el valor aproximado de q, se hará lo siguiente.

Notemos que Q contendrá subregiones de celdas del diagrama original. Se obtiene el porcentaje de cada subregión con respecto a las celdas vecinas originales y a esa subregión se le asigna le valor de cada punto original multiplicado por ese porcentaje. El valor de q será la sumatoria de todos los valores asociados a cada área.

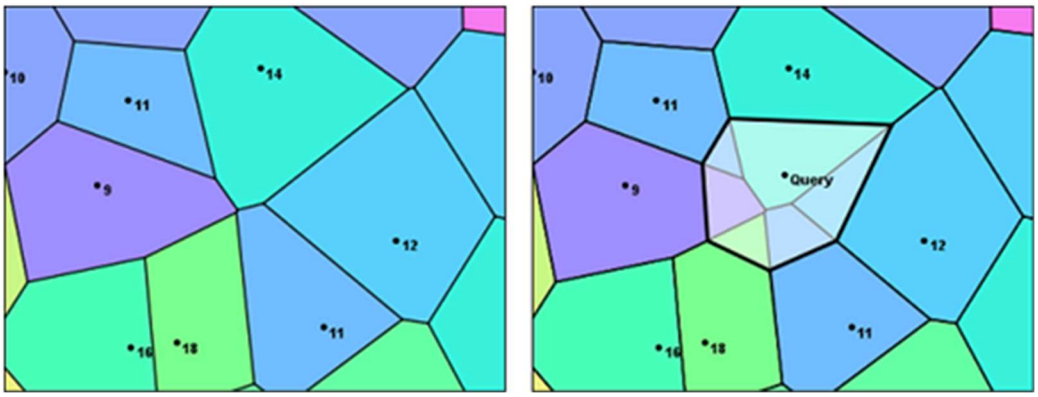


Figura 18. Diagrama Voronoi original y diagrama Voronoi después de agregar q. (Screenshot de Lucas, G, 2021)

Se observa que el valor asociado a cada subregión será directamente proporcional a la cercanía de q, son sus puntos vecinos. De esta forma si se obtiene una subregión con área muy pequeña, el valor asociado a esa región será muy pequeño ya que el porcentaje de esa subregión con respecto a la región a la que pertenece en el diagrama original será casi despreciable. A continuación, un ejemplo de esto:

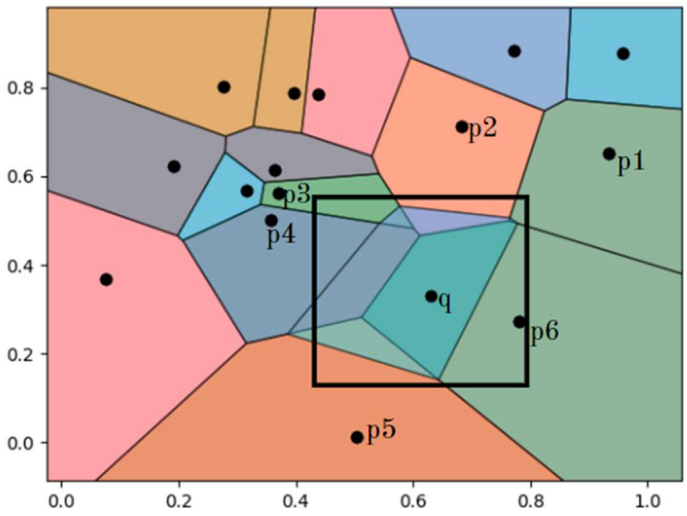


Figura 19. Diagrama Voronoi después de agregar el punto q. (Autoría propia, 2023)

Al agregar q, se obtiene la región Q que tiene subregiones pertenecientes a las celdas de los puntos p1, p2, p3, p4, p5 y p6.

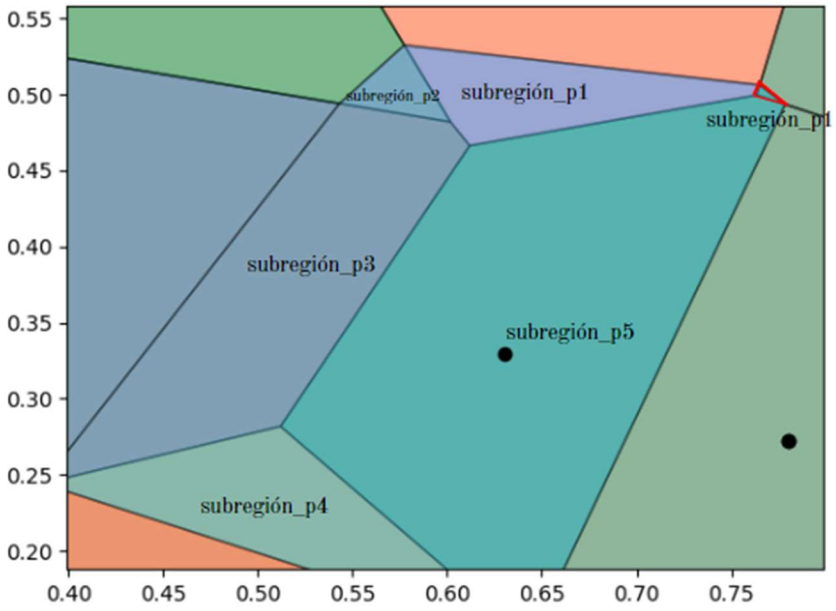


Figura 20. Celda Q, en diagrama Voronoi del conjunto de puntos $P \cup q$. (Autoría propia, 2023)

Se observa que el porcentaje del área de la $subregión_{p_1}$ respecto a la celda que contiene P_1 es cercano a 0, por lo que el valor asignado a esa área será muy pequeño.

3 Metodología

Utilizando 15 datos iniciales correspondientes a las alturas de los puntos en una región, se programará el algoritmo para la interpolación del vecino natural. Una vez con los valores aproximados de los puntos que se agregaron, se graficará el diagrama Voronoi y se utilizará una escala de colores del amarillo al azul donde, a mayor sea la altura de cada punto, la celda correspondiente a ese punto será de un color más cercano al azul, y a menor sea la altura de cada punto, la celda correspondiente a ese punto será de un color más cercano al amarillo.

Para realizar este programa se utilizaron las librerías de numpy, math, random, matplot.pyplot, y de scipy.spatial se importó Voronoi, voronoi_plot_2d y finalmente para realizar la escala de colores se utilizó Color de la librería colour.

Nota: Los nombres de las variables y funciones que se mencionan a continuación son tales como aparecen en los programas.

La función Voronoi de la librería scipy.spatial permite, a partir de un conjunto de puntos, obtener una lista de las coordenadas de todos los vértices en el diagrama, una lista de regiones con listas de los índices de los vértices que conforman cada celda (llamada regions), una lista de los índice de los vértices que indican para cada arista, entre qué vértices se encuentran (vor.ridge_vertices)

y una lista que indica para cada arista, entre qué puntos se encuentran (vor.ridge_points). Para representar los índices de vértices al infinito, se utiliza el índice -1.

Debido a que se busca, como última instancia obtener el área de algunas regiones del diagrama Voronoi, y es posible que esas regiones sean infinitas, se deben calcular los vértices que remplazarán los que son puntos al infinito. Para ello se utilizará la función voronoi_finite_polygons_2d (Virtanen, P. 2015)

El parámetro de esta función, son las listas contenidas el diagrama Voronoi (vor) que se obtuvo mediante la función Voronoi. Los valores de retorno de la función voronoi_finite_polygons_2d serán la lista de regiones con sólo vértices finitos y un arreglo de las posiciones x, y de los vértices (todos los infinitos convertidos a finitos y los restantes). Además de esta función se utilizarán dos funciones, cambia_vértices y área_agujeta_gauss, ambas de autoría propia. Estas serán explicadas más adelante.

Una vez definidas las librerías y las funciones por utilizar, se declaró la lista de coordenadas de los puntos a utilizar y se declaró una lista que contiene los valores de las alturas de cada punto. (alturas_inciales). Después, se obtuvieron la lista regions y vértices de voronoi_finite_polygons_2d.

A partir de esto, se pide al usuario cuántos puntos desea interpolar y se crea un contador para repetir el proceso de interpolación para cada punto. Una vez que el usuario ha ingresado las coordenadas del punto (llamado b en el programa), y se han validado que estén dentro del rango de la región que se está interpolando, se agrega b a la lista de puntos y se calcula el diagrama Voronoi nuevo (vorb). Después se utiliza la función voronoi_finite_polygons_2d con vorb para obtener las listas de regiones y vértices sin puntos al infinito.

Después se buscará aislar la figura que contiene únicamente las subregiones de las cuales queremos calcular el área. Para ello, se realizarán procesos de filtración a la lista de vértices originales (vértices 2) y la lista de vértices nuevos (vértices b2). Para ello se eliminan los duplicados de cada una y se almacenan las listas filtradas en v1 y v2 respectivamente, recordemos que los elementos de estas listas son listas que contienen la posición x y y de cada vértice. A partir de esto, se creará para v1 y v2, dos listas para cada uno, en la primera se agregan los vértices que sólo están en su diagrama Voronoi (ya sea el original o el contiene el punto b), ln o lnb y sus respectivos índices (índicesv o índicesvb).

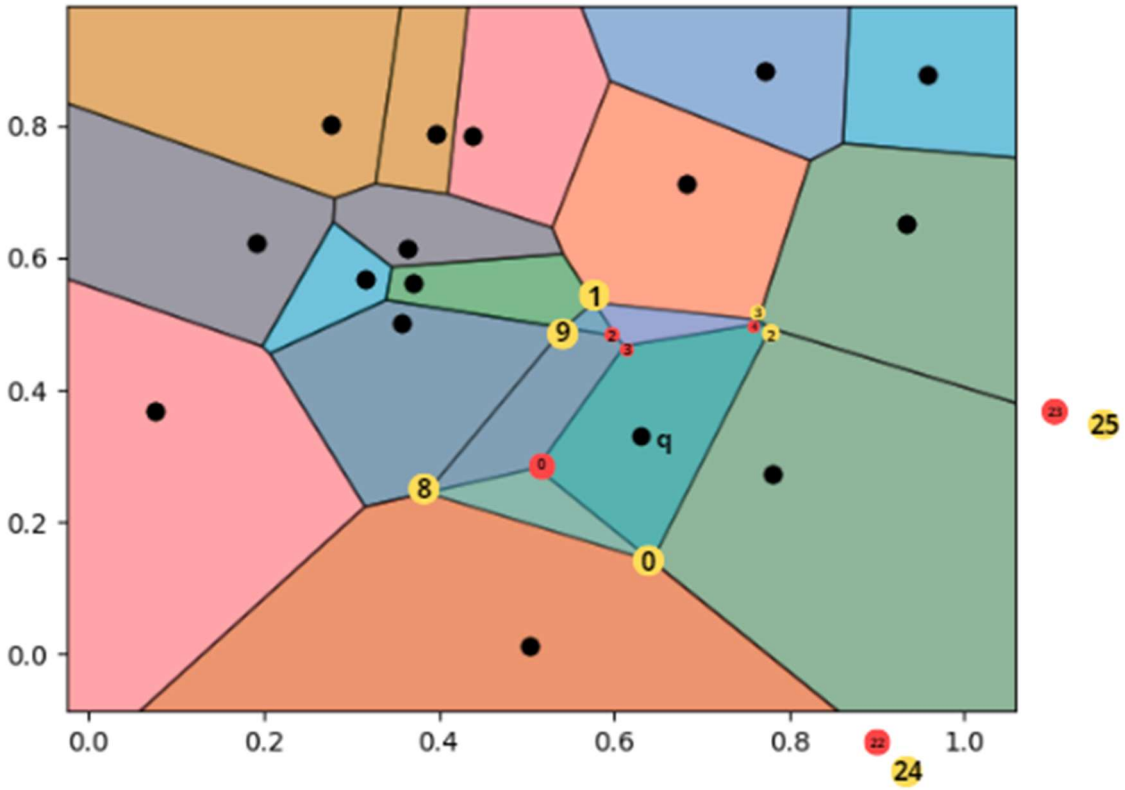


Figura 21. Ejemplo de la posición de los elementos de ln y lnb. Cuando el punto q es agregado al diagrama Voronoi. Nota: los vértices de índice 22, 24, 23 y 25 no son puntos al infinito, se encuentra fuera del rango de la gráfica.(Autoría propia, 2023)

En el ejemplo anterior, índicesv = [0,1,2,3,8,9] (vértices amarillos) e índicesvb= [0, 2,3,4,22,23] (vértices rojos). Es importante recordar que como se trata de dos diagramas Voronoi distintos, los índices del mismo vértice pueden cambiar en ambas listas, pero se mantienen constantes en las listas del mismo diagrama.

Una vez que se tienen los vértices del diagrama Voronoi original filtrados, se van a buscar las regiones del diagrama original que contienen los vértices filtrados. Es decir, las regiones del diagrama original cuyas áreas se buscará calcular más adelante. Para ello, por cada vértice en índicesv se buscan las regiones (regions) que lo contienen y si no están en la lista regiones_filtradas, se agregan (esto para evitar elementos duplicados),

A continuación, se crearán listas, para cada vértice original, de los vértices originales vecinos que tenía en el diagrama original, para ellos se busca cada vértice original en regiones_filtradas, y se añade la lista de los vecinos a la lista vecinos_todos.

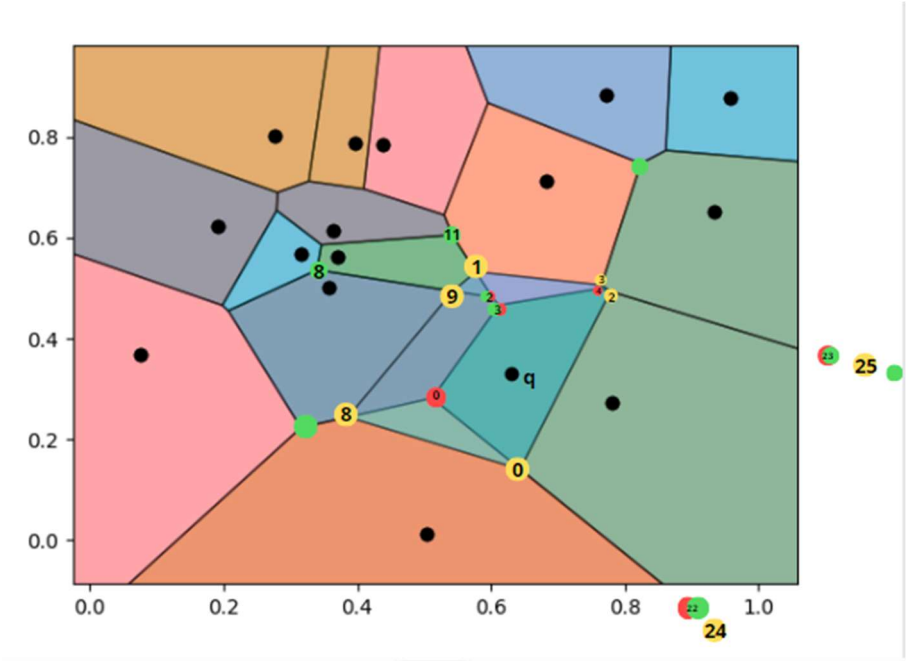


Figura 22. Ejemplo de la posición los vecinos de los vértices originales (en verde) en el diagrama original. Nota: los vértices de índice 22, 24, 23 y 25 no son puntos al infinito, se encuentra fuera del rango de la gráfica.(Autoría propia, 2023)

En este ejemplo los vecinos del vértice 2, en el diagrama original, serán el vértice con índice 8, 3 y 11.

De manera similar, se crea una lista de los vecinos nuevos, con los vecinos originales. El proceso es el siguiente, para cada vértice nuevo, se calcula la distancia a cada vértice original (utilizando la lista filtrada) y se guarda el vértice original más cercano. De esta forma se sabe que existe una arista entre ellos y por lo tanto son vecinos en el diagrama nuevo. Se crea con estos datos la lista verticenuevo_verticeo. Se va a reorganizar esta lista, añadiendo los elementos organizados a verticeo_verticenuevo, de tal manera que cumpla con las siguientes condiciones:

1) Para cada elemento en verticeo_verticenuevo, los vértices originales se encuentran en la posición [0]
2) No se consideran vértices originales que estén fuera de la celda Q.

   Para cerciorarse de la condición 2, basta con validar que el índice de los vértices sea menor que el elemento mayor de ridges_vertices, ya que esta lista contenía vértices al infinito (-1 y por lo tanto el índice de todos los restantes, (que están dentro de la gráfica) son menores al índice mayor.

Finalmente se tienen dos listas:

La lista que contiene para cada vértice original, los índices de los vértices vecinos originales y la lista que contiene para cada vértice original, los índices de los vecinos nuevos.

Vamos a calcular para cada elemento en estas listas, los ángulos que forma cada vértice con cada vecino. De esta forma se obtienen: las listas ángulos_vértices_oco y ángulos_vértices_ocn.

Se tiene entonces, la lista que contiene para cada vértice original los ángulos que forma con sus vecinos originales y la lista que contiene para cada vértice original los ángulos que forma con sus nuevos vecinos.

Si existen dos elementos en la misma lista en ambas listas, implica una colinealidad, por lo que se cambia el vecino original, por el nuevo. Para hacer este cambio, se utiliza la función que definimos como cambia_vértices, cuyos parámetros son la lista de valores a cambiar, la lista que va a cambiar (se utiliza regiones_filtradas), los índices de los vecinos que se busca cambiar y la lista de posiciones x,y de los vértices, se regresa una lista, de los vértices de cada región cambiada.

Por ejemplo, en la figura 22, se obtendría que del vértice original 2, el ángulo que forma con su vecino original es igual al ángulo que forma con su vecino nuevo, por lo que los cambiamos en la nueva lista regiones_finales.

Después se cambian los índices en regiones_filtradas por las posiciones, utilizando la lista vertices2, y se obtiene la lista regiones_filtradas_grandes.

A continuación, se calcula el área de cada región, utilizando la función que se definió como área_agujeta_gauss, que utiliza como parámetro las coordenadas de los vértices de un polígono, y devuelve el área de dicho polígono. Esta función utiliza la fórmula de las agujetas de gauss para calcular el área. De esta forma, se crean dos listas, área_regiones_finales que contiene el área de cada elemento en regiones_finales y área_regiones_finales_grandes, que contiene el área de cada elemento en regiones_finales_grandes. Después se calcula el porcentaje que representa cada subregión de la región grande y se crea la lista porcentaje.

Finalmente, se encuentran los puntos que corresponden a cada región grande y la altura de dichos puntos. Se calcula el resultado de la multiplicación del porcentaje que representa cada subregión de la región grande por el valor de el punto de esa celda. Se suman todos estos valores y se almacena en valor. Después se despliega el mapa interpolado con el valor de cada punto y se colorea cada región de la siguiente manera:

Después de ordenar los valores de las alturas de menor a mayor, guardado en la lista alturas_ordenadas, para cada celda a colorear, se encuentra el punto al que pertenece y el índice de la altura que tiene dicho punto en alturas_ordenadas, sea i dicho índice. Se crea una escala de colores de la cantidad de alturas que se tiene y se guardan en la lista colors, y se aplica el color que tenga el índice i en dicha lista para esa región.

Finalmente se calcula el rango de la gráfica y se despliega.

Además de este programa, que se tituló como “Interpolación de puntos con coordenadas ingresadas por el usuario”, se realizó un programa similar, que utiliza los mismos algoritmos matemáticos y de programación pero que permite al usuario interpolar la cantidad de puntos de su elección, sin ingresar las coordenadas de dichos puntos. Para calcular estas coordenadas, se utilizó la función uniform de la librería random, en el rango del diagrama, a este programa se le tituló “Interpolación de puntos con coordenadas aleatorias”. Así mismo, se crearon 2 programas para representar los

puntos en su estado inicial, en el programa “Voronoi Puntos Iniciales”, se computa el diagrama Voronoi con regiones infinitas y en el programa “Voronoi Puntos Iniciales Coloreados”, se colorea dicho diagrama utilizando la escala de colores descrita. Consultar apéndice.

4 Resultados:

A continuación, se presentan los resultados de la aplicación del programa, para un conjunto de 15 puntos.

Mapeo Topográfico de una región.

Puntos iniciales.

Coordenadas	Alturas Iniciales
[0.19151945 0.62210877]	40.0
[0.43772774 0.78535858]	500.0
[0.77997581 0.27259261]	20.0
[0.27646426 0.80187218]	300.0
[0.95813935 0.87593263]	190.0
[0.35781727 0.50099513]	1990.0
[0.68346294 0.71270203]	250.0
[0.37025075 0.56119619]	2000.0
[0.50308317 0.01376845]	100.0
[0.77282662 0.88264119]	220.0
[0.36488598 0.61539618]	1700.0
[0.07538124 0.36882401]	30.0
[0.9331401 0.65137814]	200.0
[0.39720258 0.78873014]	400.0
[0.31683612 0.56809865]	1750.0

Caso 0: Diagrama Voronoi .

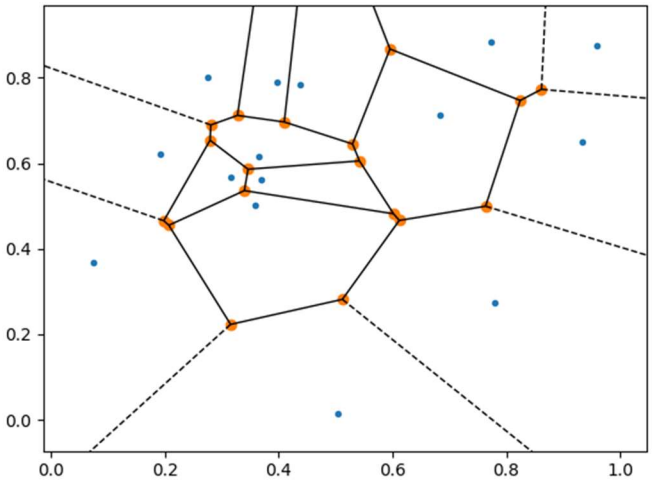


Figura 23. Diagrama Voronoi. Generado mediante el programa Voronoi Puntos Iniciales. (Autoría propia, 2023)

Al utilizar la función Voronoi se obtiene un diagrama con regiones infinitas y vértices.

Caso 1: Se aplica la escala de colores para los puntos iniciales, cuyas alturas son conocidas.

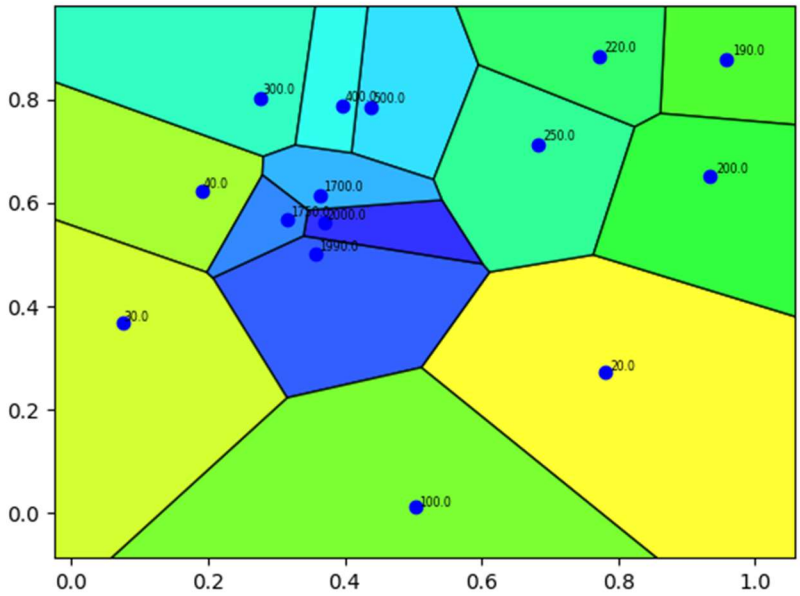


Figura 23. Diagrama Voronoi con escala de colores. 0 puntos interpolados. Generado mediante el programa Voronoi puntos iniciales coloreados (Autoría propia, 2023)

Al aplicar la escala de color, se obtienen regiones azules, cuyos valores de altura son altos, y valores amarillos, cuyos valores de altura son pequeños.

Caso 2: Se aplica el algoritmo de interpolación del vecino natural, para 3 nuevos puntos.

```
ingresa la coordenada y del punto 0  a interpolar, en el rango(-.084,.98)
.33
la altura aproximada en ese punto es,
171.82649549164594
ingresa la coordenada x del punto  1  a interpolar, en el rango (-0.24,1.058)
.29
ingresa la coordenada y del punto 1  a interpolar, en el rango(-.084,.98)
.60
la altura aproximada en ese punto es,
640.6994816979127
ingresa la coordenada x del punto  2  a interpolar, en el rango (-0.24,1.058)
.5
ingresa la coordenada y del punto 2  a interpolar, en el rango(-.084,.98)
.5
la altura aproximada en ese punto es,
1965.8049665353346
alturas ordenadas
18
el mapa interpolado es
```

Figura 24. Resultados desplegados en la consola. (Autoría propia, 2023)

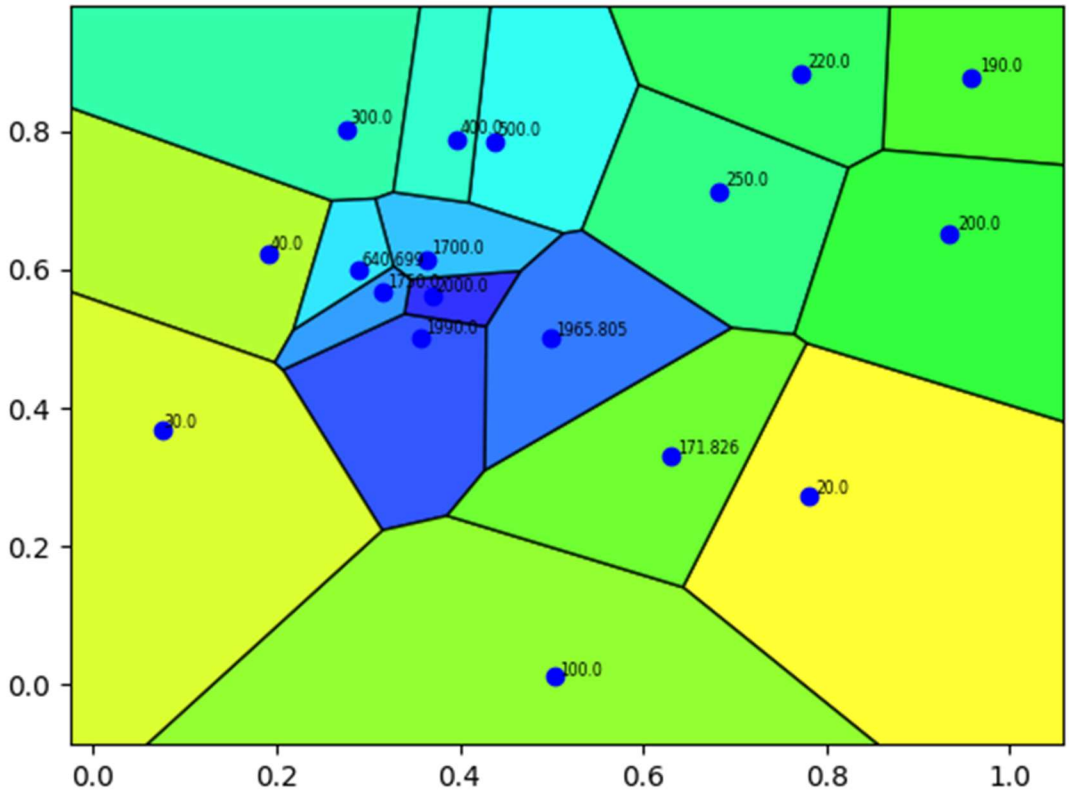


Figura 25. Diagrama Voronoi con 3 puntos interpolados, con escala de color. (Autoría propia, 2023)

Se observan que las áreas de las celdas decrecen y se obtienen las siguientes alturas para los vértices interpolados.

Punto interpolado	Coordenadas (x,)	Aproximación altura	Color
1	(0.63, 0.33)	171.82649549164594	Verde claro
2	(0.29, 0.6)	640.6994816979127	Verde-turquesa
3	(0.5, 0.5)	1965.8049665353346	Celeste

Caso 3: Se aplica el algoritmo de interpolación del vecino natural, para 8 nuevos puntos.

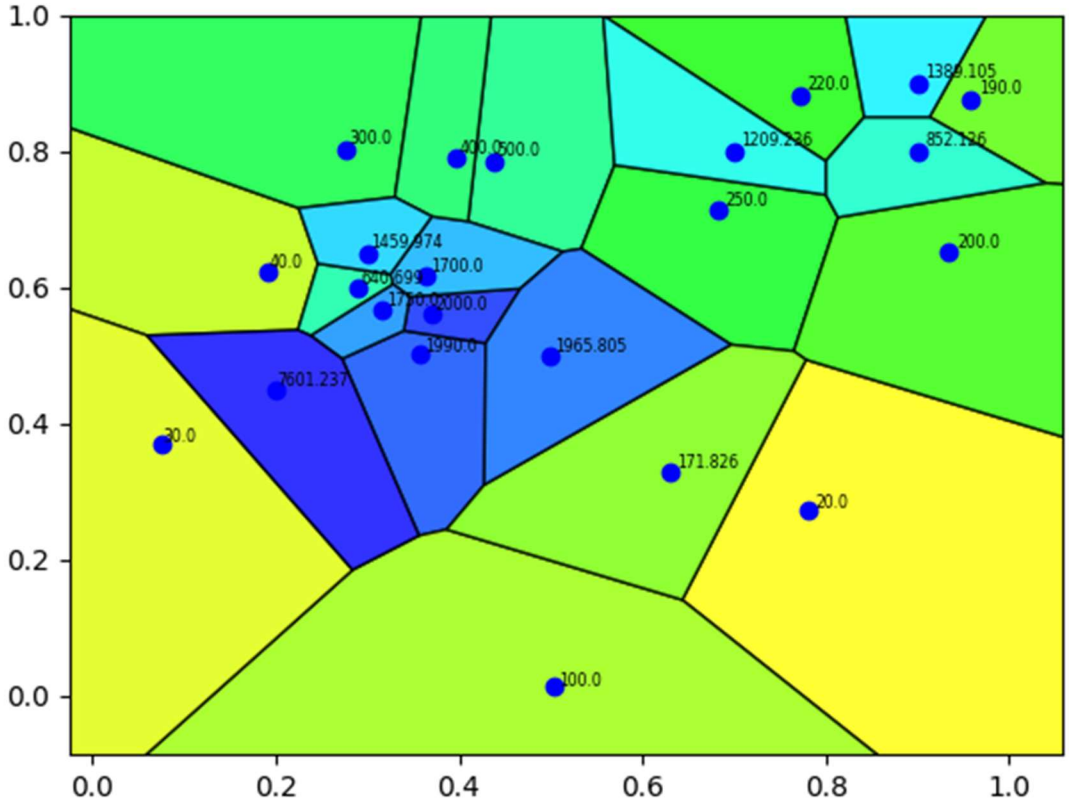


Figura 26. Diagrama Voronoi con 8 puntos interpolados, con escala de color. (Autoría propia, 2023)

Se observa que, al interpolar más puntos, se utilizan más colores en diagrama topográfico, las regiones son de menor área y existe una mayor diversidad de alturas, lo que permite realizar interpolaciones más exactas.

Punto interpolado	Coordenadas (x,)	Aproximación altura	Color
1	(0.63, 0.33)	171.82649549164594	Verde claro
2	(0.29, 0.6)	640.6994816979127	Verde-turquesa
3	(0.5, 0.5)	1965.8049665353346	Celeste
4	(0.9, 0.8)	852.1260360723734	Verde
5	(0.7, 0.8)	1209.2357416152788	Verde claro vibrante
6	(0.3,0.65)	45.659274144169984	Verde
7	(0.9,0.9)	532.783437069757	Azul claro
8	(0.2, 0.45)	75.20573468285998	Verde

Caso 4: Se aplica el algoritmo de interpolación del vecino natural, para 500 nuevos puntos.

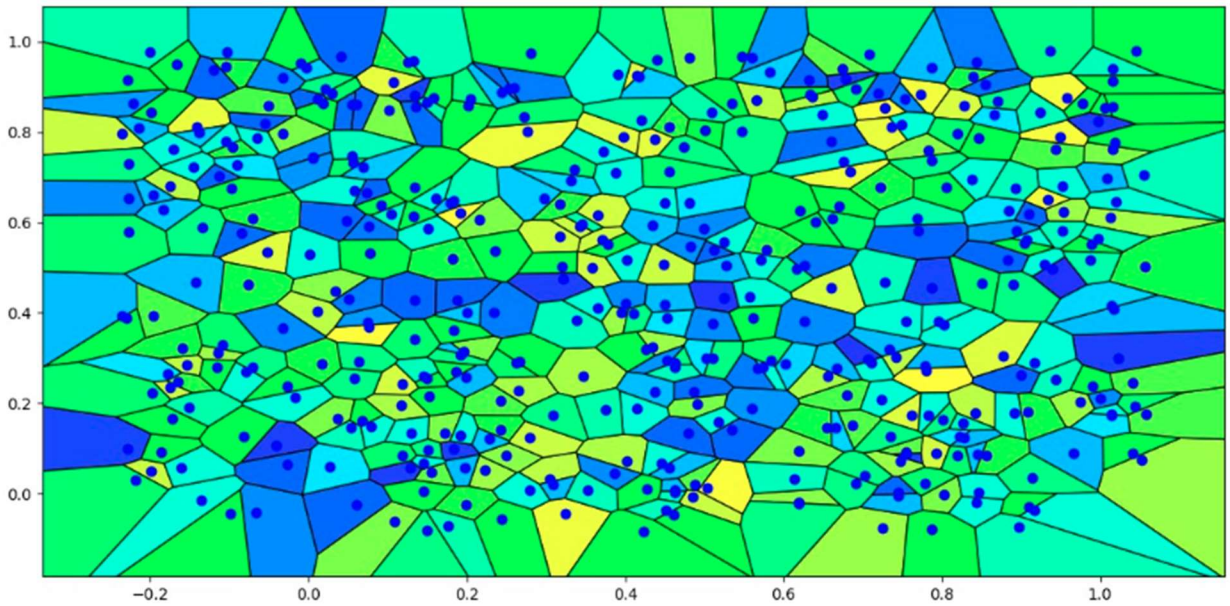


Figura 26. Diagrama Voronoi interpolado, con 500 valores aleatorios interpolados. (Autoría propia, 2023)

Caso 5: Se aplica el algoritmo de interpolación del vecino natural, para 1000 nuevos puntos.

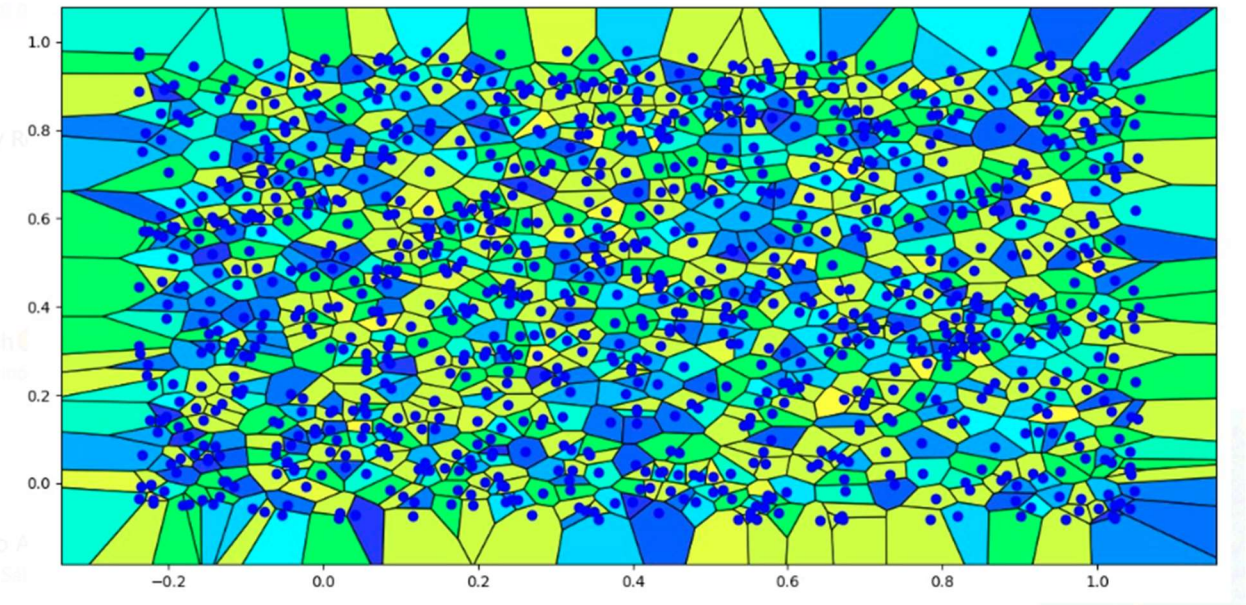


Figura 27. Diagrama Voronoi interpolado, con 1000 valores aleatorios interpolados. (Autoría propia, 2023)

Caso 6: Se aplica el algoritmo de interpolación del vecino natural, para 2000 nuevos puntos.

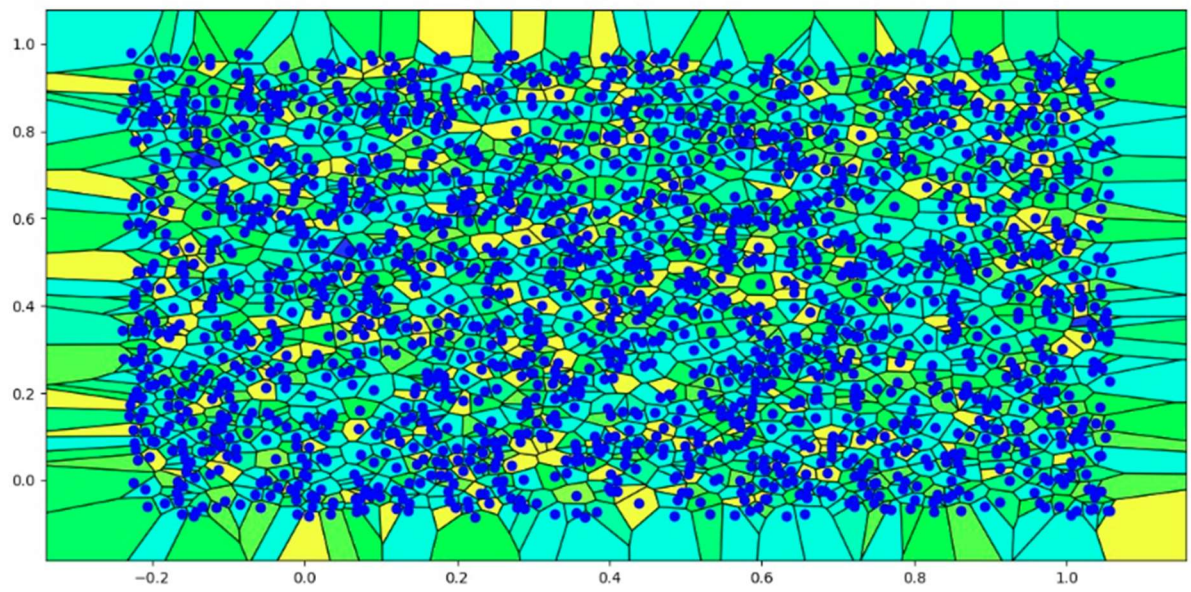


Figura 28. Diagrama Voronoi interpolado, con 2000 valores aleatorios interpolados. (Autoría propia, 2023)

5 Conclusiones

Al utilizar algoritmos para computar triangulaciones Delaunay y/o su dual el diagrama de Voronoi, se puede reducir el tiempo de cómputo exponencialmente.

Es importante utilizar las distintas propiedades de aquello que se busca computar y mantener una perspectiva abierta con respecto las formas en las que se puede hacer. Muchas veces las soluciones más simples son el resultado de ideas que a primera instancia podrían parecer que complican el problema, como lo fue proyectar todos los puntos a un paraboloide en el método *"lifting"*. Así mismo es importante recordar que se puede utilizar ideas de áreas distintas a aquella a la que nuestro problema pertenece. En documento, se exploraron distintos métodos para computar las triangulaciones Delaunay y diagramas Voronoi pero es importante señalar que existen otras triangulaciones y estructuras que son necesarias como el primer paso del algoritmo para la resolución de diversos problemas. La geometría computacional es un área de estudio muy grande y cada vez existen nuevas triangulaciones que se construyen con la finalidad de simplificar y generalizar problemas, por lo que construir una nueva triangulación con características particulares es en muchos casos, el primer paso del algoritmo empleado para la resolución de un problema más grande. Como lo fue en la interpolación de alturas.

Se confirmó, de esta forma, que el algoritmo de interpolación del vecino más cercano es altamente efectivo cuando se trata de la interpolación de conjuntos de datos con distribuciones irregulares y

valores dispersos. Además, se observó que la precisión de este método depende en gran medida de la distribución espacial de los datos y de la elección adecuada de los vecinos más cercanos. Sin embargo, también se identificó una limitación en la capacidad del algoritmo para capturar patrones más complejos y sutiles presentes en ciertos conjuntos de datos.

Cabe señalar que en esta exploración nos enfocamos en algoritmos que son computables para construir triangulaciones Delaunay, sin embargo no determinamos cuál de ellos es más eficaz, esto podría ser de utilidad al momento de decidir cuál emplear, por lo que una de las formas en la que podríamos mejorar esta exploración es al calcular lo que se conoce en programación como la complejidad del programa (o el número de operaciones que tomaría computar cada algoritmo) para determinar cuál es el óptimo.

Es importante señalar que la implementación de este programa, la interpolación de datos utilizando el algoritmo del vecino natural, puede tener aplicaciones en una variedad de campos, incluyendo el procesamiento de imágenes infrarrojas, el procesamiento de señales para suavizar o reconstruir señales faltantes, en la predicción de valores topográficos y ambientales como la dispersión de minerales en alguna región, o en la construcción de modelos tridimensionales del espacio. Cabe notar que debido a que las triangulaciones Delaunay y los diagramas Voronoi pueden ser generalizados a más dimensiones, se pueden utilizar estas generalizaciones en el análisis de datos de mayor complejidad, de ser necesario.

6 Apéndice

En la siguiente carpeta se proporcionan los 4 programas que se utilizan en este proyecto. Archivos.py para su consulta y posible compilación.

Link a Drive:
https://drive.google.com/drive/folders/13XvRFTPSBn4DnQhxSex_ua2aGc99QUH-?usp=sharing

Referencias:

Davies, J. (n.d.). Voronoi airport map. Retrieved March 9, 2023, from https://www.jasondavies.com/maps/voronoi/airports/

Draganov, O – Delaunay Triangulation. (2022). Computing Delaunay complex: Lifting to a paraboloid [Video file]. Retrieved from Computing Delaunay complex: Lifting to a paraboloid [Ondřej Draganov] - YouTube

Karimipour, Farid & Ghandehari, Mehran & Ledoux, Hugo. (2013). Watershed Delineation from the Medial Axis of River Networks. Computers & Geosciences. 59. 132-147. 10.1016/j.cageo.2013.06.004.

Kinderman, P – Convex Hull in 3D (1/5) | Computational Geometry - Lecture 09 [Video file]. Retrieved from Convex Hull in 3D (1/5) | Computational Geometry - Lecture 09 - YouTube

Kinderman, P – Delaunay Triangulation. (2021). Delaunay Triangulation (2/5) | Computational Geometry - Lecture 8 [Video file]. Retrieved from Delaunay Triangulation (2/5) | Computational Geometry - Lecture 08 - YouTube

Lucas, G. (n.d.). A fast algorithm for natural neighbor interpolation. https://gwlucastrig.github.io/TinfourDocs/NaturalNeighborTinfourAlgorithm/index.html

Shah, M – (2023). A Brief Introduction to Computational Geometry [Video file]. A Brief Introduction to Computational Geometry - YouTube

Virtanen, P. (2015). Colorized Voronoi diagram with Scipy, in 2D, including infinite regions. Colorized Voronoi diagram with Scipy, in 2D, including infinite regions · GitHub

Weygaert, Rien & Jones, Bernard & Platen, Erwin & Aragon-Calvo, Miguel. (2009). Geometry and Morphology of the Cosmic Web: Analyzing Spatial Patterns in the Universe. Computing Research Repository - CORR. 3 - 30. 10.1109/ISVD.2009.36.

Este proyecto fue elaborado por la alumna Viviana Carrizales Luna. Doy fe que fue realizado durante los años 2022-2023 y que está terminado. Estoy disponible para cualquier duda o aclaración referente al presente documento o al proceso que tomó en la elaboración de su proyecto.

M.Ed Mario Alberto Cantú Montemayor (35 años como maestro en ITESM)

Profesor de Matemáticas de Bachillerato Internacional

ITESM -PREPA TEC

Campus Eugenio Garza Lagüera

mcantu@tec.mx

(5281) 8155 - 446.