



Universidad Nacional Autónoma de México
Posgrado en Ciencia e Ingeniería de la Computación

Desarrollo de métodos para la medida de propiedades
superficiales de nubes de puntos

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

Gibrán Alfonso Zazueta Cruz

TUTOR PRINCIPAL

Dr. Alfonso Gastélum Strozzi,
Instituto de Ciencias Aplicadas y Tecnología

México, Cd. de Mx., enero 2024



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL

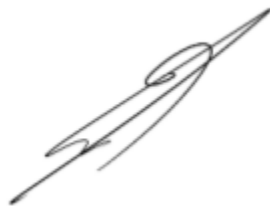
De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado "Desarrollo de métodos para la medida de propiedades superficiales de nubes de puntos", que presenté para obtener el grado de Maestro, es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Programa de Posgrado, citando las fuentes, ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad e los actos de carácter académico administrativo del proceso de titulación/graduación

Atentamente

Gibrán Alfonso Zazueta Cruz, 414107519



Índice

1. Introducción	1
1.1. Nube de puntos 3D	1
1.1.1. Obtención de nubes de puntos por fotogrametría	2
1.1.2. Obtención de nubes de puntos por Escaneo 3D	3
1.1.3. Obtención de métricas sobre nube de puntos	4
1.1.4. Interacción con nube de puntos	5
1.1.5. Procesamiento en paralelo	5
1.2. Impacto arqueológico	5
1.2.1. Análisis de restos óseos con técnicas tradicionales de Arqueología forense	6
1.2.2. Errores en las mediciones	6
1.2.3. Uso de imagenología en Arqueología forense	7
1.3. Restos encontrados en la excavación de Tingambato	8
1.4. Objetivo	9
1.5. Objetivos específicos	9
2. Marco teórico	10
2.1. Discretización del objeto 3D	10
2.2. Manejo eficiente de nubes de puntos	10
2.2.1. Octree	10
2.2.2. Cálculo de vecinos	11
3. ¿Cómo medir sobre superficies? La distancia Geodésica	14
3.1. La geodésica	14
3.2. Distancia Geodésica	16
3.3. Problema geodésico discreto	17
3.3.1. Algoritmos de caminos más cortos	17
3.3.2. La solución Geodésica poliédrica y suave	18
3.3.3. Técnicas basadas en resolver ecuaciones diferenciales	18
3.4. La ecuación eikonal	19
3.4.1. Aproximación de la eikonal como un frente de onda	20
3.4.2. Solución viscosa	22
3.4.3. Aproximación numérica	22
3.5. El Método de Marcha Rápida	23
3.5.1. Algoritmo	23
3.6. Método iterativo Rápido	24
3.7. Distancia geodésica sobre nubes de puntos	25
4. Desarrollo	27
4.1. Selección de puntos	27
4.1.1. Selección con Raycast	27
4.1.2. Implementación con CUDA	29
4.2. Medición de distancia geodésica	30
4.2.1. Método	31
4.2.2. Algoritmo	33

4.2.3. Camino geodésico	36
5. Pruebas y resultados	38
5.1. Resultados algoritmo de selección	38
5.2. Pruebas del algoritmo de distancia geodésica con geometrías conocidas	39
5.2.1. Resultados con plano	40
5.2.2. Resultados Media Esfera	43
5.2.3. Pruebas con otros objetos 3D	44
5.3. Resultados del camino geodésico	45
6. Análisis de resultados	47
6.1. Algoritmo de selección	47
6.2. Algoritmo de distancia	47
6.3. Comparación con software	48
7. Aplicación en arqueología	50
7.1. Pruebas del algoritmo de distancia geodésica con restos óseos encontrados en excavación	50
7.1.1. Resultados de tiempo	52
7.2. Pruebas del algoritmo de distancia geodésica con otros objetos arqueológicos	52
8. Discusión	56
8.1. Limitaciones del método	56
9. Conclusiones	57

Índice de figuras

1.	Diagrama de un punto X proyectándose sobre dos planos imagen con centro C y C' . Los tres se encuentran en el mismo plano epipolar junto con los puntos x y x' y los epipolos e y e'	2
2.	Distancia extrínseca o euclidiana (rojo) y distancia intrínseca o geodésica (azul) en un toroide.	4
3.	Visualización de la nube de puntos del cráneo encontrado en la segunda tumba de Tingambato [26].	8
4.	Agujeros sobre la superficie del cráneo, causados por posible desnutrición del individuo. Visualizado con el software MESHLAB [28].	9
5.	Ejemplo Clave Morton	11
6.	Vecinos por cara, arista y vértice	11
7.	Ordenamiento Z de nodos hijo dentro de un nodo padre.	12
8.	La recta geodésica de un plano es una longitud de arco en una esfera (Textura obtenida de Nasa 3D Resources https://nasa3d.arc.nasa.gov/detail/ear0xuu2)	14
9.	Triedro de Frenet sobre una curva Γ sobre la superficie S	14
10.	Curva local mínima	15
11.	Geodésicas entre dos puntos sobre una variedad de genus 2. Imagen obtenida de [33]	16
12.	Distancia geodésica de un grafo (imagen obtenida de [35])	17
13.	Aproximación discreta de una Superficie suave. Izquierda. distancia más corta con Dijkstra. Derecha, distancia más corta real	18
14.	Curva Γ creciendo en dirección de la normal (vector color rojo).	21
15.	Plantilla de vecinos para actualizar el valor de un nodo	23
16.	Diagrama del algoritmo de marcha rápida, imagen obtenida de [43]	24
17.	Iteraciones del algoritmo FIM. Imagen obtenida de [46]	25
18.	Medición de distancia intrínseca o geodésica utilizando la unión de esferas.	26
19.	Proyección a NDC de un cubo representado como malla de triángulos (izquierda) y nube de puntos (derecha).	27
20.	Rayo lanzado desde la cámara en el proceso de <i>raypicking</i>	28
21.	Medición de distancia mínima de un punto a una recta	28
22.	Diagrama de procesos del método de selección implementado	30
23.	Visualización del octree como Cuadrícula tridimensional. Arriba: nube de puntos de media esfera. Abajo: Nube de puntos de toroide.	32
24.	Diagrama de procesos del método de medición de distancia implementado.	34
25.	Gradiente de color que ejemplifica los colores con los que se visualiza el resultado.	35
26.	Camino geodésico sobre plano	37
27.	Selección de puntos visualizada por el software. Punto amarillo es el seleccionado con clic izquierdo y el punto azul es seleccionado con clic derecho.	39
28.	Objetos creados con software de modelado para pruebas. Arriba: vista ortográfica sobre Z . Abajo: vista frontal en perspectiva.	40
29.	Resultado visual del algoritmo de medición de distancia geodésica en plano de 10×10 con diferentes densidades de puntos.	41
30.	Resultado del algoritmo de medición con marcas de distancia	41
31.	Resultado visual algoritmo de medición de distancia geodésica en media esfera de radio 10.	43

32.	Espiral creada por software de modelado	44
33.	Prueba en toroide	44
34.	Prueba en paraboloides hiperbólicas	45
35.	Camino geodésico sobre plano	46
36.	Camino geodésico sobre espiral	46
37.	Comparación de tiempos del algoritmo de selección respecto a la densidad de puntos.	47
38.	Cambio en el tiempo de procesamiento del plano respecto a la densidad de puntos y la distancia máxima encontrada.	48
39.	Comparación de tiempos respecto a distancia máxima encontrada entre el algoritmo propuesto y los software meshlab y gigaMesh	49
40.	Proceso de trabajo con pieza Arqueológica. Seleccionar punto de inicio. La imagen inferior presenta un acercamiento con mayor iluminación.	50
41.	Proceso de trabajo con pieza Arqueológica. Ejecutar algoritmo de distancia de uno a todos los puntos. Se coloca una estrella para mostrar el punto de inicio.	51
42.	Proceso de trabajo con pieza Arqueológica. Seleccionar punto final del camino geodésico. Se coloca una estrella para mostrar el punto de inicio.	51
43.	Proceso de trabajo con pieza Arqueológica. Selección de zona circular	52
44.	Resultado sobre el Coyote de Ihuatzio	53
45.	Resultado sobre Figura de cerámica excavada en sitio	54
46.	Camino geodésico sobre figura de cerámica (en verde).	55

1. Introducción

El desarrollo técnico y disponibilidad de métodos para recuperar información geométrica de “objetos de la vida real” ha dado paso a la posibilidad de analizar computacionalmente una diversa variedad de piezas y superficies como datos 3D.

Ya sea con métodos de escaneo tridimensional (3D) o fotogrametría, la reconstrucción de objetos ha creado gran interés científico en desarrollar algoritmos que permitan obtener información de los datos reconstruidos. Por ejemplo, información que es comúnmente útil dentro de diversas áreas de la ciencia es la relacionada con métricas morfológicas sobre superficie; como distancia y curvatura.

Un área donde hay gran interés para este tipo de desarrollos es la arqueología. Estas técnicas computacionales facilitan la recolección de datos sobre artefactos arqueológicos e incluso permiten analizar propiedades que no se podrían encontrar de otra forma, ya sea porque no existen los instrumentos de medición adecuados o porque los objetos no pueden ser manipulados.

Un ejemplo de colaboración de este tipo es la cooperación que existe entre el Instituto Nacional de Arqueología e Historia (INAH) Michoacán y el laboratorio BIOCOTLAB del Instituto de Ciencias Aplicadas y Tecnología (ICAT) de la UNAM [1]. Producto de esta colaboración se han reconstruido por medio de fotogrametría y escaneo 3D objetos arqueológicos como figuras de barro, tepalcates, joyas y huesos, desenterrados de excavaciones en sitios arqueológicos.

El propósito de esta recolección de datos 3D es tener la posibilidad de procesar y visualizar esta información por medio del desarrollo de algoritmos y software. Esto da paso a la obtención de desarrollos como el de [2], donde se obtiene un descriptor de forma basado en el gradiente. Este descriptor permite encontrar las zonas de una pieza con un mayor cambio en su curvatura. Esto a su vez, permite a un arqueólogo la obtención de nuevos datos para clasificar y analizar una pieza arqueológica.

Parte importante del desarrollo en [2] es la implementación de un Octree y un método de cálculo de vecinos. Esto permite el procesamiento eficiente de datos de nubes de puntos, particularmente con algoritmos como los llamados ‘libres de mallas’. Además, la implementación se realiza considerando el paradigma de computación en paralelo, utilizando CUDA.

1.1. Nube de puntos 3D

Una nube de puntos es un conjunto de puntos en el espacio 3D que representan una escena u objeto [3]. Métodos para obtener representaciones discretas de superficies en forma de datos 3D, como escáneres 3D, LiDAR (*light detection and ranging*) o fotogrametría, dan como resultado nubes de puntos. En este caso, cada punto representa la localización dentro del sistema coordenado XYZ de la superficie de un objeto real.

Dependiendo el caso de uso, para la reconstrucción de la superficie los puntos obtenidos pueden pasar por un post-procesado que permite recuperar información de nubes de puntos incompletas, con ruido o valores atípicos (*outliers*) o con un muestreo no uniforme. Finalmente, es común obtener una malla triangular, ya sea para visualización o aplicación de algoritmos como el Método de Elemento Finito.

Por otro lado, procesar directamente sobre la nube de puntos permite resultados más exactos debido a que los datos no pasan por un proceso de suavizado o interpolación, además que, evitar la obtención de un mallado triangular permite ahorrar considerable tiempo de procesamiento.

1.1.1. Obtención de nubes de puntos por fotogrametría

La fotogrametría es una técnica que se basa en la geometría multivista para, por medio de 2 o más fotografías distintas de una escena, triangular la posición de los puntos que representan la superficie discretizada dentro de la escena 3D.

A continuación se presenta de forma general, y para el caso mas sencillo de 2 vistas, la resolución al problema de fotogrametría.

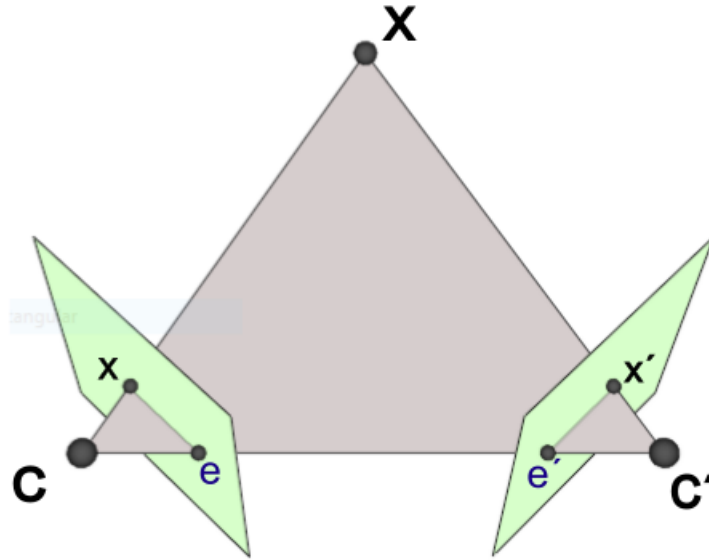


Figura 1: Diagrama de un punto X proyectándose sobre dos planos imagen con centro C y C' . Los tres se encuentran en el mismo plano epipolar junto con los puntos x y x' y los epipolos e y e' .

Dadas dos imágenes con puntos x y x' , se busca encontrar su proyección X en el espacio 3D (Fig. 1). Se siguen los siguientes pasos:

1. **Adquisición de fotografías.** Se deben poder apreciar características compartidas del objeto en ambas imágenes. Al tomar la fotografía es importante tomar en cuenta información como la posición de los centros de las cámaras.
2. **Obtención de la matriz fundamental F .** Geométricamente, la matriz fundamental representa un mapeo del espacio 2D del plano de la imagen al espacio proyectivo. Una propiedad importante de la matriz Fundamental es su correspondencia $x'^T F x = 0$. Con esto es posible encontrar F por medio de características pareadas entre las dos imágenes (al menos 7 características [4]). Esto se logra por medio de calibración, que puede ser:
 - Calibración fotogramétrica: Utilizando un patrón de dimensiones conocidas, como un tablero de ajedrez.
 - Autocalibración: Combinando algoritmos de detección de características como SIFT (*Scale-Invariant Feature Transform*) o SURF (*Speeded-Up Robust Features*) y algoritmos para detección de valores atípicos, como RANSAC (*Random Sample Consensus*).
3. **Encontrar matrices de cámara.** La matriz de cámara está compuesta de los parámetros intrínsecos (distancia focal, punto principal, etc.) y extrínsecos (rotación y translación entre

coordenadas de la cámara y el mundo) de la cámara. Esta matriz permite encontrar un punto 3D X con las relaciones $x = PX$, $x' = P'X$ (Fig. 1).

Las matrices de cámara correspondientes a una matriz F se pueden obtener con las ecuaciones 1 y 2 (obtenidas de [4]), donde e y e' son los epipolos, que son los punto de intersección de la línea que una a los centros de las cámaras con el plano de imagen.

$$P = [I|0] \quad (1)$$

$$P' = [[e'] \times F|e'] \quad (2)$$

4. Triangulación: Como se observa en la Fig. 1, los puntos correspondientes x y x' se intersecan en el espacio proyectivo sobre el punto X . Al contar con las matrices de cámara es posible triangular X con las relaciones $x = PX$, $x' = P'X$.

Cabe mencionar que hay diferentes formas de lograr la triangulación dependiendo, por ejemplo, del conocimiento de la escena, o si las cámaras están calibradas.

Esta misma idea se transmite a la fotogrametría con N-vistas. Para este caso, aunque los algoritmos puedan ser sencillos en implementación, son difíciles de optimizar debido a la gran cantidad de operaciones matriciales que llegan a involucrar, especialmente cuando se procesan miles de fotografías de alta definición.

Afortunadamente existen soluciones de software que realizan el proceso de fotogrametría de manera eficiente. Software como 3DF Zephyr [5] o Qlone [6] permiten obtener de manera ágil una reconstrucción 3D por fotogrametría al recibir como entrada una serie de fotografías de la escena.

Sin embargo, el uso de un buen software de reconstrucción no es suficiente para garantizar buenos resultados. Es igual de importante para reconstruir apropiadamente el objeto 3D considerar cosas como: el uso de la cámara y lente correcto, un ajuste adecuado de la distancia focal y velocidad de obturación, el uso de la correcta iluminación y el enfoque adecuado del objeto desde diferentes ángulos. Todo esto es más fácil de lograr cuando se realiza fotogrametría de “rango corto”, donde a diferencia de la fotogrametría aérea, se tiene un mayor control sobre las condiciones de la toma.

1.1.2. Obtención de nubes de puntos por Escaneo 3D

Cuando se habla de escaneo 3D se hace referencia a diversas tecnologías basadas en el uso de la luz (ópticas) o en contacto, que logran obtener un modelo 3D con la forma de un objeto de la vida real.

Las tecnologías de contacto, como una Máquina de Medición por Coordenadas (CMM, por sus siglas en inglés), son opciones más precisas, pero a la vez más costosas, difíciles de manipular y tardadas. Debido a estas dificultades es que en este trabajo el enfoque será en tecnologías ópticas, cuyo funcionamiento consiste en emitir luz o radiación hacia el objeto para capturar información de su superficie con una cámara o sensor. Entre este tipo de escáner encontramos tecnologías como: LIDAR, escáner de luz estructurada, tomografía computarizada, cámaras tiempo de vuelo (TOF, por sus siglas en inglés), entre otros.

El tipo de escáner óptico mas común es el de luz estructurada. Este consta de una cámara y un emisor de luz y funciona proyectando un patrón de luz sobre la superficie del objeto a escanear (franjas). La luz deformada es capturada por la cámara. Al comparar el patrón proyectado con el

capturado se puede obtener información de profundidad del objeto, que a su vez, permite calcular las coordenadas para formar una nube de puntos.

La principal ventaja del escaneo 3D sobre otras tecnologías de reconstrucción es la capacidad de capturar pequeños detalles sobre la superficie de los objetos con mayor exactitud y precisión, además de no verse tan afectado por las variantes condiciones de luz sobre la pieza a reconstruir.

1.1.3. Obtención de métricas sobre nube de puntos

La obtención de métricas sobre una superficie 3D discretizada es una tarea donde convergen áreas de ciencias de la computación como procesamiento geométrico de datos, visión computacional y graficación. Es una tarea de gran interés ya que permite la obtención de datos geométricos desde los datos 3D. El uso de algoritmos para identificar métricas de una superficie puede representar una gran ayuda para la rápida clasificación de un objeto o incluso para obtener descriptores que no son posibles de obtener mediante la interacción manual con el objeto.

Realizar este procesamiento sobre nubes de puntos representa trabajar con una mayor exactitud, debido a que se procesan los datos directamente obtenidos de la superficie real, sin transformarlos en una malla de triángulos. Para aplicaciones de tiempo real esto también implica una mayor velocidad de procesamiento.

En este trabajo la métrica de interés es la distancia geodésica (Cap. 3) sobre la superficie. Esto puede ser visto como la distancia mínima entre dos puntos siguiendo la curvatura del objeto o como el camino más corto entre dos puntos sin dejar la superficie. A esto también se le llama *distancia intrínseca*. La distancia intrínseca se mide sobre la geometría intrínseca de la superficie. Esto difiere de una distancia extrínseca que sería una distancia euclidiana (Fig. 2).

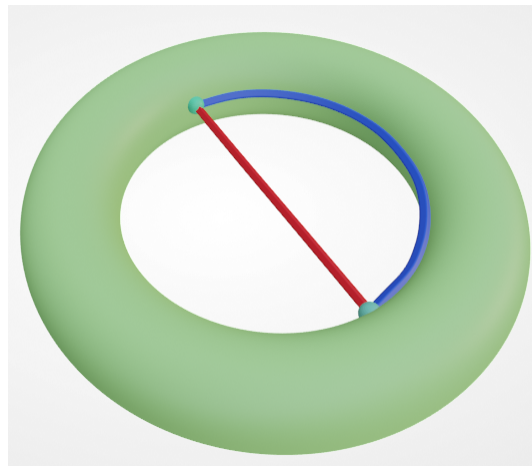


Figura 2: Distancia extrínseca o euclidiana (rojo) y distancia intrínseca o geodésica (azul) en un toroide.

En particular se buscan resolver los siguientes 2 casos de medición de distancia geodésica:

1. Distancia geodésica desde una fuente: Desde un punto de inicio la distancia a todos los demás puntos de la nube
2. Distancia geodésica de punto a punto: Dados dos puntos distintos de la nube calcular la distancia geodésica entre ellos.

Ambos casos entregan información de interés para el análisis de una pieza y, como se verá en el Capítulo 4, resolver el primer caso resuelve el segundo.

1.1.4. Interacción con nube de puntos

Para lograr realizar mediciones específicas sobre un objeto de manera virtual, además del renderizado, es muy útil poder interactuar de alguna manera con la nube de puntos y posibilitar la selección de zonas específicas de la superficie. Esto permite una manera rápida y sencilla de encontrar resultados, incluso para alguien que no está familiarizado con el funcionamiento del algoritmo.

Lo más sencillo para conseguir este tipo de funcionamiento es implementar una interacción utilizando el ratón de la computadora. Un método común de selección de objetos 3D, utilizando el ratón, consiste en lanzar un rayo desde la posición de la cámara con dirección hacia la posición espacial 3D que representa puntero del ratón sobre la ventana de renderizado. A este tipo de métodos se le conoce como *Selección Raycast* o *Raypicking* [7], debido a la técnica de renderizado conocida como Ray Casting que consiste en lanzar un rayo desde la cámara para encontrar intersecciones del rayo con los objetos de la escena y así discriminar que objetos serán renderizados.

Con un método como este se podrá seleccionar y observar de forma intuitiva las zonas o puntos de la nube desde donde se medirán distancias como las mencionadas: de punto a punto o desde una sola fuente.

1.1.5. Procesamiento en paralelo

El procesamiento de nubes de puntos, y otros tipos de datos 3D, suele ser altamente paralelizable. Es por esto que desarrollos como [2] realizan paralelismo por medio de tarjetas gráficas con el uso de tecnologías como CUDA [8].

CUDA o Arquitectura Unificada de Dispositivos de Cómputo, es una plataforma de desarrollo de computo paralelo creada por la empresa NVIDIA para la programación de sus Unidades de Procesamiento Gráfico (GPU's). Su modelo de programación funciona por medio de 'kernels' que son un código a ejecutarse en el dispositivo de forma paralela en n hilos de ejecuciones, donde la cantidad de hilos es definida por el usuario.

Los algoritmos desarrollados en este trabajo buscarán ser paralelizables y aprovechar al máximo la arquitectura de las tarjetas gráficas actuales, buscando así una velocidad de procesamiento casi en tiempo real.

1.2. Impacto arqueológico

La arqueología es un área de las ciencias donde el uso de herramientas de reconstrucción 3D ha mostrado gran utilidad. En [9] se argumenta que dentro de la arqueología el procesamiento de un objeto 3D puede llevar a dos direcciones distintas; su uso para documentación o para análisis. En el caso de tareas de documentación lo menciona como "nuevas técnicas para obtener una sencilla y precisa documentación de hallazgos arqueológicos" [9]. Lo anterior tanto para el almacenamiento, visualización y exposición de la información.

Por otra lado las tareas de análisis conllevan un procesado posterior a la recolección de datos, que permitirá al arqueólogo obtener medidas cuantificables de la pieza. El análisis de objetos arqueológicos por computadora cobra gran importancia no sólo al hacer el trabajo del arqueólogo más sencillo sino por otorgar la capacidad de realizar mediciones que no serían posibles con métodos tradicionales.

En [10] se observa como ejemplo la obtención de parámetros métricos como longitud, altura y el ancho de un grupo de bifaces del periodo paleolítico. De igual manera, se presenta el posterior

cálculo de centro de masa, volumen y área superficial. En este caso, el uso de métodos computacionales eliminó la ambigüedad de las mediciones manuales y permitió obtener parámetros que son difíciles de obtener manualmente, como el centro de masa.

Otro ejemplo interesante es el de [11] donde se reconstruyeron con fotogrametría piezas de mármol encontradas en el fondo del mar que se cree eran transportadas por barcos romanos que encallaron. Los objetos reconstruidos permitieron obtener parámetros como volumen y centro de masa para después posicionarlos a prueba y error sobre modelos reconstruidos de distintos barcos romanos de la época. Después de un análisis hidrostático y de cargas se concluyeron probables características de diseño de los barcos que transportaban este tipo de carga.

1.2.1. Análisis de restos óseos con técnicas tradicionales de Arqueología forense

Una tarea arqueológica común tiene que ver con el análisis de restos óseos. Esta es un área en la que la arqueología llega a converger con las ciencias forenses pues las técnicas de medición, registro y análisis de datos son muy similares. Es de esta relación que surge la disciplina de *Arqueología forense*

Los huesos pueden ser un registro muy duradero y confiable de donde se obtiene información sobre la vida y muerte de la persona. Además de análisis biológicos o químicos, como de DNA o radiocarbono, el tamaño o forma de ciertos huesos puede dar información de la edad y el sexo del humano. Marcas, fracturas o agujeros pueden indicar la causa de la muerte o cambios en el tamaño, forma y densidad del hueso ayudan a identificar enfermedades o padecimientos que pudo sufrir el individuo.

En campo, el arqueólogo experimentado identifica huesos basándose en su experiencia visual e incluso táctil (existe el ejercicio de palpar tus propios huesos mientras sostienes una pieza, para imaginarte como el hueso a identificar “encajaría” en tu propio cuerpo. [12]). Dependiendo del estado de los huesos, el antropólogo forense con su conocimiento es capaz de conjeturar si los huesos son humanos o no, así como identificar edad, sexo y el posible contexto y causa de la muerte.

Ya en el laboratorio se busca tener un registro cuantificable de los restos óseos excavados. Las técnicas tradicionales de medición que han sido empleadas por siglos consisten en utilizar instrumentos de medición como: calibrador deslizante, calibrador de extensión, cinta métrica, regla, antropómetro, tabla osteométrica, mandibulómetro, goniómetro, entre otros.

Las mediciones se realizan con la ayuda de manuales [13][14][15], como guía, aunque la experiencia del arqueólogo también juega un papel importante en la recolección de datos.

1.2.2. Errores en las mediciones

En [16] se realizó un análisis de error en medidas osteométricas del esqueleto humano realizadas por métodos manuales. Se pide a participantes con diferentes niveles de experiencia realizar mediciones de acuerdo al manual *Data Collection Procedures for Forensic Skeletal Material de Moore-Jansen* [17].

Los 3 factores de error se detallan de la siguiente manera:

- **Error de Medición Técnico relativo (TEM)** Variabilidad entre observadores al tomar la misma medida
- **ANOVA de mediciones repetitivas** Análisis de la varianza de la repetibilidad de resultados obtenidos por la misma persona.

- **Índice de Error Escalado (SEI).** Error intraobservador, determina si la variabilidad de medición realizada por la misma persona es constante entre las 4 rondas de mediciones

Entre los resultados se puede destacar que se encontraron dentro del manual casos donde la definición de la medida no era clara y otros donde los puntos de referencia eran poco claros e inconsistentes.

Estos casos junto con mediciones con altos valores TEM y SEI y significativas diferencias en el ANOVA de mediciones repetidas, fueron eliminadas para la segunda revisión del manual [15]. En esta edición se pueden consultar los valores de error obtenidos por todas las mediciones y pruebas.

Aunque eliminar mediciones no confiables evita errores, de alguna manera se reduce la cantidad de datos estadísticos que pueden ser de utilidad al extraer información de una pieza. Esto a su vez elimina factores importantes de análisis que pueden ayudar a clasificar o analizar dicho objeto forense.

1.2.3. Uso de imagenología en Arqueología forense

La imageología moderna ha creado nuevas posibilidades de medición y análisis de huesos. En [18] y [19] se destacan los siguientes tipos de imágenes utilizadas en arqueología

1. Fotografía: Útil para registrar el proceso de recolección de los huesos y el contexto de su hallazgo. La facilidad de la obtención de la fotografía ayuda a tener el registro en todo momento del estado de la pieza.
2. Radiografía: La radiografía ayuda a encontrar características útiles dentro de la estructura interna del hueso. Por ejemplo, para estimar la edad biológica en los restos de un subadulto se utiliza la radiografía para evaluar el desarrollo de los huesos y la formación dental dentro del cráneo[18].
3. Tomografía computarizada: Una desventaja de la radiografía es que el objeto 3D es proyectado a una imagen 2D. La Tomografía computarizada (TC) consiste en una serie de radiografías que son combinadas para crear una reconstrucción 3D. Es por esto que la TC permite lograr el mismo análisis que con la radiografía, pero con una mayor claridad al considerar la posición espacial de las zonas de interés dentro de la pieza. Sin embargo, su uso en arqueología es limitado debido a su poca accesibilidad y alto costo.
4. Resonancia magnética: La resonancia magnética (RM) tiene un uso menor a la TC dentro de la arqueología forense debido a que no provee el detalle suficiente para analizar huesos, sin embargo se han documentado casos positivos de su uso[20][21].
5. Escaneo de superficie: El escaneo 3D es una opción menos costosa para obtener un modelo 3D. Su uso está en auge, principalmente para registro y replicación de huesos (con impresión 3D) para exhibición [22].
6. Fotogrametría. La fotogrametría es otra técnica moderna y más económica de obtener modelos 3D. A diferencia del escaneo, la fotogrametría requiere de un especialista con experiencia para obtener buenos resultados, tal vez es por esto que su uso no ha sido a tan gran escala. El uso de una cámara de buena calidad permite recuperar de los objetos texturas de gran calidad, por lo que el uso hasta ahora dentro de la arqueología ha sido de documentación y visualización de piezas, aunque el análisis 3D es posible, igual que en otros métodos de captura 3D.

1.3. Restos encontrados en la excavación de Tingambato

La zona arqueológica de Tingambato se localiza en el municipio de Santiago Tingambato en Michoacán, México. El primer registro de la existencia de esta zona data de 1842, aunque por años la relevancia de este asentamiento en el México Prehispánico no fue estudiada.

En [23][24] se presenta un análisis, basado en estudios recientes, que propone una periodificación de Tingambato. Con la ayuda de pruebas de radiocarbono y análisis geofísico y arquitectónico de la zona, realizado por tecnología LiDAR, se teoriza que esta zona fue habitada aproximadamente desde el año 0 hasta alrededor del 600 DC.

En 1978 se realizó una primera excavación en Tingambato, con el registro de una tumba [25] donde se localizaron restos óseos y objetos ornamentales.

En 2011 se encontró una segunda tumba con los restos completos de un solo individuo, que al día de hoy es referido como la ‘princesa guerrera’ [26]. De estos restos se han obtenido diversas superficies de huesos. Por ejemplo, del cráneo se ha obtenido una nube de puntos por medio de escaneo 3D. Para este proceso se utilizó el escáner de luz estructurada Revopoint MINI que hace uso del software Revo Scan 5 [27] para reconstruir la nube de puntos. En la Fig. 3 se observan los restos del cráneo visualizados con el software desarrollado en [2].



Figura 3: Visualización de la nube de puntos del cráneo encontrado en la segunda tumba de Tingambato [26].

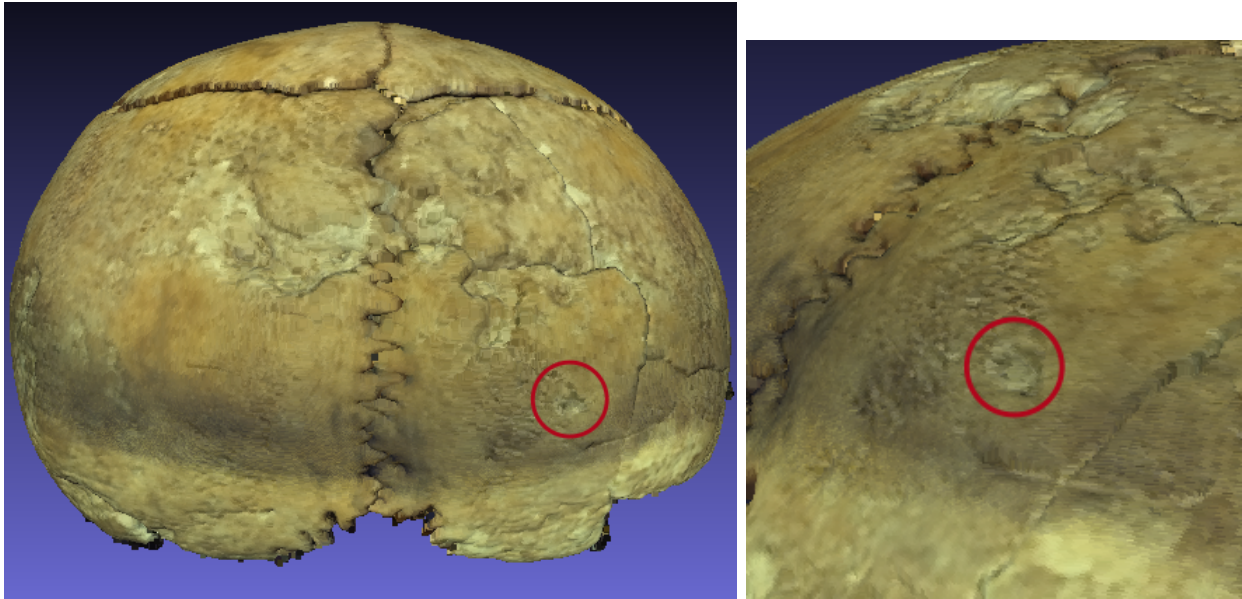


Figura 4: Agujeros sobre la superficie del cráneo, causados por posible desnutrición del individuo. Visualizado con el software MESHLAB [28].

En el cráneo de la princesa guerrera se pueden observar orificios (Fig. 4). Algunos de estos agujeros son resultado de la exposición del cráneo al ambiente, otros, al ser analizados por expertos del INAH en arqueología forense son identificados como posibles rasgos de una enfermedad común en la antigua ciudad de Tingambato. Sin embargo, debido al pequeño tamaño de los hundimientos sobre la superficie es complicado realizar mediciones morfológicas precisas con métodos manuales.

1.4. Objetivo

Partiendo del octree y el método de cálculo de vecinos implementado en [2] se pretende diseñar e implementar un algoritmo para medición de distancia geodésica sobre nube de puntos. La paralelización de los métodos buscará un procesamiento eficaz.

La implementación de los algoritmos se pondrá en uso al analizar restos óseos humanos encontrados en el sitio de Tingambato, donde los métodos arqueológicos existentes para medición de huesos aún tienen espacio para mejorar en precisión y variedad.

1.5. Objetivos específicos

- Desarrollar herramientas que permitan a un especialista interactuar con un objeto 3D para seleccionar regiones de interés.
- Desarrollar un método de medición de distancia geodésica sobre nube de puntos.
- Probar los algoritmos desarrollados sobre huesos obtenidos en excavación para comprobar la utilidad del método.

2. Marco teórico

2.1. Discretización del objeto 3D

Una manera popular de obtener métricas es operar sobre una malla 3D o *mesh*. Normalmente los métodos dentro de este enfoque aprovechan que se conocen las vecindades entre nodos para definir operadores matemáticos, como la derivada. A su vez estos operadores permiten construir ecuaciones diferenciales que representen fenómenos físicos.

Otra aproximación a este problema es trabajar directamente sobre la nube puntos. A este tipo de métodos se les conoce como ‘Métodos Libres de Malla’ o en inglés ‘Mesh-free’.

Los métodos libres de malla modelan el comportamiento del sistema mediante ecuaciones que no consideran la geometría predefinida para la discretización del dominio. Estos métodos son útiles para simulaciones computacionalmente costosas.

2.2. Manejo eficiente de nubes de puntos

Procesar datos con una técnica libre de mallas tiene sus dificultades, más que nada, porque no hay una relación directa entre los puntos de la nube posicionados en el espacio. Es por ello que se necesitan estructuras de datos eficientes que permiten resolver problemas como el de la búsqueda de vecinos más cercanos.

En este trabajo se retoma el trabajo de [29][2] donde se desarrolló un sistema de alto rendimiento para simulaciones físicas computacionales con el método libre de mallas llamado Hidrodinámica de Partículas Suavizadas (SPH, por sus siglas en inglés) [30]. Para ello se desarrolló una estructura de datos tipo Octree dinámico donde se almacenan las partículas y un método de cálculo de vecinos, que es una tarea elemental a resolver en los sistemas libres de mallas.

2.2.1. Octree

El octree es una estructura de datos de tipo árbol que divide espacialmente el espacio volumétrico 3D recursivamente en 8 cuadrantes, es decir, 8 nodos hijos. El octree cuenta con una profundidad que es el número máximo de subdivisiones realizadas.

En [29] el octree está diseñado como un octree lineal, completo e indexado, con una profundidad que se puede especificar del 1 al 10. La navegación del octree se realiza con la técnica de la clave Morton [31][32] que es almacenada en un dato de 32 bits. La clave Morton es única para cada nodo (pero no para cada partícula) y está referenciada a la posición espacial del nodo, de la siguiente manera:

Obtención de la clave Morton

- Se toman los 2 bits más significativos para una etiqueta que indica si el nodo tiene partículas y los 30 restantes para el índice del nodo.
- La etiqueta se llena con 00_2 si el nodo está vacío o 01_2 si contiene al menos 1 partícula.
- Para el índice primero se calcula la posición normalizada del nodo respecto al volumen total del octree. Para la posición n_x, n_y, n_z representada por un valor tipo double, se normaliza como:

$$\bar{n}_x = \frac{n_x}{oct_{width}} \quad \bar{n}_y = \frac{n_y}{oct_{height}} \quad \bar{n}_z = \frac{n_z}{oct_{depth}}$$

- Posteriormente se convierte el double a un valor entero de 10 bits. Para esto se realiza un corrimiento a la derecha multiplicando por $400_{16} = 1000000000_2$

$$\bar{n}_{xi} = (\bar{n}_x)400_{16} \quad \bar{n}_{yi} = (\bar{n}_y)400_{16} \quad \bar{n}_{zi} = (\bar{n}_z)400_{16}$$

- Finalmente se forma el índice de la clave Morton intercalando los 10 bits de los 3 enteros, comenzando por los bits más significativos. Esto se muestra en la Fig. 5.

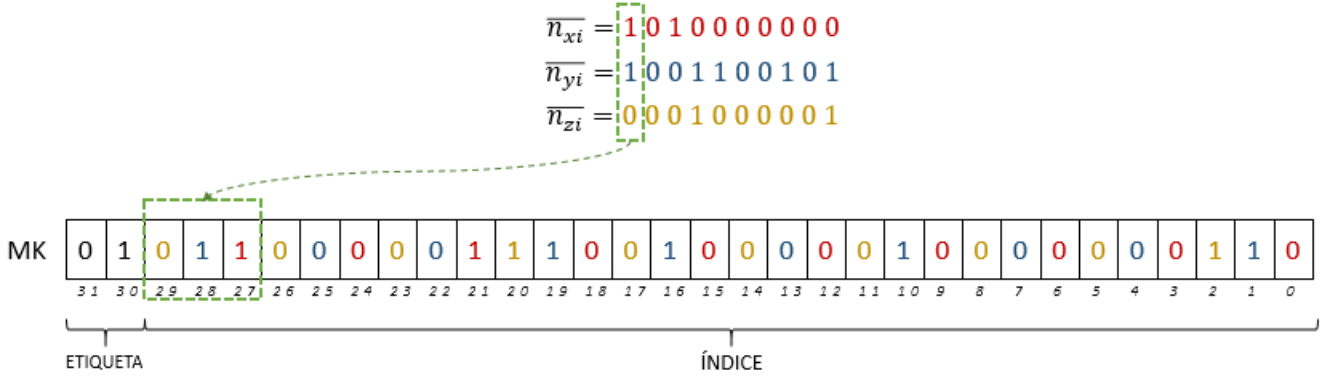


Figura 5: Ejemplo Clave Morton

2.2.2. Cálculo de vecinos

Respecto al cálculo de vecinos, en [29] se aprovecha la clave Morton (MK) para encontrar los 26 nodos vecinos a un nodo del árbol utilizando operaciones binarias.

La vecindad radial del nodo se divide en tres grupos, vecinos por cara, arista y vertice (Fig. 6)

$$V_{cara} = \{V_R, V_L, V_U, V_D, V_F, V_B\}$$

$$V_{arista} = \{V_{RU}, V_{RD}, V_{RF}, V_{RB}, V_{LU}, V_{LD}, V_{LF}, V_{LB}, V_{UF}, V_{UB}, V_{DF}, V_{DB}\}$$

$$V_{vertice} = \{V_{RUF}, V_{RUB}, V_{RDF}, V_{LUF}, V_{LUB}, V_{LDF}, V_{LDB}\}$$

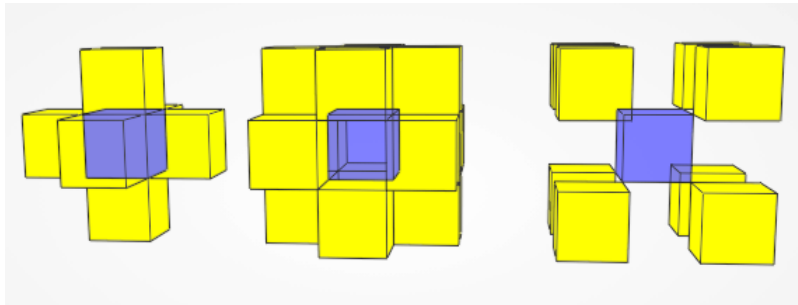


Figura 6: Vecinos por cara, arista y vértice

Para entender el método hay que analizar el ordenamiento espacial de los nodos respecto a la clave Morton, también llamada indexado Order Z. Considerando los nodos hijos de un nodo padre, las “Morton key” seguirán este ordenamiento en Z, como se muestra en la Fig. 7.

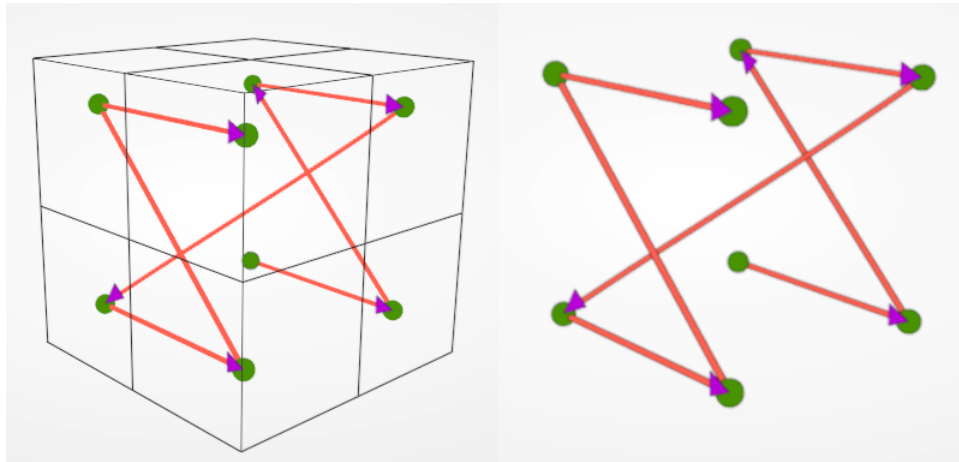


Figura 7: Ordenamiento Z de nodos hijo dentro de un nodo padre.

Donde el nodo en la esquina posterior, inferior, izquierda es el primero, respecto al ordenamiento y los nodos están ordenados del 0 (000_2) al 7 (111_2).

Entonces, para encontrar los vecinos se utilizan operaciones binarias. Primero, si se consideran los vecinos por cara:

Vecino derecho-izquierdo

Se multiplica la MK del nodo actual por 001_2 , existen 2 posibles resultados

- a) Si el resultado es 000_2 el nodo actual se encuentra del lado izquierdo de su nodo padre.
- b) Si el resultado es 001_2 el nodo actual se encuentra del lado derecho del nodo padre.

Ahora, para encontrar el vecino derecho:

- Si a), el vecino se encuentra a la derecha del nodo padre, el vecino se obtiene con la operación XOR entre la MK original y 001_2
- Si b), el vecino puede existir si el nodo padre tiene vecino a la derecha. Si existe, el vecino está a la izquierda del nuevo nodo padre encontrado a la derecha, el resultado se obtiene con XOR entre la nueva MK y 001_2 .

Para encontrar el vecino izquierdo:

- Si a), el vecino puede existir si el nodo padre tiene vecino a la izquierda. Si existe, el vecino está a la derecha del nuevo nodo padre encontrado a la izquierda, el resultado se obtiene con XOR entre la nueva MK y 001_2 .
- Si b), el vecino se encuentra en el nodo padre y se obtiene con la operación XOR entre la MK original y 001_2 .

Vecino arriba-abajo

Primero, se realiza la operación AND entre la MK y 010_2 . Si el resultado es 000_2 el nodo se encuentra en la parte inferior, si es 010_2 en la superior.

Las operaciones para encontrar los vecinos de arriba-abajo siguen la misma lógica ya mostrada.

Vecino frontal-posterior

Finalmente, para el último caso se realiza la operación AND entre la MK y 100_2 . Si el resultado es 000_2 el nodo se encuentra en la parte posterior, si es 100_2 el nodo se encuentra en la parte frontal.

Las operaciones para encontrar los vecinos frontal-posterior siguen la misma lógica ya mostrada.

3. ¿Cómo medir sobre superficies? La distancia Geodésica

3.1. La geodésica

En el espacio continuo la geodésica es una generalización de la recta euclidiana para espacios curvos. Por ello, se asume que esta debe cumplir con propiedades de la recta euclidiana, como son: el ser la curva más “recta” o directa o el ser la curva más corta. A continuación se explican estas propiedades en el contexto de la geodésica.

Rectitud

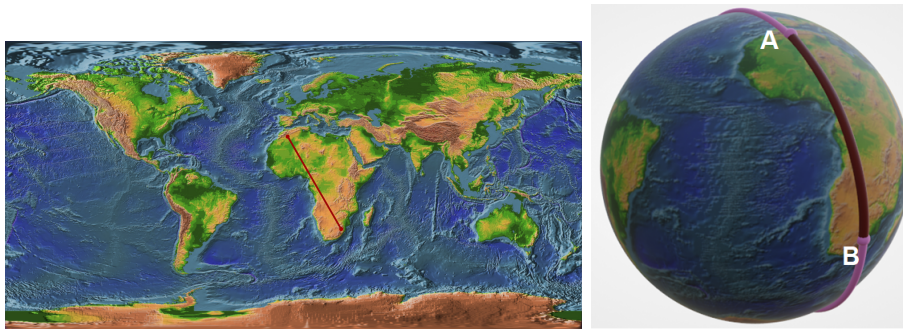


Figura 8: La recta geodésica de un plano es una longitud de arco en una esfera (Textura obtenida de Nasa 3D Resources <https://nasa3d.arc.nasa.gov/detail/ear0xuu2>)

Considerando que una línea recta geoméricamente no tiene curvatura y dinámicamente no tiene aceleración, se puede encontrar una condición similar para la curva geodésica.

Tomando en cuenta el triedro de Frenet para una curva Γ sobre la superficie $S \in R^3$ como la que se visualiza en la Fig. 9.

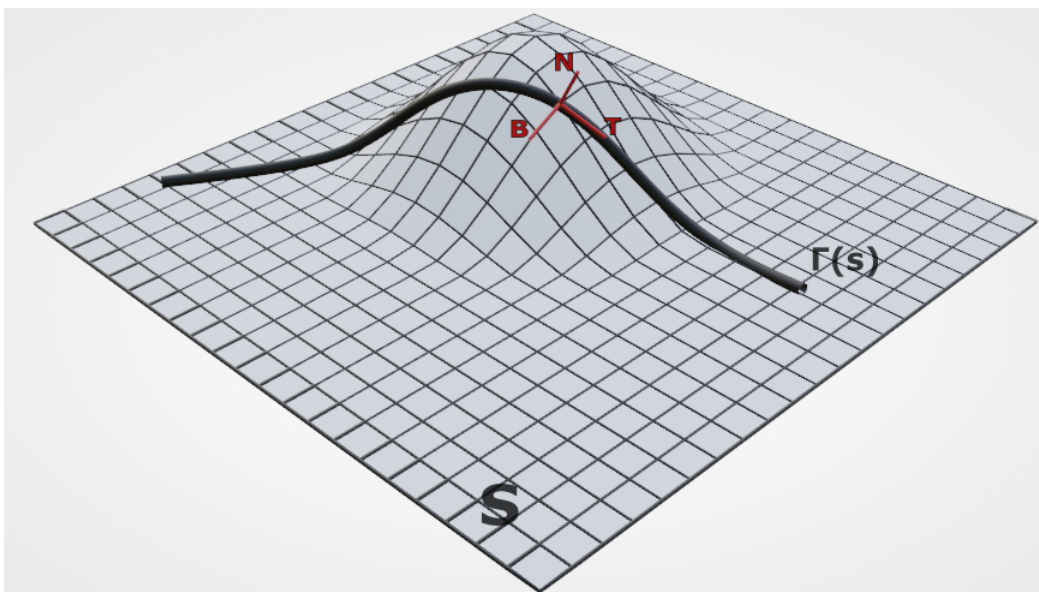


Figura 9: Triedro de Frenet sobre una curva Γ sobre la superficie S

En las ecuaciones 3 y 4 se muestran las curvaturas de la curva Γ con vector tangente T , normal N y binormal $B = T \times N$.

$$k_n = \left\langle N, \frac{dT}{ds} \right\rangle \quad (3)$$

$$k_g = \left\langle B, \frac{dT}{ds} \right\rangle \quad (4)$$

donde \langle, \rangle es el producto punto y $\frac{dT}{ds}$ el cambio de la tangente sobre la superficie .

Al moverse sobre la curva Γ , se sigue a la curvatura normal k_n y cualquier cambio sobre esta dirección será una curvatura geodésica k_g . Una forma sencilla de entenderlo es imaginar que se conduce un auto siguiendo la curva sobre la superficie. En este caso, cualquier giro del volante ocasionará una curvatura geodésica.

La curvatura normal es inherente a la superficie S por lo que la curva geodésica puede tener un $k_n \neq 0$. Cuando una curva es geodésica, el vector tangente es paralelo a la curva (como en la fig. 9), esto es indicativo de que se esta siguiendo el camino mas corto. Sin embargo, al existir curvatura geodésica el triedro rota y el vector tangente deja de ser paralelo.

De esta manera encontramos un símil de rectitud, ya que al igual que esperamos que una recta euclidiana no tenga curvatura, podemos esperar que la curva geodésica entre dos puntos tenga una curvatura geodésica igual $k_g = 0$.

Curva más corta

De nuevo tomamos el caso de la recta euclidiana. Si miramos a la recta localmente, podemos decir que, dados 2 puntos de su vecindario local, la distancia sobre la recta es la distancia mínima entre ellos. Es decir, la recta es un mínimo local de distancia.

Si llevamos este pensamiento a una variedad resulta en lo mismo.

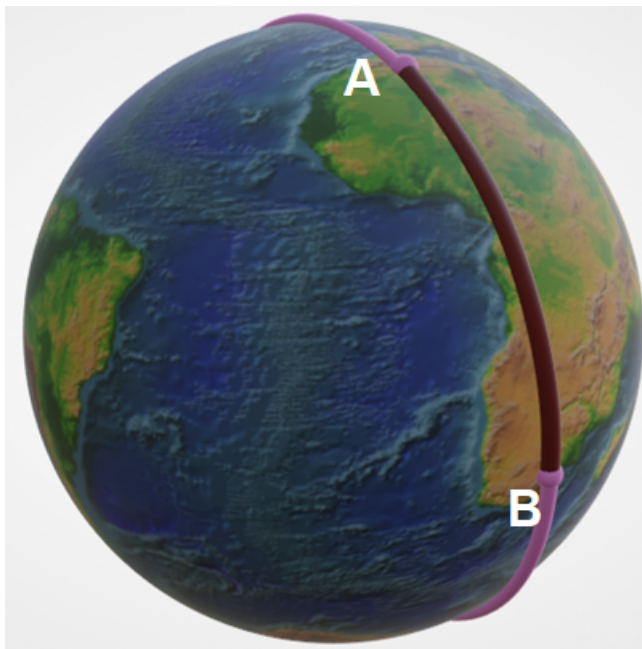


Figura 10: Curva local mínima

De la figura 10, se tiene una longitud de arco en rojo que por un lado es localmente mínima, pero que además, permite asumir con facilidad que también la distancia globalmente más corta entre A y B. Al ser la más corta global es válido asumirla como una geodésica.

Por otro lado, la curva rosa también cumple con tener mínima distancia local por lo que también es una geodésica aunque claramente no es globalmente más corta.

Otro ejemplo interesante es el del toroide doble de la Fig. 11. Aquí se ven 3 geodésicas cuyo trazo se puede imaginar retomando la idea de rectitud. Si se camina por la superficie comenzando de A y solo siguiendo la curvatura normal de la superficie (sin cambiar de dirección) se llegará en los 3 casos al mismo punto.

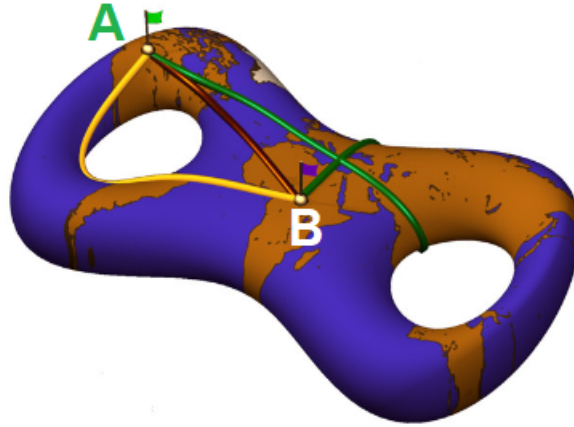


Figura 11: Geodésicas entre dos puntos sobre una variedad de genus 2. Imagen obtenida de [33]

Como se puede ver en estos ejemplos la idea de distancia mínima local nos lleva a afirmar que para una variedad:

- La distancia global mínima no es igual a la geodésica.
- Entre dos puntos sobre la superficie pueden existir múltiples geodésicas.

3.2. Distancia Geodésica

Tomando en cuenta lo explicado en las secciones anteriores llegamos a una definición de distancia geodésica, descrita como; la geodésica mínima entre dos puntos, o lo que es igual, el camino globalmente más corto entre ellos.

Definición

La distancia geodésica d_g entre dos puntos p, q de una variedad V es:

$$d_g(p, q) = \inf_{\Gamma[p, q] \rightarrow V} L[\Gamma] \quad (5)$$

donde L es una función que asigna un valor real a la longitud de la curva Γ .

Esto se interpreta como que d_g es el ínfimo de la longitud de todas las curvas geodésicas que van a p a q .

3.3. Problema geodésico discreto

3.3.1. Algoritmos de caminos más cortos

Como ya se hizo alusión, otra forma de plantear el problema de la geodésica es considerar el camino más corto. Para resolver este problema, un algoritmo muy famoso en ciencias de la computación es el algoritmo de Dijkstra [34], el cual procesa superficies discretizadas como grafos en $O(n \log(n))$ donde n es el número de nodos del grafo. Por ejemplo, uno de los muchos posibles usos del algoritmo es la resolución de un laberinto (Fig. 12).

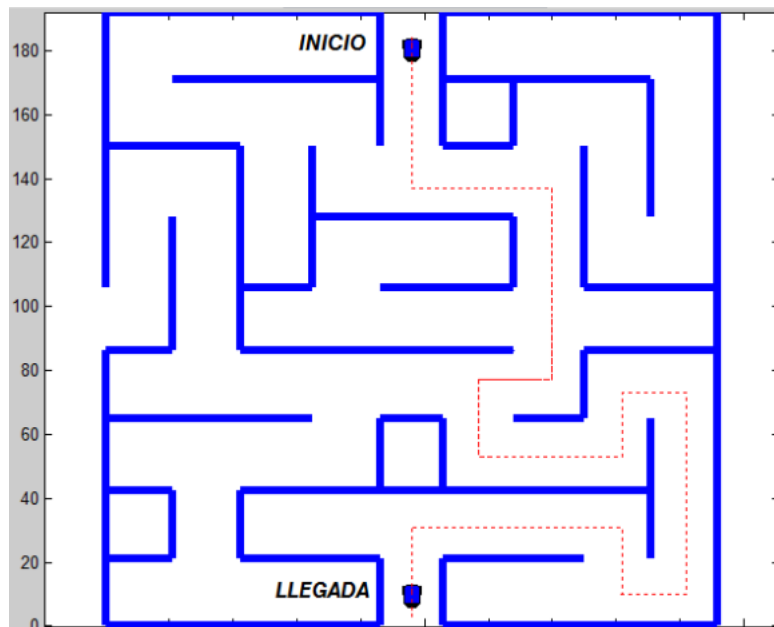


Figura 12: Distancia geodésica de un grafo (imagen obtenida de [35])

En este caso la geometría de la superficie del laberinto está dada por los caminos y obstáculos. Dentro de esta superficie la resolución (única) del laberinto entre un punto y el otro cumple con ser una distancia geodésica. Al ser este el camino más corto sobre esta superficie discretizada se puede considerar que la solución del laberinto es una distancia geodésica.

Ahora ¿qué sucede si en un lugar de un laberinto o un grafo nuestra superficie discreta es una aproximación a una superficie real o suave? Un posible problema con Dijkstra se puede observar en la Fig.13

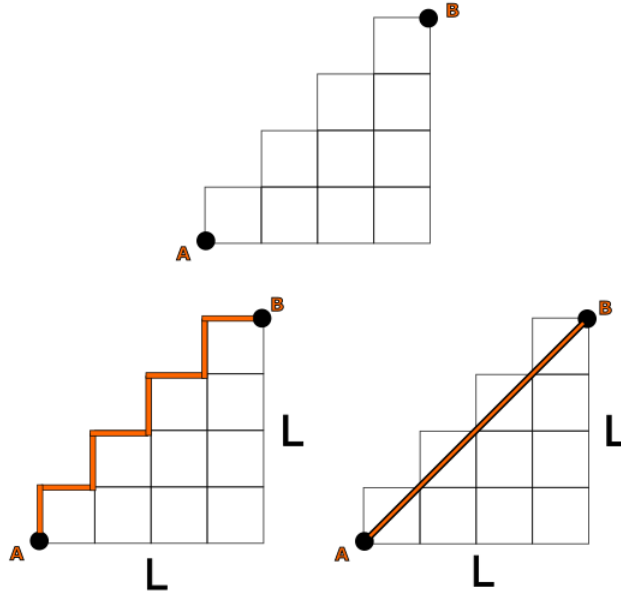


Figura 13: Aproximación discreta de una Superficie suave. Izquierda. distancia más corta con Dijkstra. Derecha, distancia más corta real

Mientras la distancia más corta real es de $d = \sqrt{L^2 + L^2}$, el resultado de Dijkstra es $2L$ y lo será siempre sin importar que tan subdividido este el g . Esto sucede debido a que el algoritmo de Dijkstra solo avanza por las aristas del grafo, que son las interconexiones conocidas de la malla que representa la superficie discretizada.

3.3.2. La solución Geodésica poliédrica y suave

Como se explica en [36], el resultado obtenido con Dijkstra en la sección anterior no es incorrecto. Resultados como los de Dijkstra y otros algoritmos con bases en computación geométrica son útiles para encontrar la distancia geodésica cuando la geometría de la superficie discreta es exacta, cosa que sucede, por ejemplo, en superficies creadas por software de modelado 3D o CAD.

A este tipo de circunstancias se les clasifica en [36] como el “Problema geodésico poliédrico”, diferenciándolos del “Problema geodésico suave” donde la distancia se mide sobre una geometría discreta que es una aproximación de una geometría real

Para casos como el segundo, como ya se mostró, métodos poliédricos como Dijkstra no se aproximan a la solución de una geodésica real. Esto da pauta a buscar algoritmos que den resultados más exactos al problema de medir distancia geodésica sobre superficies suaves que, como se verá a continuación, son algoritmos relacionados con la simulación computacional y resolución de ecuaciones diferenciales parciales.

3.3.3. Técnicas basadas en resolver ecuaciones diferenciales

La idea general de estos métodos consiste en resolver con simulación computacional un problema de valor inicial o de frontera para generar un superficie de coste sobre la geometría discretizada. Dicho coste estará relacionado con la distancia y con un algoritmo de optimización, como descenso del gradiente, se podrá obtener un camino mínimo o distancia geodésica.

En [36] se hace una clasificación de estos métodos entre “Métodos de Frente de Onda” y “Métodos de difusión”, la diferencia siendo el tipo de ecuación que caracteriza a la distancia.

En los “Métodos de Frente de Onda” el problema se transforma en resolver la propagación de un frente de onda para medir distancias. Una forma popular de simular la propagación de la onda es con la ecuación eikonal cuyo origen y uso se explica en la siguiente sección.

3.4. La ecuación eikonal

La ecuación eikonal es la ecuación fundamental de óptica geométrica que permite a la vez, encontrar el avance de un frente de onda de partículas y de los rayos de luz, que son las curvas ortogonales al frente de onda. Es por ello que se puede derivar la ecuación desde el análisis de la propagación de un rayo de luz o desde la ecuación de una onda electromagnética que se propaga en el espacio (partiendo de las ecuaciones de Maxwell). Para el primer caso se puede obtener utilizando la formulación conocida en mecánica clásica como de Hamilton-Jacobi, como se muestra a continuación:

Partiendo del principio de Fermat, el tiempo en que se propaga la luz por un medio isotrópico desde un punto A a B , también llamado ‘longitud de camino óptico’, se encuentra con la ecuación 6.

$$S = \int_A^B n ds \quad (6)$$

donde n es el índice de refracción del medio y ds la longitud de arco sobre la trayectoria

$$ds = \sqrt{dx^2 + dy^2 + dz^2}$$

La ecuación 6 se parametriza respecto a Z , para obtener:

$$S = \int_A^B n dz \sqrt{\dot{x}^2 + \dot{y}^2 + 1} \quad (7)$$

donde \dot{x} \dot{y} son derivadas respecto a z . De la ec. 7 podemos obtener el langrangiano (ec. 8)

$$L(x, \dot{x}, y, \dot{y}, z) = n(x, y, z) \sqrt{1 + \dot{x}^2 + \dot{y}^2} \quad (8)$$

Si definimos la ec. 8 con los momentos generalizados, donde un momento generalizado p_i es:

$$p_i = \partial L / \partial \dot{x}_i = n \frac{\dot{x}_i}{\sqrt{\dot{x}_1^2 + \dot{x}_2^2 + 1}}$$

se obtiene la ecuación 9

$$(\nabla S)^2 = p_x^2 + p_y^2 + p_z^2 = n^2 \quad (9)$$

que es una representación clásica de la llamada ‘ecuación eikonal escalar’, donde n es un valor real.

El langrangiano se puede obtener de esta función de momentos (ec. 9), de la siguiente manera:

$$S = \int_A^B n \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} ds$$

$$L = n \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}$$

$$L = n \frac{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}{\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}}$$

$$L = \dot{x}p_x + \dot{y}p_y + p_z \quad (10)$$

Para obtener el hamiltoniano se aplica la transformada de Legendre a 10, que consiste en multiplicar la derivada de la función original por la variable de derivación y restar la función original. Esto da como resultado:

$$H = \dot{x}p_x + \dot{y}p_y - L$$

que es igual a :

$$H = -P_z$$

$$H = -\sqrt{n^2 - p_x^2 - p_y^2} \quad (11)$$

Finalmente, de 11 se obtiene la ecuación de Hamilton-Jacobi.

$$\left(\frac{\partial S}{\partial z}\right) - \sqrt{n^2 - \left(\frac{\partial S}{\partial x}\right)^2 - \left(\frac{\partial S}{\partial y}\right)^2} = 0 \quad (12)$$

Al elevar la ec. 12 al cuadrado y reacomodar sus términos se obtiene

$$\left(\frac{\partial S}{\partial z}\right)^2 + \left(\frac{\partial S}{\partial x}\right)^2 + \left(\frac{\partial S}{\partial y}\right)^2 = n^2 \quad (13)$$

que es la forma general de la ecuación eikonal. En óptica geométrica la ecuación eikonal es la EDP de Hamilton Jacobi de la luz que se propaga por un medio.

En [37] se puede encontrar una solución para el caso de la evolución de una onda, utilizando las ecuaciones de Maxwell. Aquí solo se mostrará la relación entre la ecuación eikonal y un frente de onda que avanza.

3.4.1. Aproximación de la eikonal como un frente de onda

Analizamos el problema en 1 dimensión. Consideremos una curva $\Gamma \in \Omega$ que se expande con velocidad $F > 0$ (Fig. 14). Ya que solo la componente normal de la velocidad es la que produce el avance de la curva hacia adelante, ignoramos la componente tangencial, por lo que F es una velocidad normal.

$$\frac{dx}{dt} = F(x, t)n$$

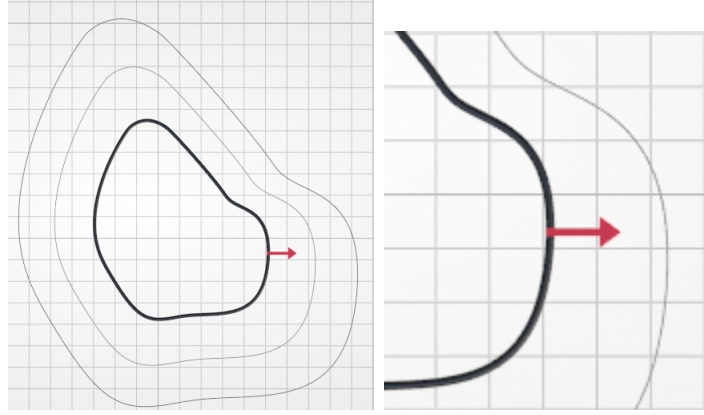


Figura 14: Curva Γ creciendo en dirección de la normal (vector color rojo).

Para caracterizar la posición en el espacio se puede utilizar una función $T(x)$ que nos da el tiempo de llegada t al pasar por la coordenada x . Entonces el tiempo de llegada se puede ver como:

$$T(x(t)) = t$$

derivando respecto a t resulta:

$$\begin{aligned} \frac{dT}{dx} \cdot \frac{dx}{dt} &= 1 \\ \frac{dT}{dx} F n &= 1 \end{aligned}$$

Que en dimensiones mayores es igual a:

$$\nabla T F n = 1 \tag{14}$$

donde n es la dirección hacia donde avanza la curva y es igual a la dirección del gradiente normalizada

$$n = \frac{\nabla T}{|\nabla T|}$$

Sustituyendo la n en la ec. 14 se obtiene:

$$|\nabla T| = \frac{1}{F}$$

donde el tiempo de llegada T es la solución del problema de frontera conocido como la ecuación eikonal.

$$(15) \quad \begin{cases} |\nabla T(x)| F(x) = 1, x \in \Omega \\ T(x) = 0, x \in \Gamma, \end{cases}$$

La función F es dada y en óptica geométrica $\frac{1}{F}$ es el índice de refracción $n(x)$.

3.4.2. Solución viscosa

Las soluciones viscosas son una teoría de soluciones ‘débiles’ para EDP no lineales, como la ecuación eikonal.

En [38] se demostró que la ecuación de Hamilton-Jacobi estática:

$$\begin{cases} H(x, u(x), Du(x)) = 0, & x \in \Omega \\ u(x) = \phi, & x \in \partial\Omega, \end{cases} \quad (16)$$

no tiene una solución completa y solo admite soluciones débiles. Por lo tanto, en [38] se aproxima una solución al agregar un término de viscosidad.

La misma idea se puede aplicar a la ecuación eikonal. Para encontrar su solución se busca una solución viscosa T , que es el tiempo necesario para que la curva alcance el punto (x, y, z) . Lo anterior es lo mismo que interpretar el resultado viscoso como la función de distancia. En [39] se presenta con mayor detalle como las soluciones viscosas de la ecuación eikonal son funciones de distancia y como la distancia euclidiana es solución de la ecuación eikonal homogénea.

3.4.3. Aproximación numérica

Debido a que la ecuación eikonal es una ecuación diferencial parcial hiperbólica y que su modelado asemeja una onda que se propaga hacia el frente, un método numérico de discretización con esquema ‘upwind’ (contraviento) es apropiado para producir una solución viscosa de la ecuación. Este tipo de solución también toma en cuenta que la velocidad de propagación es siempre positiva $F > 0$, por lo que la onda se propaga hacia la frontera.

Consideremos la ecuación 17, que propone [40]:

$$[\max(D_{ijk}^{-x}T, 0)^2 + \min(D_{ijk}^{+x}T, 0)^2 + \max(D_{ijk}^{-y}T, 0)^2 + \min(D_{ijk}^{+y}T, 0)^2 + \max(D_{ijk}^{-z}T, 0)^2 + \min(D_{ijk}^{+z}T, 0)^2]^{1/2} = F_{ij} \quad (17)$$

Donde D es el operador de diferencias hacia atrás y hacia adelante en x , y y z , que aproxima la solución con el método de euler.

$$D_i^{-x} = \frac{\phi_i - \phi_{i-1}}{h} \quad D_i^{+x} = \frac{\phi_{i+1} - \phi_i}{h}$$

Otro esquema tipo upwind ampliamente utilizado es el llamado de Godunov [41]:

$$[\max(D_{ijk}^{-x}T, -D_{ijk}^{+x}T, 0)^2 + \max(D_{ijk}^{-y}T, -D_{ijk}^{+y}T, 0)^2 + \max(D_{ijk}^{-z}T, -D_{ijk}^{+z}T, 0)^2]^{1/2} = F_{ij} \quad (18)$$

En ambos casos las diferencias son tomadas de los vecinos siguiendo una plantilla de un grid cartesiano 3D como se muestra en la Fig. 15.

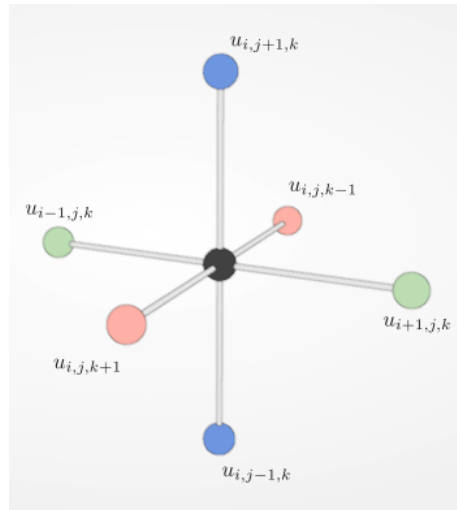


Figura 15: Plantilla de vecinos para actualizar el valor de un nodo

3.5. El Método de Marcha Rápida

El método de de Marcha Rápida [42] o Fast Marching Method (FMM) resuelve la ecuación de propagación de frente de onda (eikonal) sobre un grid ortogonal cuando la velocidad de propagación es positiva, es decir, que avanza hacia adelante. Cuando se considera que la propagación tiene velocidad positiva y negativa, esto se resuelve con los métodos llamados de level set.

Es el método más popular para resolver la ecuación eikonal. Pertenece a los métodos llamados de 'etiquetado' o 'label setting', igual que el algoritmo de caminos mínimos de Dijkstra. Por otro lado, este método si converge a la distancia geodésica real aún manteniendo la complejidad $O(n \log(n))$ en el peor caso.

3.5.1. Algoritmo

Como se mencionó anteriormente, el algoritmo está inspirado en Dijkstra y tiene la misma metodología de *congelar* valores cuando a estos ya se les ha calculado su distancia máxima. Para esto es necesario poder encontrar vecinos, ya sea con la malla de triángulos o con un método libre de mallas.

La idea general del algoritmo es que, al estar sobre un nodo o punto se calcula su distancia utilizando la información de sus vecinos que ya tienen solución. Posteriormente se avanza hacia los vecinos que no han sido visitados por el algoritmo. Durante el procesamiento esto se sigue con el uso de al menos 3 listas o estructuras de datos. La primera almacena a los nodos *aceptados* que son los que ya se utilizaron para calcular el valor de sus vecinos y su solución ya fue encontrada. Los nodos de la *Banda de Prueba* que representa al frente de onda que va creciendo en cada iteración, agregando nodos para obtener su solución, y los nodos *lejanos* que son los que no han sido alcanzados por el algoritmo aún. Este funcionamiento se observa en el diagrama de la Fig. 16

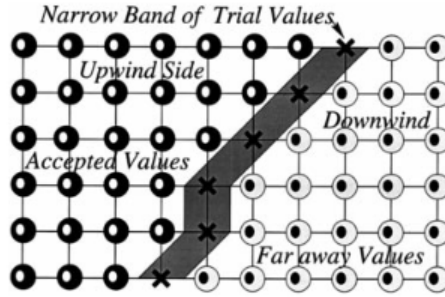


Figura 16: Diagrama del algoritmo de marcha rápida, imagen obtenida de [43]

A continuación se muestran los pasos del algoritmo. Primero con su inicialización y posteriormente con un ciclo iterativo.

Algoritmo 1: Algoritmo FMM

Inicialización:

Inicializar el nodo fuente con valor 0 y el resto como infinito;
 Agregar nodo fuente a la *Banda de Prueba*;

Iteración:

mientras *hay elementos en la Banda de Prueba* **hacer**

- Selecccionar el nodo con valor menor de la *Banda de Prueba*;
- Resolver la ecuación eikonal para los vecinos cardinales al nodo seleccionado y agregarlos a la *Banda de Prueba*;
- El punto seleccionado se *Acepta* y sus vecinos entran a la Banda de Prueba;

fin

Para que este algoritmo sea eficiente es muy común que para la Banda de Prueba se utiliza una estructura Montículo (Heap). El montículo se actualiza cada que un valor del grid es reemplazado.

3.6. Método iterativo Rápido

Un problema con el algoritmo de marcha rápida es que, al igual que Dijkstra, no es paralelizable. La actualización se da celda por celda en un orden estricto, debido a que se toma en cada paso el valor menor actual de distancia.

Se han propuesto diferentes alternativas para lograr la paralelización. Estas buscan eficientar con la elección de otro tipo de estructura de datos o cambiar la metodología con que funciona el método [44]. Entre estos últimos casos esta el llamado Método iterativo Rápido[45] (FIM, por sus siglas en inglés).

Así como FMM es un algoritmo de etiquetado, FIM está basado en algoritmos llamados de 'corrección de etiquetado' o 'label correcting', como el algoritmo de Belmann-Ford que permite encontrar el camino mas corto en una gráfica. Este tipo de algoritmos se basan en iteraciones y en actualizar la distancia mínima encontrada en cada nodo por cada iteración.

Es así que estos métodos abren la posibilidad a la paralelización ya que no hay un orden de actualización y todos los nodos se pueden procesar a la vez. A diferencia de FMM donde la lista se ordena para procesar los valores mas pequeños antes de los mas grandes, FIM desecha la lista ordenada para buscar la paralelización. Por otro lado, esto a su vez causa que la complejidad de operaciones de FIM incremente ($O(kN)$), sin embargo, para nuestro caso, donde se busca distancia

geodésica, la simulación de onda es relativamente sencilla, sin colisiones con otras ondas y con una función de velocidad constante. Con estas condiciones el término k resulta muy pequeño.

Algoritmo:

En este caso el algoritmo sustituye la *Banda de Prueba* por una *Lista Activa* donde se mantienen los nodos hasta que convergen a una solución.

Algoritmo 2: Algoritmo Eikonal - Inicialización y Actualización

Inicialización:

Inicializar condiciones de frontera en el grid;
 Inicializar el nodo fuente a 0 y el resto del grid a infinito;
 Agregar los vecinos cardinales del nodo fuente a la lista activa L ;

Actualización:

```

mientras  $L$  no está vacía hacer
  para cada nodo  $x$  en  $L$  hacer
    Calcular un nuevo  $U_x$  resolviendo la ecuación eikonal;
    Revisar si el punto convergió comparando  $U_x$  actual y anterior;
    si el nodo convergió entonces
      Remover  $x$  de  $L$ ;
      Agregar los vecinos no convergidos de  $x$  a  $L$ ;
    fin
  fin
fin
    
```

En la Fig. 17 se muestra el diagrama del avance del algoritmo.

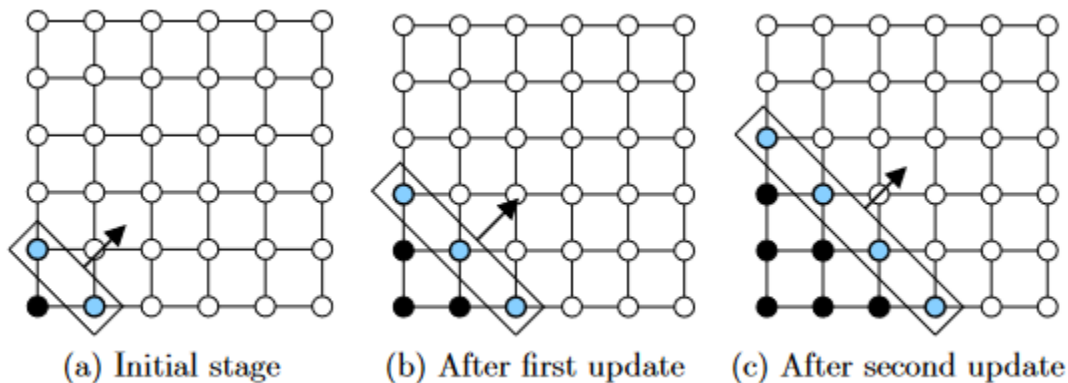


Figura 17: Iteraciones del algoritmo FIM. Imagen obtenida de [46]

3.7. Distancia geodésica sobre nubes de puntos

En [47] se propone un método para medir distancia sobre sub-variedades de R^n y nubes de puntos. El método se basa en aproximar la distancia intrínica calculando distancias euclidianas

extrínsecas sobre una banda que rodea a la variedad. En el caso de nubes de puntos está banda era construida como la unión U de esferas de radio R , donde el centro de cada esfera era un punto submuestreado de la nube de puntos (Fig. 18).

Para resolver la distancia, sobre la banda creada sobre la superficie se resuelve la ecuación eikonal con el método de Marcha Rápida. De esta manera se obtiene un algoritmo para medir distancia geodésica con la misma precisión y complejidad del FMM.

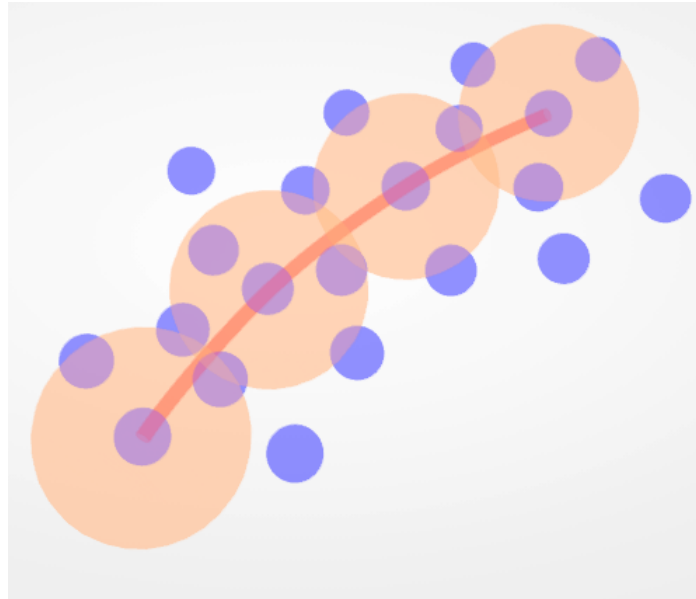


Figura 18: Medición de distancia intrínseca o geodésica utilizando la unión de esferas.

4. Desarrollo

4.1. Selección de puntos

Se presenta el desarrollo de un método de selección de puntos con una técnica *raycast*

4.1.1. Selección con Raycast

Para entender la manera en que funciona el *raycasting* hay que recordar el pipeline de renderizado gráfico para objetos 3D. En general, el proceso consiste en utilizar las matrices Modelo M , Vista V y Proyección P para realizar transformaciones entre los siguientes sistemas de coordenadas:

- Coordenadas del mundo: Son las coordenadas tridimensionales del espacio 3D. Para colocar un objeto en el mundo se multiplican sus vértices por la matriz Modelo.

$$P_{mundo} = M \times P_{obj}$$

- Coordenadas de la cámara: La escena 3D se traslada al espacio de la cámara para su renderizado. Esto se logra multiplicando por la matriz Vista.

$$P_{vista} = V \times P_{mundo}$$

- Coordenadas Proyectadas: Se proyectan los objetos 3D a una imagen 2D. Para ello se multiplica por la matriz de proyección o se dividen las coordenadas x,y entre la z (Coordenadas homogéneas).

$$P_{proj} = P \times P_{vista}$$

- Coordenadas de dispositivo normalizadas (NDC por sus siglas en inglés): Se obtienen coordenadas homogéneas que van de -1 a 1.
- Coordenadas de pantalla: Las coordenadas normalizadas se pasan a la ventana de renderizado. Las coordenadas de pantalla van en x de 0 al ancho de la pantalla y en y , de 0 al alto de la pantalla.

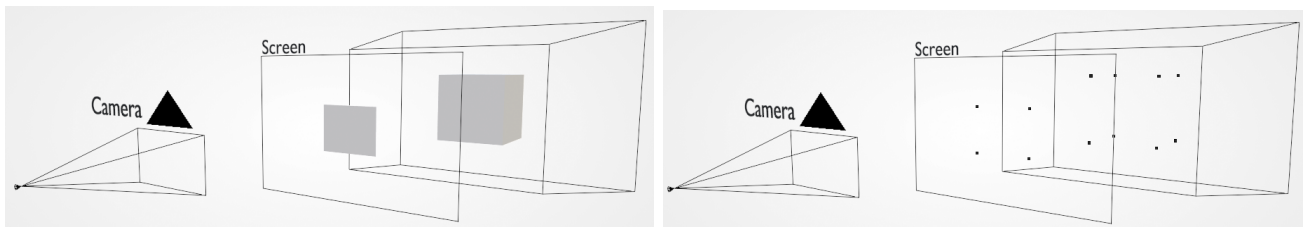


Figura 19: Proyección a NDC de un cubo representado como malla de triángulos (izquierda) y nube de puntos (derecha).

En la Fig. 19 se muestra una representación de un objeto en una escena 3D proyectado a la pantalla.

Ahora tomemos en cuenta que al dar clic sobre la ventana lo que se obtiene son las coordenadas de pantalla que representa a ese píxel. Para recuperar las coordenadas del mundo de la geometría sobre la que se hizo clic se lanza un rayo desde el centro de la cámara con dirección hacia la posición en coordenadas del mundo del punto seleccionado (Fig. 20).

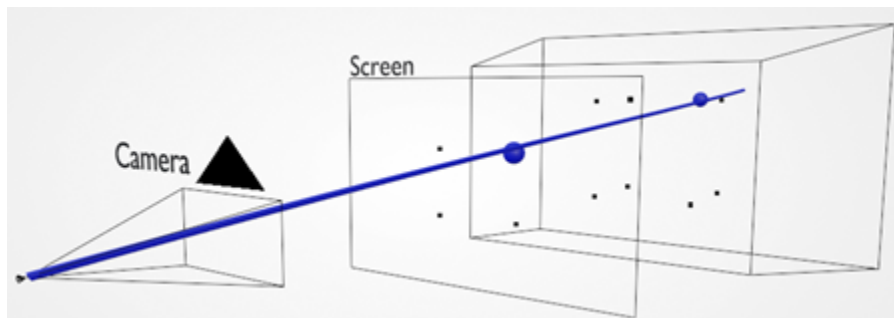


Figura 20: Rayo lanzado desde la cámara en el proceso de *raypicking*

De esta manera, tenemos un rayo:

$$r = O + Dx \tag{19}$$

con origen O y dirección D .

Una vez se tiene el origen y dirección del rayo o recta el siguiente paso es encontrar el punto o puntos que intersecta o que son más cercanos.

Existen numerosas opciones de algoritmos para encontrar la intersección de un rayo para diferentes tipos de geometría. Una opción adecuada para el caso actual sería utilizar la división espacial del octree, que es realizada como un cubo para cada nodo. Al utilizar algoritmos para encontrar intersecciones con una caja cualquiera [48] o caja delimitadora orientada en ejes (AABB por sus siglas en inglés) con métodos como el ‘método slab’ [49] es posible encontrar los nodos del nivel más bajo de profundidad que intersectan al rayo. El siguiente paso y último paso sería encontrar la distancia más cercana de las partículas dentro de los nodos intersectados.

Por otro lado, otra opción viable debido a la paralelización, es obtener la distancia de cada partícula a la recta, de forma paralela. Este caso es paralelo para cada partícula, mientras que el anterior es paralelo para cada nodo y después de encontrar todas las intersecciones de nodos con el rayo, hay que buscar la partícula más cercana dentro de los nodos intersectados.

Para aprovechar mejor la paralelización se opta por la segunda opción, donde la distancia más cercana se calcula utilizando la proyección del punto sobre la recta, como se muestra en la figura 21.

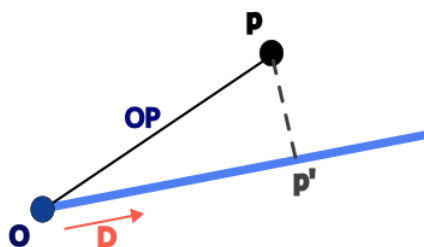


Figura 21: Medición de distancia mínima de un punto a una recta

Como se sabe, la distancia mas corta entre el punto y una recta es la distancia perpendicular del punto hacia la recta. Para esto se encuentra la proyección de una partícula con posición P sobre una recta con origen O y dirección D. Esto se obtiene con el producto punto entre los vectores D y OP.

$$t = D \cdot OP$$

Para encontrar el punto P' :

$$P' = O + t \cdot D$$

Finalmente la distancia del punto a la recta es:

$$dist = \|P' - P\|$$

Una vez se obtienen todas las distancias mas cercanas entre cada punto y la recta, el último paso es seleccionar entre todos los puntos el mas cercano a la recta. Este punto se tomará como el seleccionado.

En la siguiente sección se explica la implementación de este método con CUDA.

4.1.2. Implementación con CUDA

Al detectar el click del ratón se obtienen las coordenadas de pantalla respecto a la posición actual del cursor $cursor_x, cursor_y$. Posteriormente se obtiene el origen y la dirección del rayo con las matriz de proyección P y vista V , siguiendo un proceso de transformación de coordenadas inverso al descrito en la sección anterior. Primero se obtienen las coordenadas NDC.

$$NDC_x = 2 \times \frac{cursor_x}{width} - 1$$

$$NDC_y = 1 - 2 \times \frac{cursor_y}{height}$$

$$\vec{NDC} = \begin{bmatrix} NDC_x \\ NDC_y \\ -1 \\ 0 \end{bmatrix} \quad (20)$$

En la ec. 20 se establece Z como -1. Con esto se puede encontrar dirección en coordenadas del mundo

Para la dirección del rayo, se obtiene con un vector director que parte del origen con dirección a las coordenadas 3D del punto donde se hizo clic.

$$\vec{D} = \frac{V^{-1}P^{-1}\vec{NDC}}{\|V^{-1}P^{-1}\vec{NDC}\|} \quad (21)$$

En cuanto al origen, se puede almacenar la posición actual de la cámara en coordenadas del mundo o es posible obtener la posición con la inversa de la matriz V .

$$\vec{O} = V^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (22)$$

Todo este procesamiento se realiza en ejecución serial dentro del CPU.

Posteriormente se copia el punto de origen y vector dirección a la memoria de la GPU y se lanza el kernel *distanceLine2Point* donde cada hilo de ejecución encuentra la distancia mínima desde cada partícula hasta la recta. La ejecución se da en una cantidad de hilos cercana a la cantidad de partículas.

Después se utiliza la biblioteca de CUDA *thrust* como una medida rápida de ejecutar en dispositivo un ordenamiento de los valores de distancia de menor a mayor. Con esto se logra encontrar el mínimo de las distancias encontradas y lo que será la MK y posición del punto seleccionado.

Para identificar la partícula seleccionada, el atributo *selected* de la estructura *Partícula* de la partícula seleccionada cambia su valor a 1. Finalmente, con el kernel *colorPoint* se cambia el color de la partícula y se identifica como la elegida.

En la Fig. 22 se observa el diagrama de procesos del método descrito. La ejecución considera tanto código en “Host” (CPU) y “Device” (GPU).

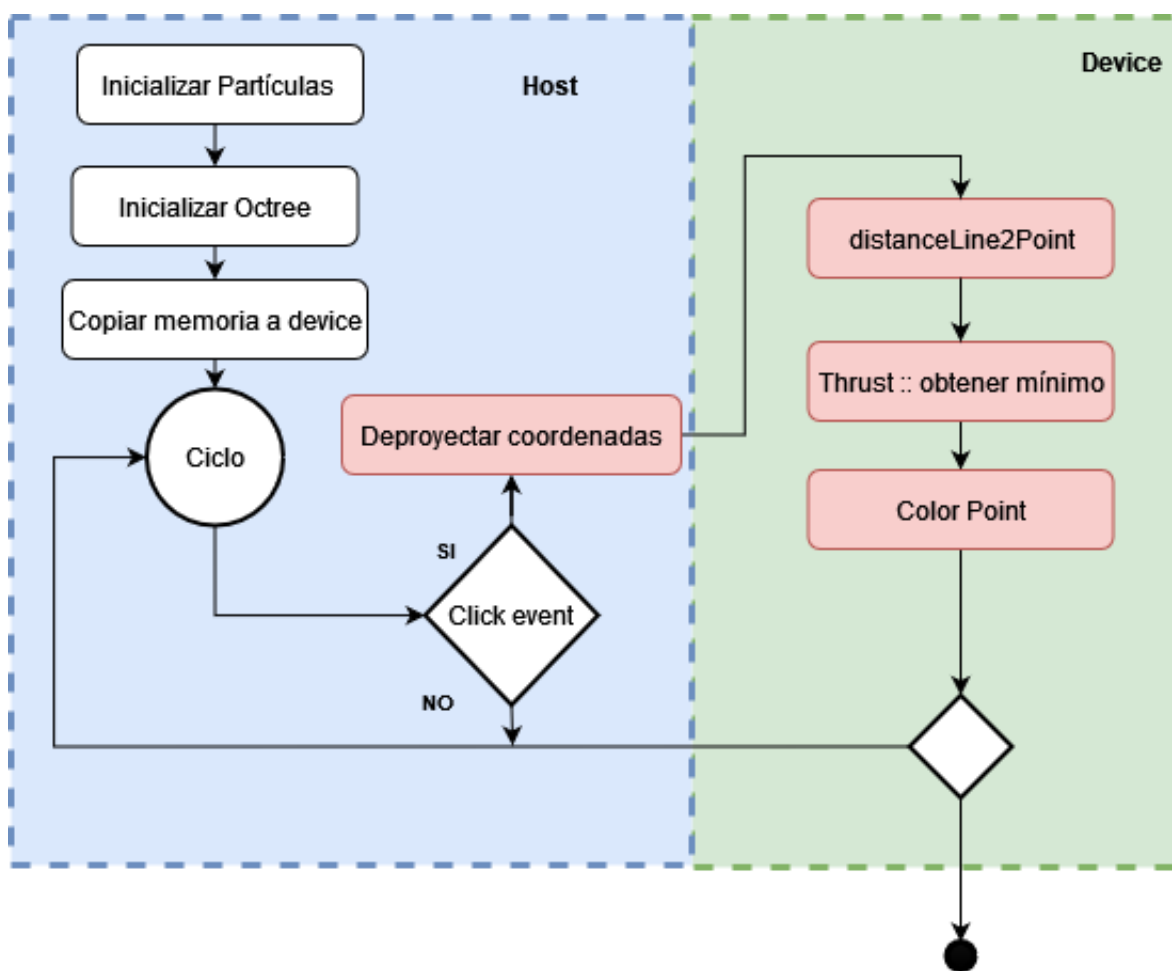


Figura 22: Diagrama de procesos del método de selección implementado

4.2. Medición de distancia geodésica

A continuación se presenta el algoritmo propuesto para resolver el problema de distancia y camino geodésico.

4.2.1. Método

Inspirado en [47] se plantea aproximar la distancia geodésica por medio de distancias euclidianas extrínsecas. Para ello se aprovecha la división espacial del octree implementado en [29]. Se busca que el octree funcione como una cuadrícula tridimensional (Fig. 23) sobre el que se resuelve la ecuación eikonal (ec. 3.4.1).

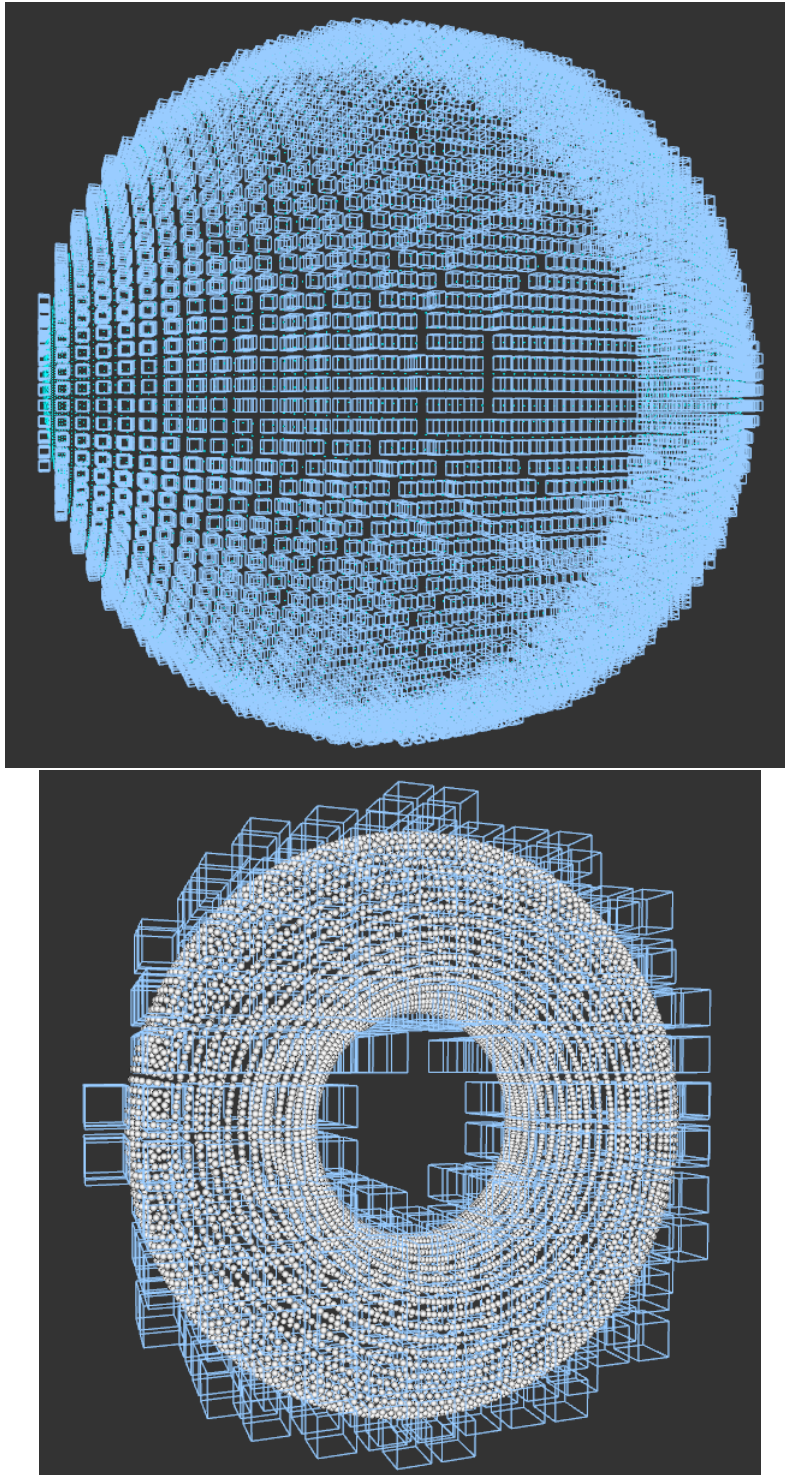


Figura 23: Visualización del octree como Cuadrícula tridimensional. Arriba: nube de puntos de media esfera. Abajo: Nube de puntos de toroide.

Para aprovechar la paralelización con CUDA, en lugar de utilizar el FMM la resolución de la ec. 3.4.1 sobre la nube se basa en el algoritmo de FIM [45].

En un principio se utilizan los 6 vecinos cardinales que considera el algoritmo de FIM para avanzar el frente de onda. Para esto se utiliza el mismo método de cálculo de vecinos para los nodos del octree, implementado en [29].

El esquema de discretización utilizado para resolver la ecuación es el de Godunov hacia adelante. Para obtener la solución se resuelve la ecuación cuadrática (ec. 18) de acuerdo al algoritmo siguiente, presentado en [50]:

Algoritmo 3: Pseudocódigo discretización [50]

```

 $X_{min} = \min(\phi^{-x}, \phi^{+x});$ 
 $Y_{min} = \min(\phi^{-y}, \phi^{+y});$ 
 $Z_{min} = \min(\phi^{-z}, \phi^{+z});$ 

ordenar( $X_{min}, Y_{min}, Z_{min}$ ) como  $a \leq b \leq c$ ;

si  $|a - c| < \delta$  entonces
|  $u = (a + b + c + \sqrt{(a + b + c)^2 - 3(a^2 + b^2 + c^2 - cost)}) * (1/3);$ 
en otro caso
| si  $|a - b| < \delta$  entonces
| |  $u = (a + b + \sqrt{2 * cost^2 + (a - b)^2}) * (1/2);$ 
| en otro caso
| |  $u = a + cost;$ 
| fin
fin

```

La ecuación cuadrática del Algoritmo 3 se obtiene de la ecuación 23.

$$\left[\frac{U(x) - U(x)^{x_{min}}}{\Delta X} \right] + \left[\frac{U(x) - U(x)^{y_{min}}}{\Delta Y} \right] + \left[\frac{U(x) - U(x)^{z_{min}}}{\Delta Z} \right] = \frac{1}{f(x)^2} \quad (23)$$

donde Δ es el espaciado en la cuadrícula ($\Delta X = \Delta Y = \Delta Z = 1$), $f(x) = 1$ es la velocidad de propagación, U es la aproximación ϕ a un nodo $x = (i, j, k)$ y $U(x)^{min}$ es el mínimo sobre los vecinos adyacentes en el eje x,y,z.

Sin embargo, al probar este algoritmo en diferentes geometrías 3D se obtienen resultados inconsistentes debido a que en ocasiones no se logra encontrar el vecino adecuado para continuar el avance del frente de onda. Una opción para solucionar este problema es disminuir el nivel de profundidad del árbol sobre los nodos que se ejecuta el algoritmo. Sin embargo, esta opción creará menor precisión en el método. Otra opción es aumentar el número de vecinos que se buscan, lo cual es fácil de realizar ya que el método de búsqueda de vecinos considera encontrar hasta 26 vecinos del nodo.

Por lo tanto, se opta por buscar 26 vecinos en lugar de 6. En este caso, el cálculo de distancia se realiza sumando la distancia a la que convergió el nodo actual mas la distancia euclidiana entre el nodo actual y el nodo vecino. En la siguiente sección se especifica mejor el funcionamiento del método.

Es de esta manera que se resuelve la ecuación eikonal con función de velocidad constante 1 y comenzando desde 1 punto hacia todos los demás puntos, es decir, se obtiene la distancia intrínseca de un punto a todos.

4.2.2. Algoritmo

Para la implementación del algoritmo se define la estructura *OctreeNode*, con los siguientes atributos:

- **morton_key:** Clave Morton del nodo.

- **particleIndex:** Índice (de un arreglo que contiene ordenados a todos los puntos de la nube [29]) de la primer partícula del nodo.
- **p_count:** Cantidad de puntos en el nodo.
- **statusVis:** Variable para llevar registro de la lista activa. Los valores posibles son: 1 para los nodos *lejanos*, 0 para los que están en la *lista activa*, -1 para los nodos que convergen y -3 para el nodo inicial (source).
- **fimSolution:** Resultado de la iteración actual del algoritmo.
- **fimSolution_last:** Resultado de la iteración anterior del algoritmo. Se guarda el resultado para comparar y saber si el nodo convergió.
- **cost:** Costo relacionado al nodo.

En la Fig. 24 se muestra el diagrama de procesos del algoritmo tanto en *Host* como *Device*.

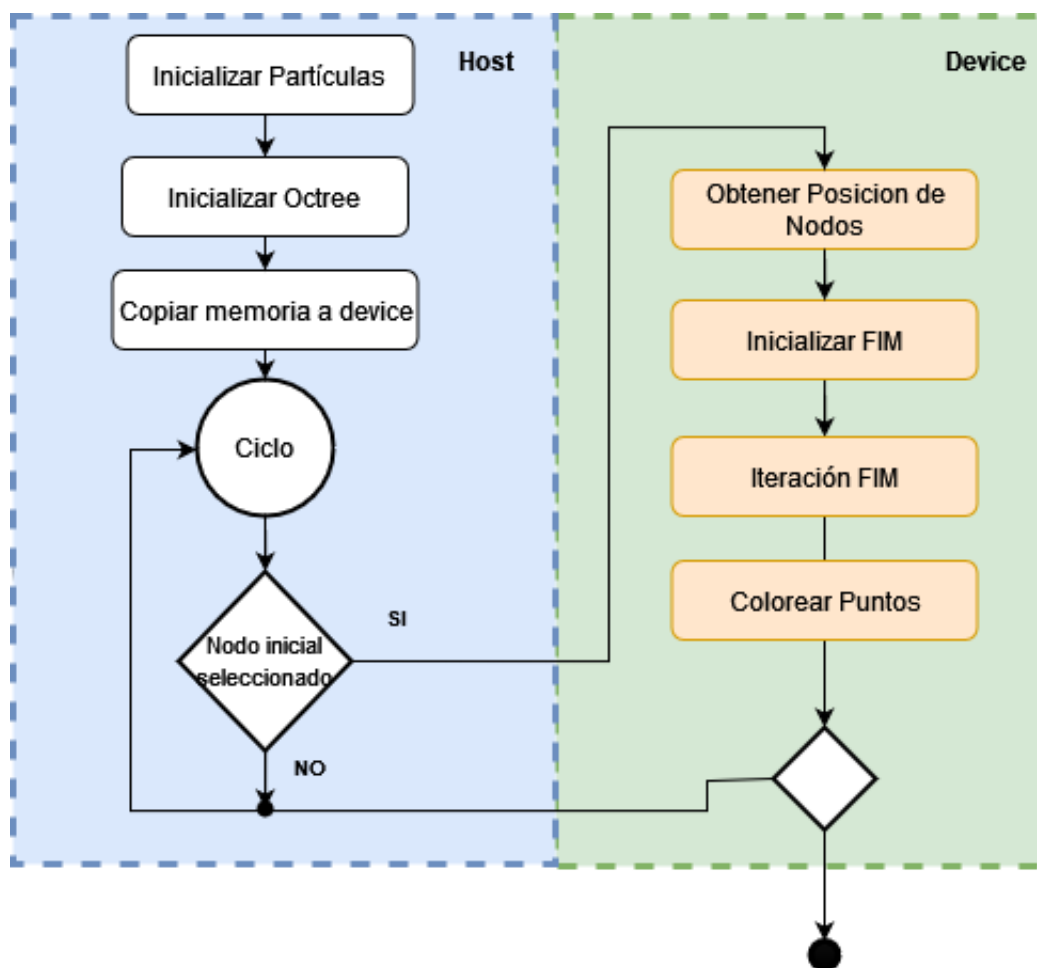


Figura 24: Diagrama de procesos del método de medición de distancia implementado.

Los procesos son los siguientes:

- **Obtener posición de nodos:** Para ello se decodifica la clave Morton:

- Se forman las posiciones normalizadas de 10 bits al ir tomando cada 3 dígitos de la clave Morton, dependiendo de si se busca la pos x,y o z.
- Se obtiene la posición normalizada dividiendo entre 400_{16}

$$\bar{n}_x = \frac{\bar{n}_{xi}}{400_{16}} \quad \bar{n}_y = \frac{\bar{n}_{yi}}{400_{16}} \quad \bar{n}_z = \frac{\bar{n}_{zi}}{400_{16}}$$

- Se obtiene la posición en el espacio multiplicando la posición normalizada por las dimensiones del octree.

$$n_x = \bar{n}_x \cdot oct_{width} \quad n_y = \bar{n}_y \cdot oct_{height} \quad n_z = \bar{n}_z \cdot oct_{depth}$$

- **Inicializar FIM:** Primera iteración del algoritmo. Converge el nodo inicial y sus vecinos se agregan a la lista.
- **Iteracion FIM:** Se calcula *fimSolution* para todos los nodos de la lista activa con la ecuacion 24.

$$fimSolution = \phi + costo \tag{24}$$

donde el costo es la distancia euclidiana entre el nodo actual y su vecino con el valor menor de distancia ϕ .

$$costo = \sqrt{(U_{min_x} - U_x)^2 + (U_{min_y} - U_y)^2 + (U_{min_z} - U_z)^2}$$

- **Colorear puntos:** Se colorea el resultado *fimSolution* de forma interpolada con el kernel *paintDistance*. El valor del gradiente va de 0 a al valor de distancia máxima encontrado. Los colores siguen el siguiente orden (Fig. 25): amarillo \rightarrow naranja \rightarrow rojo \rightarrow morado \rightarrow azul.



Figura 25: Gradiente de color que ejemplifica los colores con los que se visualiza el resultado.

El Algoritmo se ejecuta como se muestra en el Alg. 4.

Algoritmo 4: Algoritmo para método propuesto

Inicialización

Inicializar $statusVis$ en el nodo fuente como -3 , y en todos los demás como 1 ;
Inicializar $fimSolution$ en el nodo fuente como 0 , y en los demás como ∞ ;
Inicializar la posición espacial de cada nodo. Lanzar el kernel $iInitFastIterative$ para la primera iteración ;

Actualización

```
mientras Lista activa no está vacía hacer
  para cada Nodo X en Lista activa hacer
    Calcular  $fimSolution_X$  con ec. 24.
    si  $fimSolution_X \leq fimSolution_{last_X}$  entonces
      Se buscan los 26 nodos vecinos;
      Para los nodos vecinos N no vacíos se hace una primera iteración del algoritmo
      para calcular  $fimSolution$  con la ec 24;
      si  $fimSolution_N \leq fimSolution_{last_N}$  entonces
        | N se agrega a la lista activa ( $statusVis = 0$ );
      fin
      Remover X de la Lista activa ( $statusVis = -1$ );
    fin
  fin
fin
Lanzar el kernel paintDistance.
```

4.2.3. Camino geodésico

Se ha desarrollado un algoritmo para dibujar el camino geodésico entre dos puntos especificados. Para ello se selecciona un segundo punto después del nodo inicial (con un clic derecho) y se avanza desde este buscando en los nodos vecinos el valor de distancia menor. Se itera esta operación hasta que se encuentra el nodo inicial ($FIMSolution = 0$).

El algoritmo descrito da como resultado un camino formado por una lista de nodos. Posteriormente, con la posición de dichos nodos se forman múltiples curvas de Bézier de 3er orden, interconectadas, donde los puntos de control son dichos nodos. El camino obtenido se muestra con color verde en la Fig. 26. El ejemplo es de un plano con una partícula por nodo del octree. En verde se colorean las partículas dentro de los nodos seleccionados como parte del camino geodésico y con la línea punteada se dibujan las curvas de Bézier.

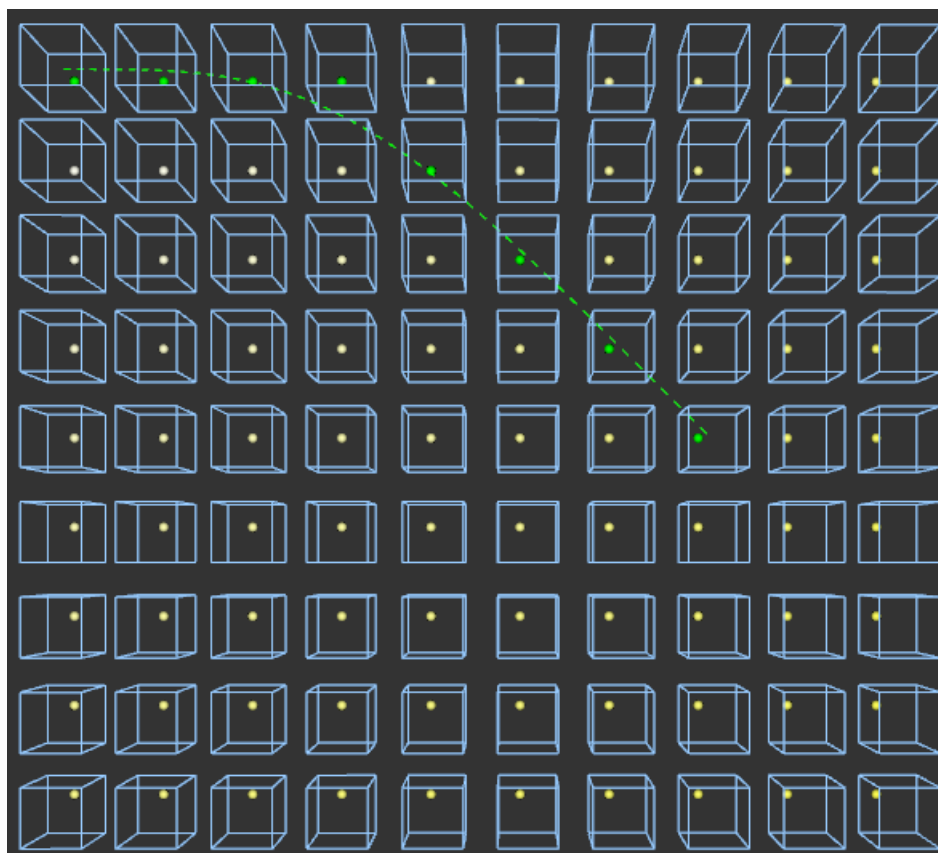


Figura 26: Camino geodésico sobre plano

Además de esto, para facilitar el uso del algoritmo al medir superficialmente zonas de interés como marcas o agujeros sobre la superficie, se implementó la posibilidad de hacer múltiples selecciones con el ratón que formen una sola curva continua. Cada que se hace un nuevo clic el algoritmo utiliza el último punto tomado como punto final como el nuevo punto inicial y encuentra una nueva geodésica. La curva resultante será una suma de la geodésica calculada en cada clic y la distancia total la suma de las distancias encontradas.

5. Pruebas y resultados

Las pruebas para los resultados mostrados se realizan con un equipo con las siguientes especificaciones:

- CPU: AMD FX8370
- Memoria: 12Gb RAM
- GPU: Geforce GTX 970

La tarjeta gráfica cuenta con las siguientes características:

- Procesador:
 - 1664 CUDA Cores
 - Frecuencia base de 1050 Mhz
 - Frecuencia boost de 1178 Mhz
- Memoria:
 - 4Gb de memoria GDDR5
 - 256 bit de bus de memoria
 - Ancho de banda de memoria de 225 GB/s

5.1. Resultados algoritmo de selección

En la tabla 1 se muestran los tiempos de procesamiento del algoritmo de selección desglosado por sus etapas:

- Deployectar: Deployectar coordenadas de pantalla.
- distance2Line: Medir distancia de cada punto al rayo.
- Thrust min: Encontrar distancia mínima con Thrust.
- colorPoint: Cambiar identificación y color del punto seleccionado.

El procesamiento se compara entre nubes con diferentes densidades de puntos. En este caso la densidad es la cardinalidad del conjunto de puntos que forman la nube.

Tabla 1: Tiempos de procesamiento de algoritmo de selección con nubes de puntos de diferentes densidades

Densidad	108,875	254,012	495,612	1,968,409	7,868,025
Procesos	Tiempo (ms)				
Deployectar	5	6	5	4	270
distance2Line	0.073	0.15	2.47	12.947	137.68
Thrust min	0.392	0.375	0.306	0.38	2.64
colorPoint	0.004	0.015	0.007	0.019	0.067
Total	5.469	6.54	7.783	17.346	410.387

En la Fig. 27 se muestra la ventana del programa renderizando una nube de puntos (puntos de color gris) con 2 puntos seleccionados. El punto amarillo es el punto de inicio del algoritmo y se selecciona con clic izquierdo. El segundo punto, en azul, sería el final del camino geodésico y se selecciona con clic derecho.

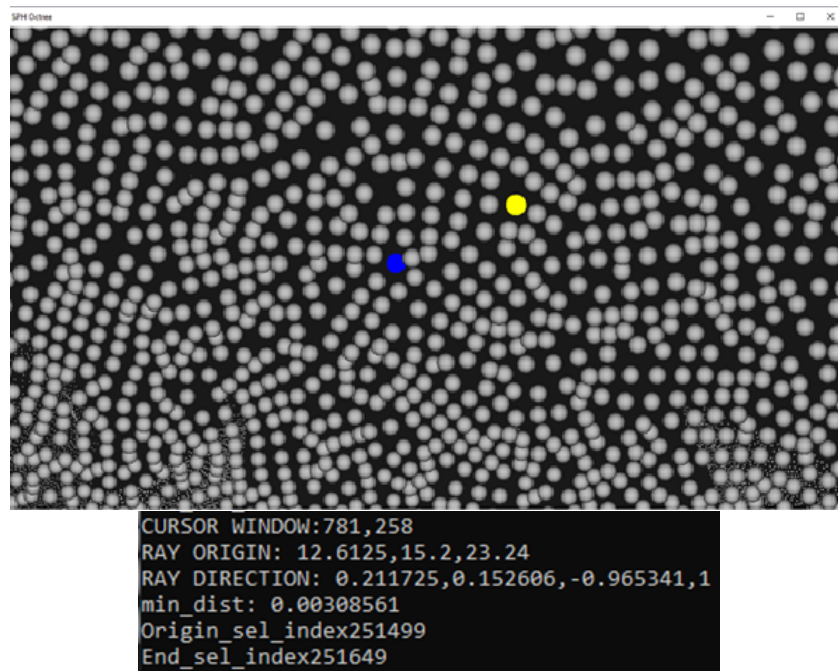


Figura 27: Selección de puntos visualizada por el software. Punto amarillo es el seleccionado con clic izquierdo y el punto azul es seleccionado con clic derecho.

5.2. Pruebas del algoritmo de distancia geodésica con geometrías conocidas

Se pone a prueba el funcionamiento del algoritmo con planos cuadrados y medias esferas, de diferentes dimensiones y densidades de puntos, creadas con el software de modelado 3D Blender [51]. Esto tiene el objetivo de utilizar objetos 3D con geometría sencilla y distancias simples de calcular y comparar. El software Blender crea estos objetos con un espaciado uniforme y unidades en *Blender Units*. Esta es una unidad interna del programa que se puede referenciar a una unidad real. Para el caso de los resultados mostrados, la información se deja como una unidad adimensional.

Los modelos de plano y esfera se muestran en la Fig. 28.

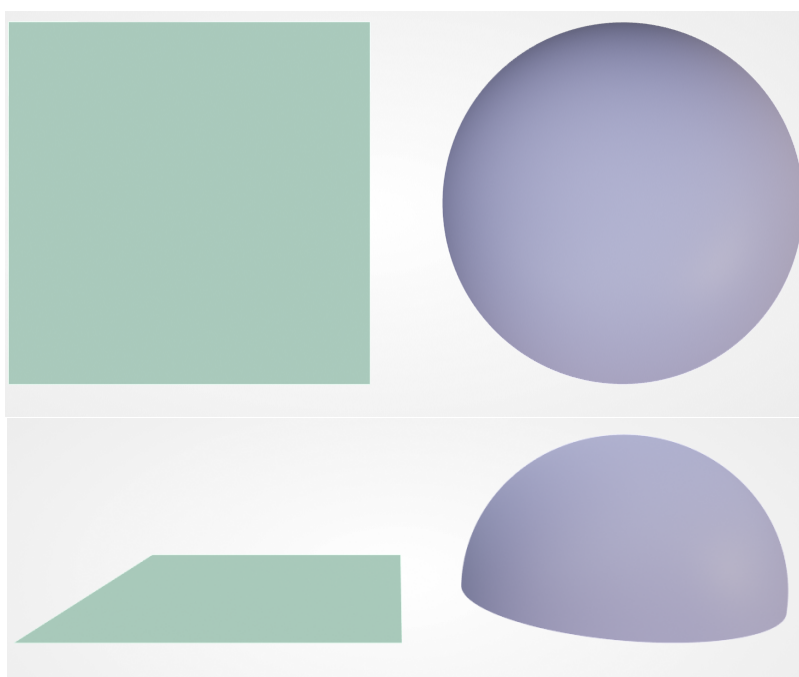


Figura 28: Objetos creados con software de modelado para pruebas. Arriba: vista ortográfica sobre Z. Abajo: vista frontal en perspectiva.

Un punto importante para el uso del programa es la profundidad máxima del octree a utilizar, ya que de esto dependerá lo refinado de la cuadrícula tridimensional sobre la que avanzará el algoritmo. Entre mas refinada sea la cuadrícula mas preciso será el resultado y mas tardado el procesamiento. Al interpretar resultados obtenidos en [29], se decide para las pruebas un nivel de profundidad máximo de 8, pensando en un buen punto medio entre velocidad de procesamiento y precisión.

5.2.1. Resultados con plano

En la figura 29 se presenta el resultado visual del algoritmo al correrlo sobre una nube de puntos que representa un plano de dimensiones 10x10. El punto inicial en este resultado es la esquina superior izquierdo del plano. Cada punto de la nube se renderiza de acuerdo a un pseudocolor que representa la distancia del punto actual al punto inicial o fuente. Los colores inician en blanco para los valores más cercanos y terminan en azul para los mas lejanos.

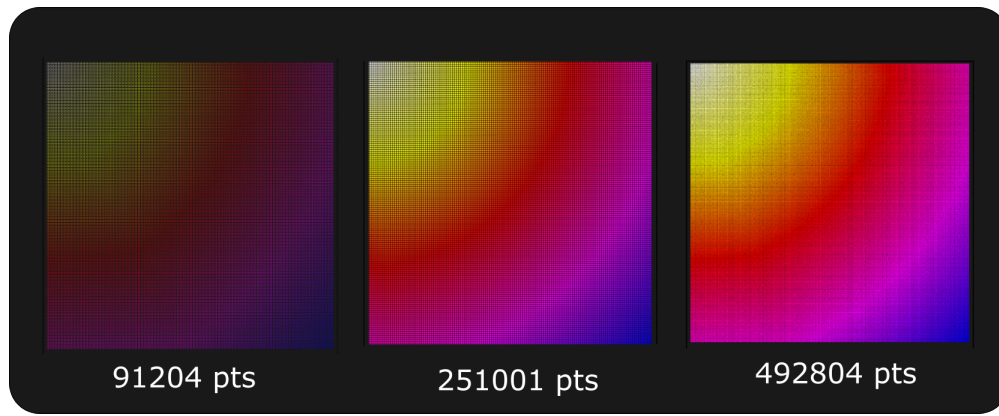


Figura 29: Resultado visual del algoritmo de medición de distancia geodésica en plano de 10x10 con diferentes densidades de puntos.

En la imagen 30 se muestra con valores numéricos como cambia el color respecto al valor de distancia calculado.

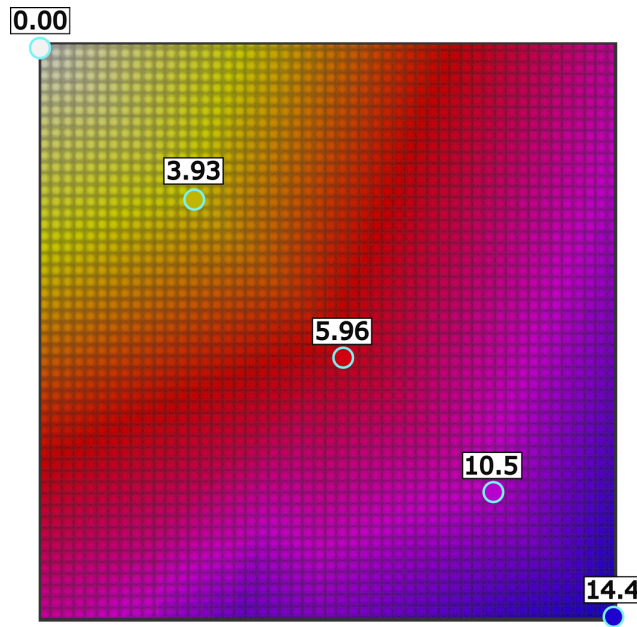


Figura 30: Resultado del algoritmo de medición con marcas de distancia

En las tablas 2 y 3 se observan resultados numéricos de distancia para planos de diversas dimensiones, junto con el cálculo de error porcentual y total. Las mediciones consideradas son ‘Lado a Lado’, es decir un lado del cuadrado y ‘Diagonal’, que es la diagonal del plano.

Tabla 2: Resultados de medir distancia Lado a Lado en un plano cuadrangular de diferentes dimensiones

		Medición 'Lado a Lado'			
Dimensiones	# de puntos	Dist. Real	Dist. Algo.	Error Rel. %	Error total
1x1	108,875	1	1.13	13	0.13
10x10	254,012	10	10.04	0.4	0.04
50x50	495,612	50	49.87	0.26	0.13
100x100	1,968,409	100	100.078	0.078	0.078
200x200	7,868,025	200	199.687	0.156	0.313

Tabla 3: Resultados de medir distancia Diagonal en un plano cuadrangular de diferentes dimensiones

		Medición 'Diagonal'			
Dimensiones	# de puntos	Dist. Real	Dist. Algo.	Error Rel. %	Error total
1x1	108,875	1.414	1.599	13.083	0.185
10x10	254,012	14.14	14.47	2.333	0.33
50x50	495,612	70.71	70.532	0.251	0.178
100x100	1,968,409	141.421	141.531	0.077	0.11
200x200	7,868,025	282.842	282.4	0.156	0.442

En la tabla 4 se observan los tiempos de ejecución para el plano de 10x10 con diferentes densidades de puntos y en 5 los resultados de tiempos para planos de diferentes dimensiones. Los tiempos están desglosados en los 4 procesos que componen el procesamiento del algoritmo. Los tiempos seleccionados son los mas altos encontrados después de probar el algoritmo 10 veces.

Tabla 4: Tiempos por proceso de un plano de 10x10 con diferentes densidades

Densidad de puntos	2,704	92,412	254,012	497,008
Procesos	Tiempo (ms)			
nodePos	9.776	11.57	11.58	3.434
initFIM	3.406	3.45	3.411	3.434
FIM	191.17	195.35	195.87	189.72
paintDistance	3.408	3.4517	3.6	3.783
Total	207.76	213.8217	214.461	200.371

Tabla 5: Tiempos por proceso de planos con diferentes dimensiones

Dimensión	1x1	10x10	50x50	100x100	200x200
Procesos	Tiempo (ms)				
nodePos	9.7809	9.75	9.863	9.818	9.819
initFIM	3.406	3.405	3.393	3.424	3.4155
FIM	102.01	195.87	414.5	659.58	913.75
paintDistance	3.16	3.27	3.425	3.981	3.24
Total	118.3569	212.295	431.181	676.803	930.2245

5.2.2. Resultados Media Esfera

Para la media esfera se considera la distancia mas larga desde un extremo a otro, es decir, la mitad de la circunferencia de la esfera. Esta distancia se calcula como $d = \pi * r$, donde r es el radio de la esfera. El resultado visual se puede observar en la Fig. 31

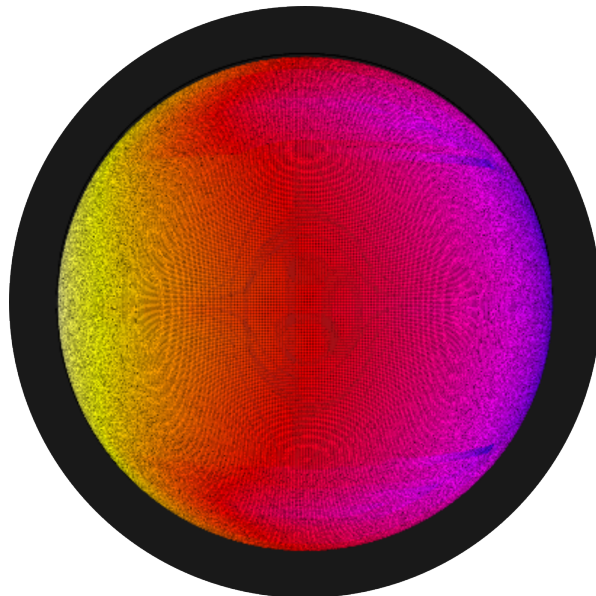


Figura 31: Resultado visual algoritmo de medición de distancia geodésica en media esfera de radio 10.

En la tabla 6 se muestran los resultados con esferas con diferentes medidas de radio y en la tabla 7 los tiempos al procesar la media esfera.

Tabla 6: Resultados de distancia en esferas de diferentes radios
Medición "Perímetro"

Radio	# de puntos	Dist. Real	Dist. Algo.	Error %	Error total
1	3,977	3.1415	2.928	6.7961165	0.2135
5	250,209	15.7	16.7	6.36942675	1
10	999,617	31.415	32.9263	4.81075919	1.5113
50	3,996,033	157	164.8	4.96815287	7.8
100	3,996,033	314.15	331.3	5.45917555	17.15

Tabla 7: Tiempo por procesos con media esfera de diferentes radios

Radio	1	5	10	50	100
Procesos	Tiempo (ms)				
nodePos	9.771	9.78	9.767	9.905	9.89
initFIM	3.39	3.39	3.402	3.404	3.99
FIM	43.179	151.39	261.569	534.0596	912.899
paintDistance	3.414	3.558	4.218	6.5	6.44
Total	59.754	168.118	278.956	553.8686	933.219

5.2.3. Pruebas con otros objetos 3D

En la Figura 32 se muestra el resultado visual con una espiral creada con software de modelado 3D. La forma de la espiral permite ver como se desarrolla la distancia geodésica desde un extremo al otro del objeto (desde color blanco hasta azul).

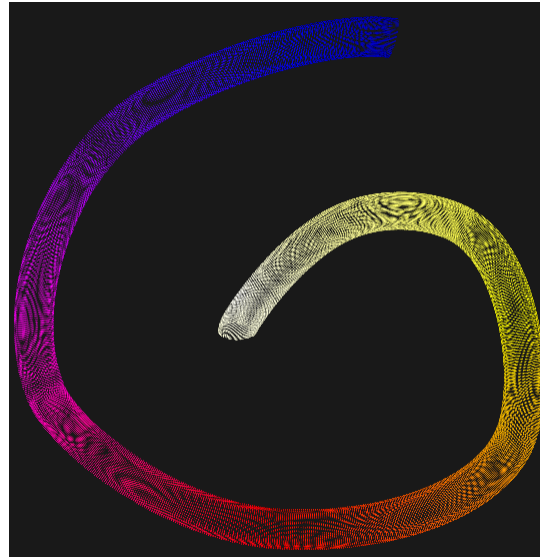


Figura 32: Espiral creada por software de modelado

En la fig. 33 se muestra una prueba con un toroide. Esto permite probar el algoritmo sobre una superficie con un agujero (genus=1).

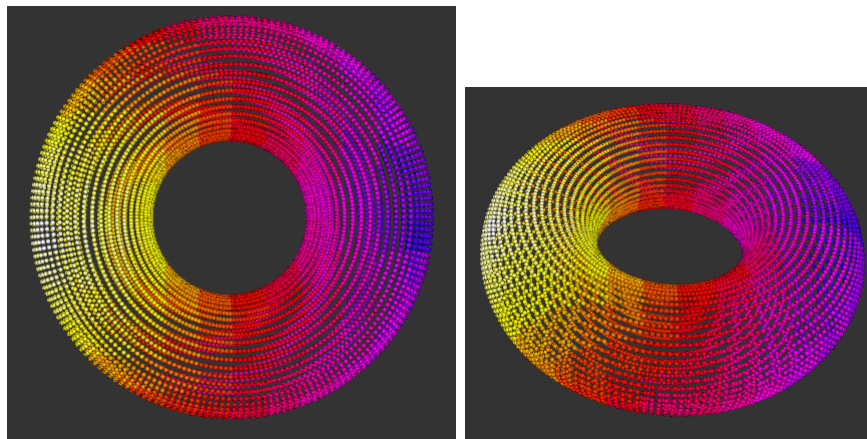


Figura 33: Prueba en toroide

Finalmente en la fig. 34 se muestra el resultado con un paraboloide hiperbólico.

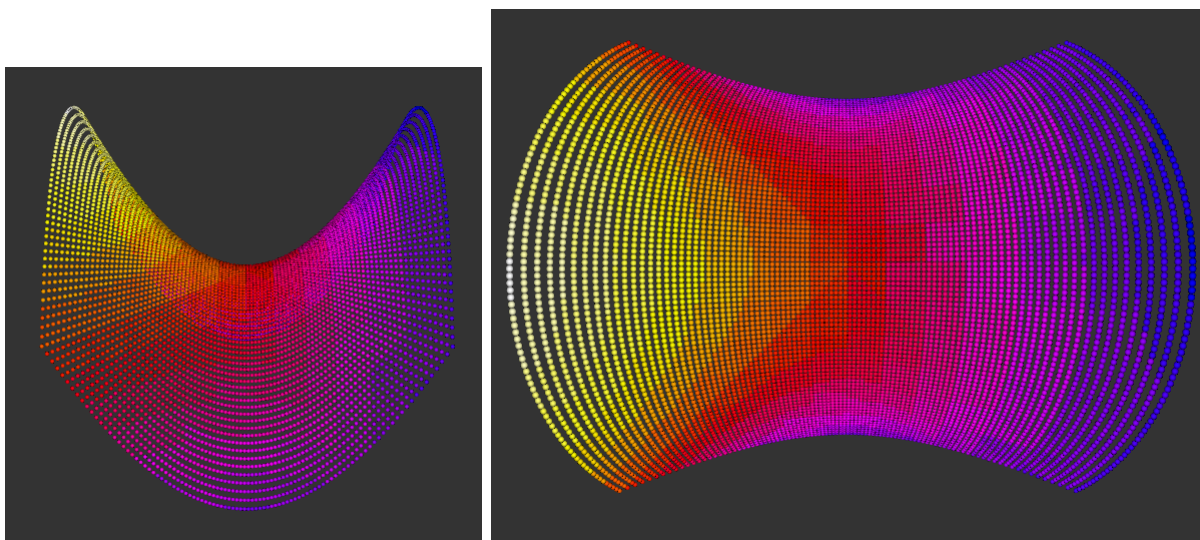


Figura 34: Prueba en paraboloides hiperbólico

En la tabla 8 se recompilan los resultados de tiempo para las pruebas con estos objetos.

Tabla 8: Resultados de tiempo de algoritmo de distancia para espiral, toroide y parab. hip.

Objeto	Densidad	Dist. Máx medida	Tiempo[ms]
Espiral	107,624	39.73	305.4
Toroide	9,216	2.6	46.29
Parab. Hip.	145,161	12.62	125.77

5.3. Resultados del camino geodésico

A continuación se muestran resultados del algoritmo de camino geodésico. En la Fig. 35 se muestra la traza sobre un plano de 10x10 y en Fig. 36 el camino geodésico sobre la espiral. El punto de inicio se marca con 1 y el final con 2.

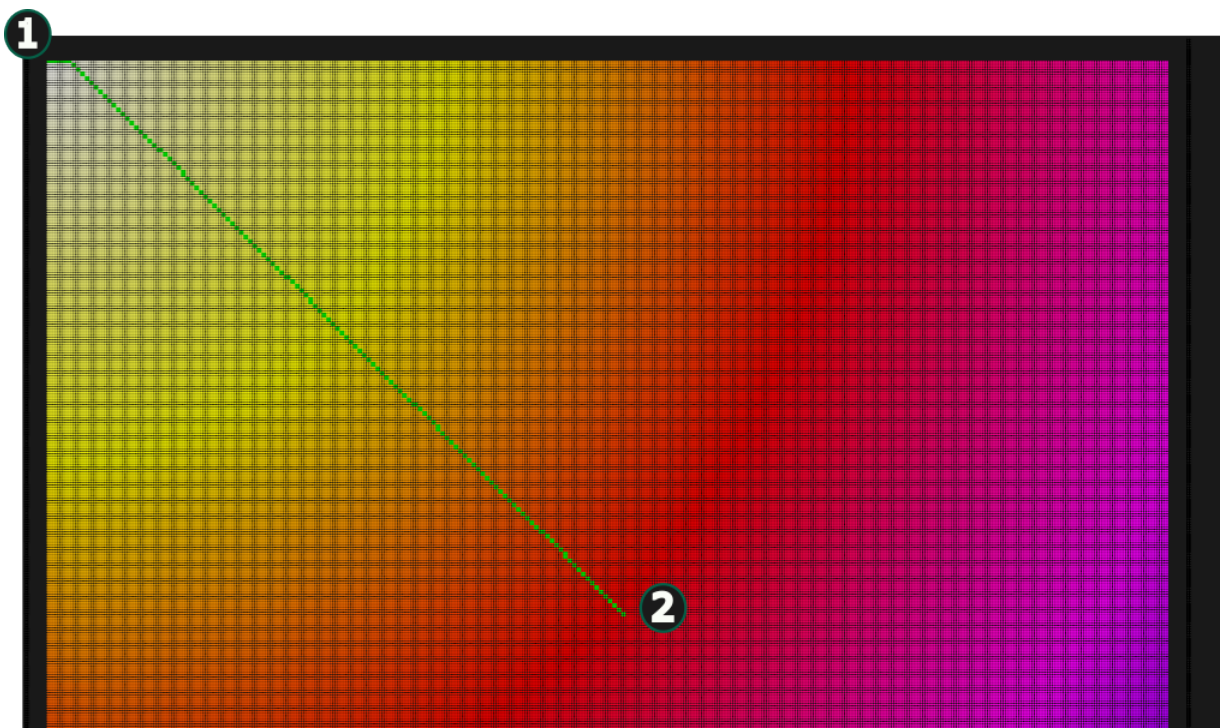


Figura 35: Camino geodésico sobre plano

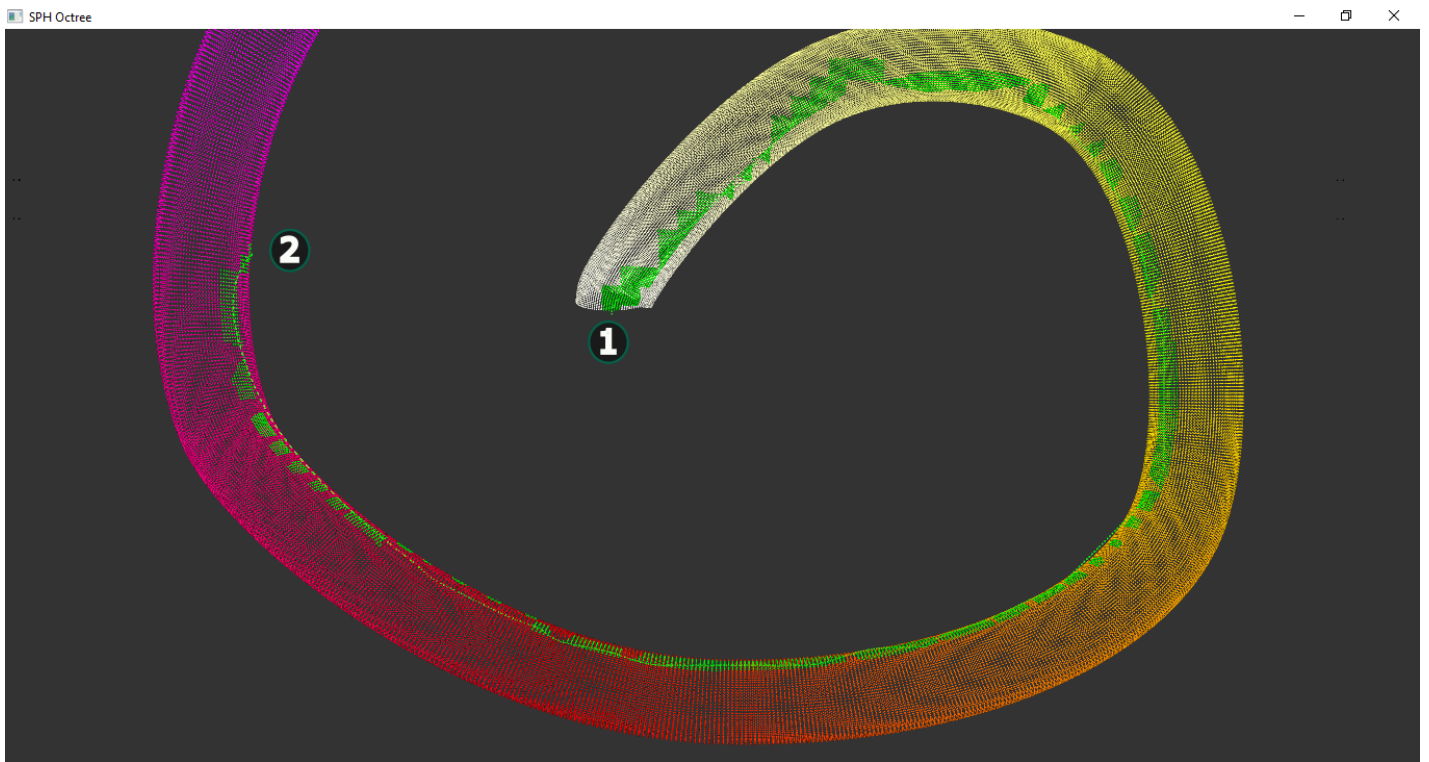


Figura 36: Camino geodésico sobre espiral

6. Análisis de resultados

6.1. Algoritmo de selección

Dentro del uso del algoritmo la selección de puntos debe ser una tarea muy rápida de realizar, permitiendo la posibilidad de seleccionar sin problemas puntos de la zona de manera continua.

En la tabla 1 donde se muestran los tiempos de procesamiento del algoritmo se observa un tiempo de procesamiento muy rápido (menor a $20ms$) hasta alcanzar $\approx 2 \times 10^6$ de puntos. Por otro lado, al llegar a una pieza con 7 millones de puntos se observó el aumento exponencial en el tiempo de procesamiento del algoritmo.

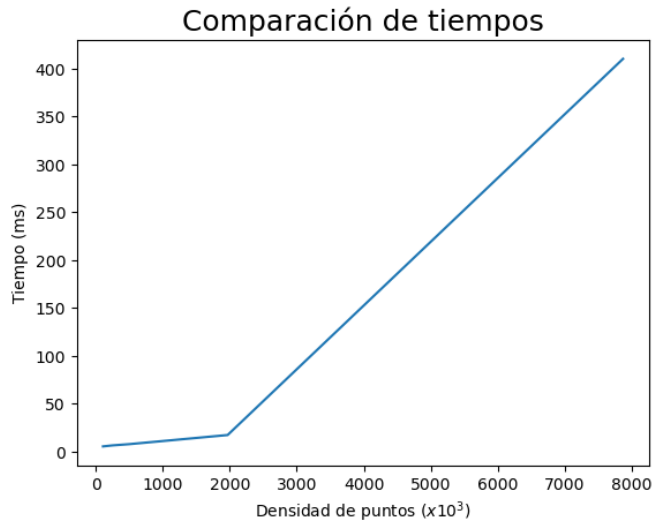


Figura 37: Comparación de tiempos del algoritmo de selección respecto a la densidad de puntos.

Aún cuando el aumento de tiempo es grande, el tiempo máximo de respuesta de $410ms$ es un resultado aceptable para esta operación.

6.2. Algoritmo de distancia

Aunque los resultados con geometrías conocidas muestran un porcentaje bajo de error relativo a la distancia máxima medida (tablas 2, 3), cuando la distancia máxima es muy grande esto ocasiona un error absoluto que llega a ser considerable. Analizando el comportamiento de este error en la medición se asume que está directamente relacionado con la discretización por cuadrícula tridimensional que se logra con el uso del octree. Esto es parte de la aproximación que se utilizó basándose en la idea de aproximar la distancia geodésica con distancias euclidianas. En este caso las distancias euclidianas son las distancias entre el centro de cada nodo del octree.

Esto es más claro al considerar la geometría de la pieza, pues en un plano el error será menor que en una pieza 3D. El caso de la esfera es la de mayor error posible pues es como seguir una circunferencia con solo líneas rectas de tamaño igual a la celda de menor tamaño de la cuadrícula. En este caso entre menor el tamaño de las celdas, mejor será la aproximación.

Hasta ahora no se han hecho pruebas controlando las dimensiones de los nodos del octree o la cantidad de partículas admisibles en cada uno. En el desarrollo del octree utilizado [2] se evita llegar a un octree completo, con una partícula para cada nodo, por evitar problemas de memoria

en la GPU. Sin embargo, podría buscarse un método para refinar ciertas partes del octree donde se procesará el camino geodésico, permitiendo encontrar resultados mas exactos en cuanto a distancia.

Por otro lado, respecto a la velocidad de procesamiento, hay dos aspectos que modifican los resultados: distancia máxima y densidad de puntos. En la figura 38 se muestra una comparación de como ambos factores afectan el procesamiento del algoritmo.

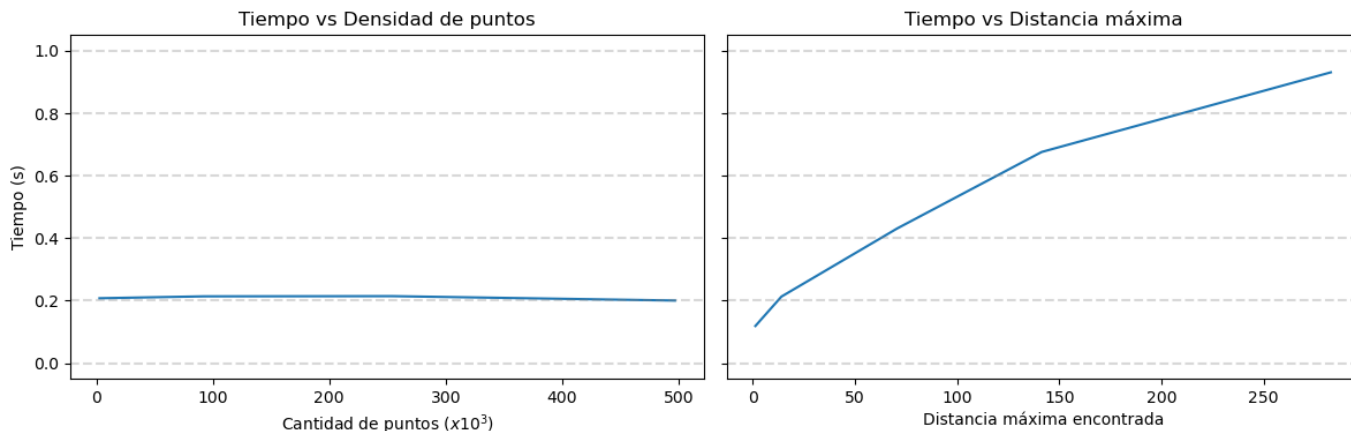


Figura 38: Cambio en el tiempo de procesamiento del plano respecto a la densidad de puntos y la distancia máxima encontrada.

Como se observa en la Fig. 38, el aumento en densidad de puntos no causa un cambio significativo, es la dimensión o distancia máxima a medir lo que cambia el tiempo de procesamiento, casi linealmente. Esto es un comportamiento esperado ya que una mayor distancia es una mayor cantidad de iteraciones necesarias para que la expansión de la onda llegue a ese punto. Por otro lado, la paralelización permite que la densidad de puntos no afecte en tiempo de procesamiento.

6.3. Comparación con software

En las tablas 9 y 10 se comparan los tiempo de ejecuciones de planos y media esfera de diferentes tamaños con dos software de procesamiento de nube de puntos. El primero es Meshlab [28] que es un estándar en el uso científico de software para análisis y manipulación de objetos 3D y el segundo es GIGAMESH [52] que es un software con enfoque para el uso arqueológico en análisis 3D. Los tiempos del algoritmo propuesto son la suma de sus 4 procesos.

Tabla 9: Comparación de tiempos al procesar planos de diferentes dimensiones con el algoritmo propuesto y los software Meshlab y GIGAMESH

Dim	Densidad de p	Dist. Max.	Tiempo (ms)		
			Propuesta	MESHLAB	GIGAMESH
1x1	108,875	1.414	119	609	603
10x10	254,012	14.14	213	1259	1677
50x50	495,612	70.71	431	2114	3514
100x100	1,968,409	141	676	8279	15889
200x200	7,868,025	282	931	35125	89739

Tabla 10: Comparación de tiempos al procesar la media esfera con diferentes dimensiones utilizando el algoritmo propuesto y los software Meshlab y GIGAMESH

Dim (radio)	Densidad de p	Dist. Max.	Tiempo (ms)		
			Propuesta	MESHLAB	GIGAMESH
1	3,977	3.1415	59.754	448	19
5	250,209	15.7	168.118	1271	1363
10	999,617	31.415	278.956	4162	7118
50	3,996,033	157	553.8686	14140	30418
100	3,996,033	314.15	933.219	43678	33044

En la Fig 39 se grafica la comparación de tiempos respecto a la distancia máxima encontrada.

Comparación de tiempos entre algoritmo y software

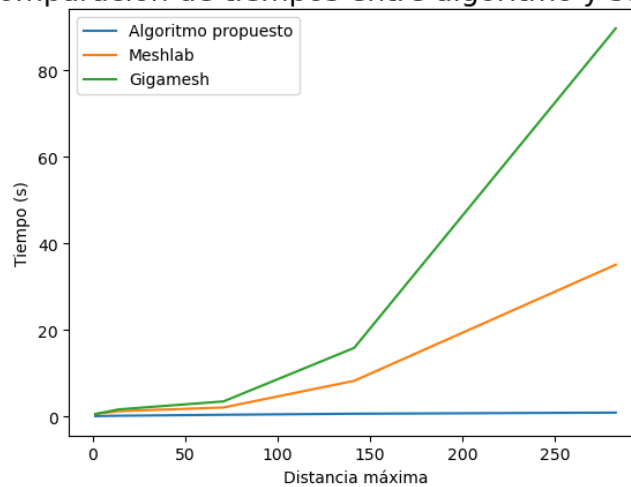


Figura 39: Comparación de tiempos respecto a distancia máxima encontrada entre el algoritmo propuesto y los software meshlab y gigamesh

Al comparar con algoritmos de distancia geodésica en un software popular como Meshlab y uno especializado en arqueología como GIGAMESH, se observa que el algoritmo propuesto tiene un procesamiento más rápido. En los peores casos de prueba, para Meshlab con la media esfera, el algoritmo propuesto fue aproximadamente un 50% más rápido y para GIGAMESH con el plano de 200x200 el algoritmo propuesto fue 100 veces más rápido.

7. Aplicación en arqueología

7.1. Pruebas del algoritmo de distancia geodésica con restos óseos encontrados en excavación

A continuación se muestran resultados del proceso de trabajo que se debe realizar para utilizar el algoritmo en el análisis de una pieza arqueológica.

Los restos óseos procesados son los del cráneo encontrado en la tumba II en la excavación de Tingambato [26]. La nube se recuperó por medio de escaneo 3D por luz estructurada y tiene una densidad de 1,211,511 puntos.

1. **Seleccionar el punto de inicio.** Se puede hacer un acercamiento a la zona de interés con el uso del teclado para mover la cámara. Se posiciona el puntero del ratón sobre el punto a seleccionar y se hace clic. El punto se muestra en rojo (Fig. 40).

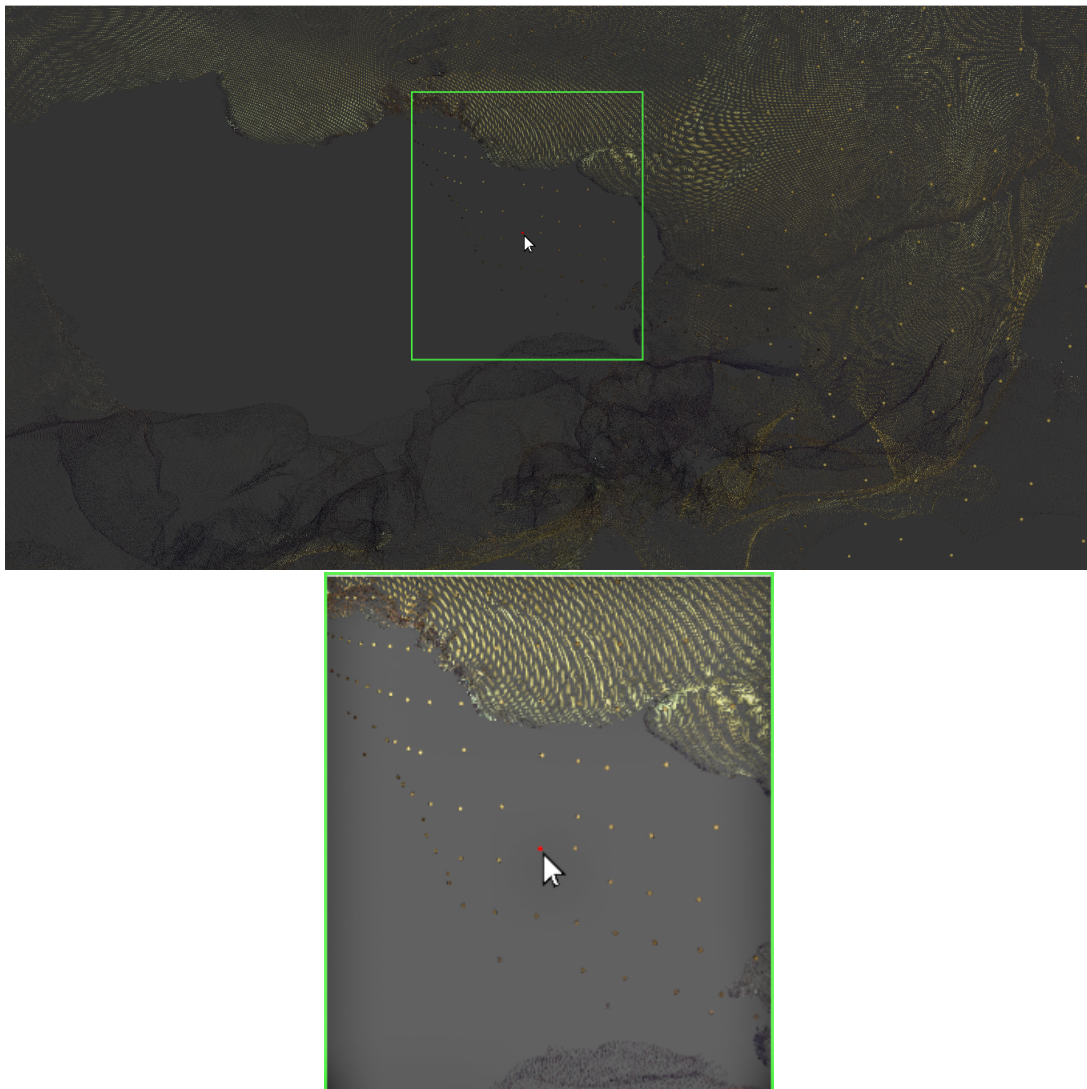


Figura 40: Proceso de trabajo con pieza Arqueológica. Seleccionar punto de inicio. La imagen inferior presenta un acercamiento con mayor iluminación.

2. Ejecutar el algoritmo de distancia.

Se ejecuta el algoritmo al presionar una tecla del teclado. Al terminar cada paso del algoritmo la consola mostrará sus tiempos de procesamiento. Después se mostrará la nube con un pseudocolor que va del amarillo al azul (Fig. 41). El pseudocolor está referido a la distancia desde ese punto al punto inicial.

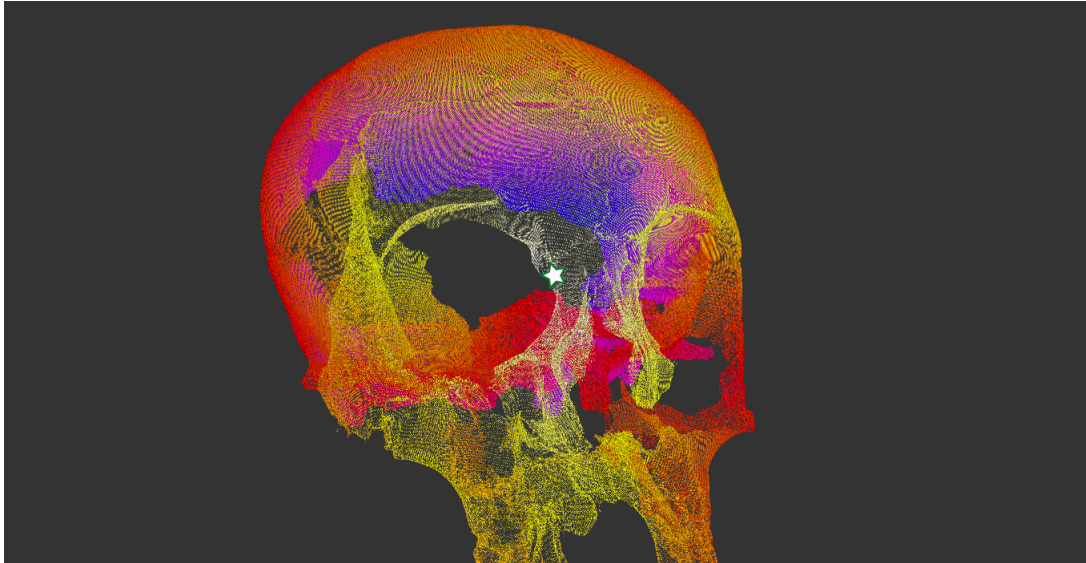


Figura 41: Proceso de trabajo con pieza Arqueológica. Ejecutar algoritmo de distancia de uno a todos los puntos. Se coloca una estrella para mostrar el punto de inicio.

3. Camino geodésico.

Para visualizar el camino geodésico se presionar el botón izquierdo sobre un segundo punto de la nube. El camino geodésico se mostrará en verde 42.

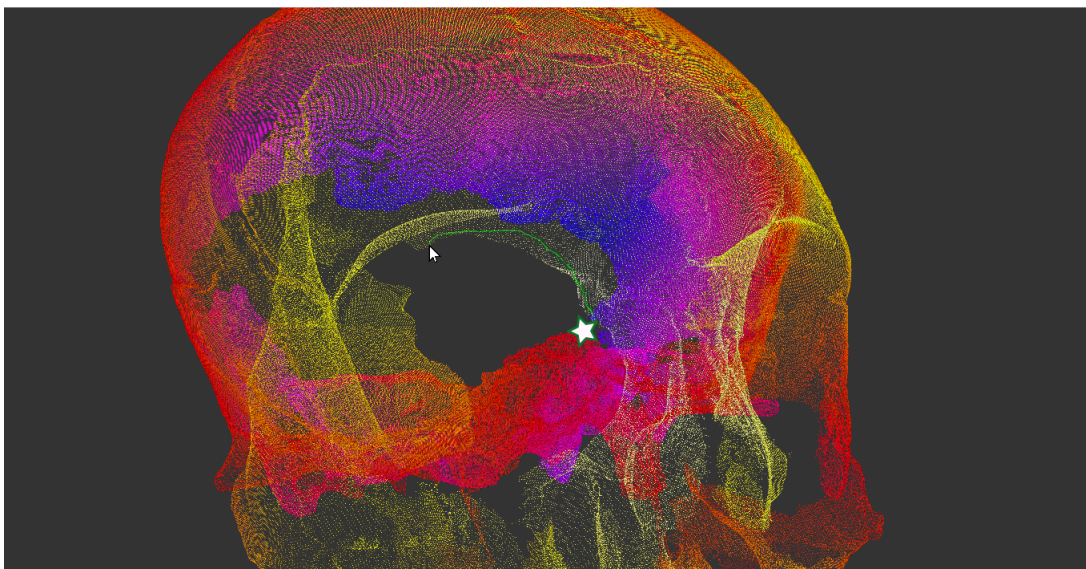


Figura 42: Proceso de trabajo con pieza Arqueológica. Seleccionar punto final del camino geodésico. Se coloca una estrella para mostrar el punto de inicio.

4. Camino geodésico múltiple

Después de dibujar el primer camino geodésico se selecciona un segundo punto. El algoritmo calculará automáticamente la segunda geodésica y la consola mostrará la distancia total. En la Figura 43 se muestra un ejemplo donde se usa el camino geodésico múltiple para seleccionar una zona circular.

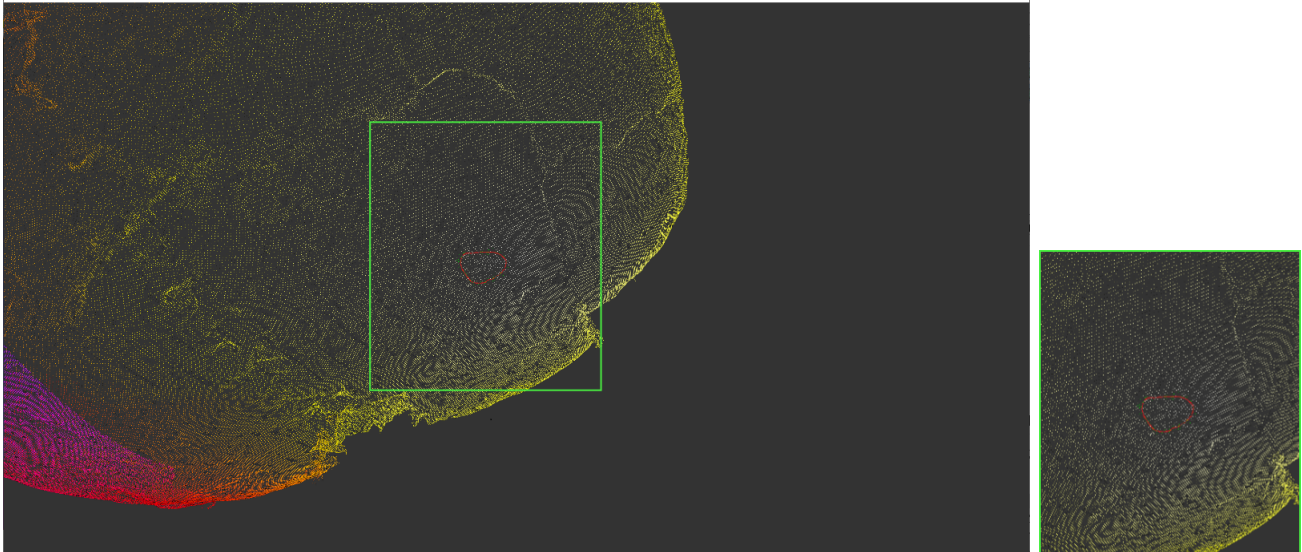


Figura 43: Proceso de trabajo con pieza Arqueológica. Selección de zona circular

7.1.1. Resultados de tiempo

La nube de puntos del cráneo cuenta con una densidad de 1,211,511 puntos. Los tiempos de procesamiento fueron: Para selección, un promedio de 8.295 ms. Para el algoritmo de distancia, 670.948 ms, para una distancia máxima de 295.788.

7.2. Pruebas del algoritmo de distancia geodésica con otros objetos arqueológicos

En la Fig. 44 se muestra el algoritmo sobre la nube de puntos de una reconstrucción por fotogrametría de una escultura encontrada en la Zona Arqueológica de Ihuatzio, Michoacán. El análisis es valioso debido a que la forma y tamaño de la figura permite hacer comparaciones de distancia con la pieza real.

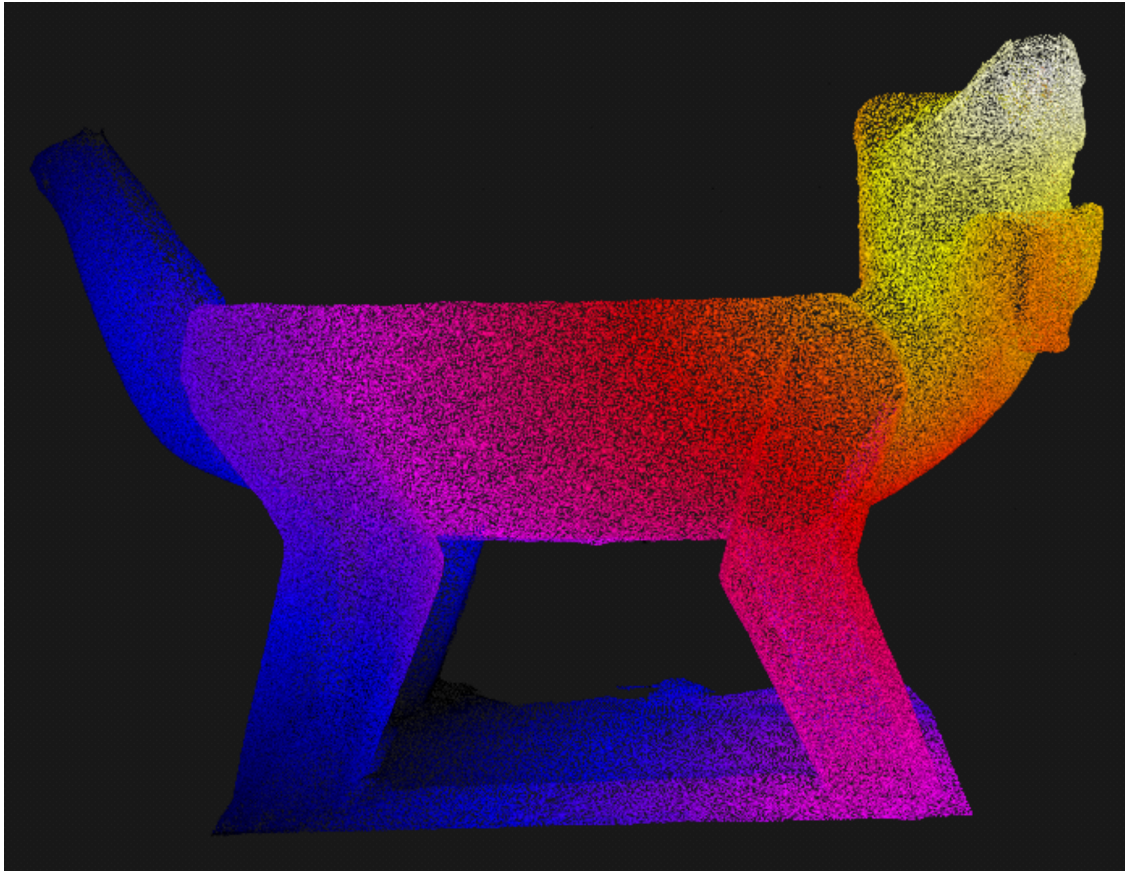


Figura 44: Resultado sobre el Coyote de Ihuatzio

En la Fig. 45 se muestra una pequeña figura de cerámica encontrada Tingambato. El resultado del algoritmo permite encontrar medidas en las hendiduras de la cerámica que representa una forma humanoide.

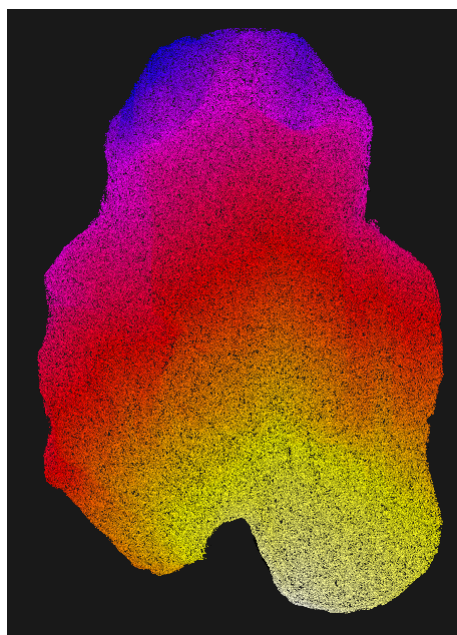


Figura 45: Resultado sobre Figura de cerámica excavada en sitio

Los resultados de tiempo para estos resultados se muestran en la tabla 11.

Tabla 11: Resultados de tiempo para procesamiento de distancia en objetos arqueológicos

Objeto	Densidad	Dist. Máx medida	Tiempo[ms]
Coyote	511944	33.91	289.159
Fig. Cerámica	444047	21.727	192.85

En la Fig. 46 se muestra el camino geodésico sobre la figura de cerámica.

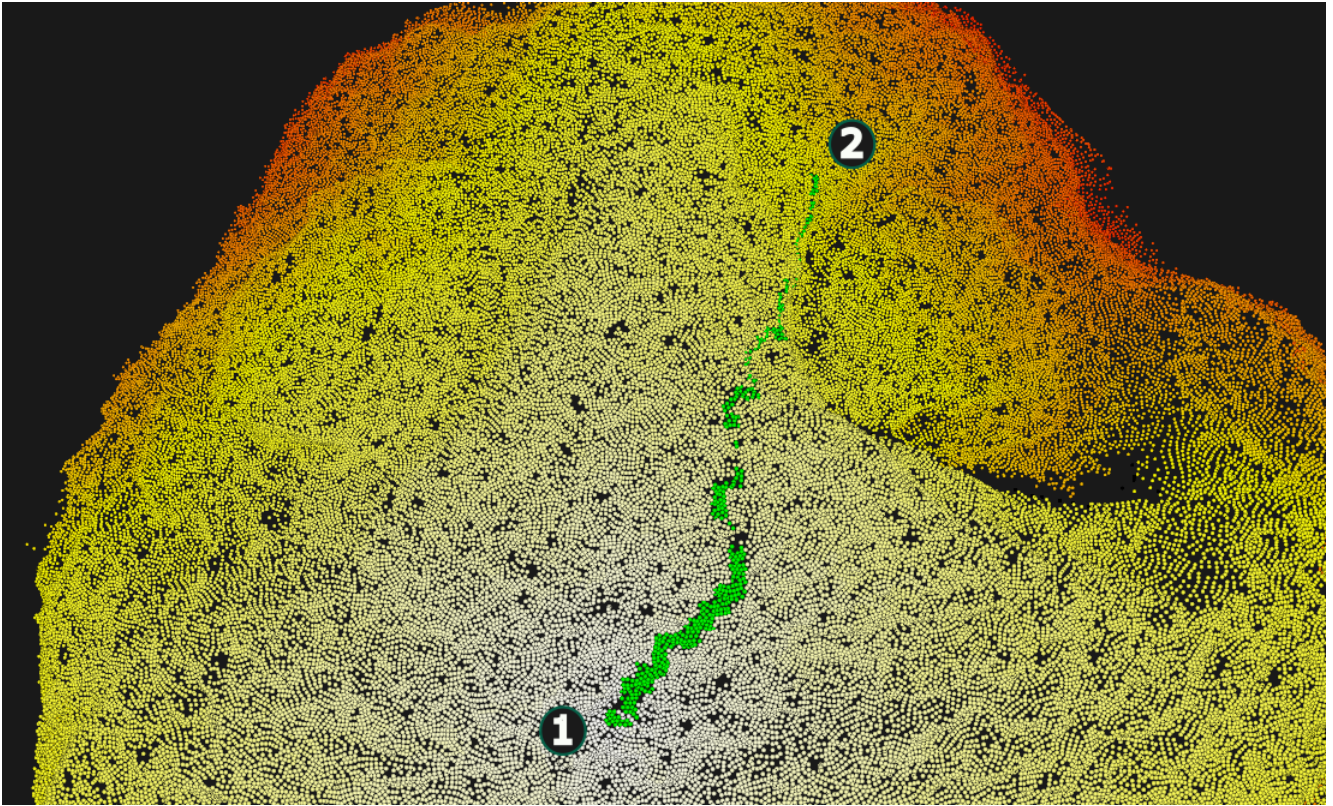


Figura 46: Camino geodésico sobre figura de cerámica (en verde).

Los resultados con pruebas reales muestran la capacidad del algoritmo de trabajar con geometrías de superficies reconstruidas. La velocidad de procesamiento se mantuvo dentro de lo esperado y la difusión del algoritmo muestra resultados adecuados.

8. Discusión

Hay dos cosas importantes a rescatar en este trabajo. Por un lado ¿Dónde nace la necesidad de usar nubes de puntos? Como se ha mencionado, el uso pensado para el algoritmo es sobre objetos 3D obtenidos mediante algún dispositivo de captura tridimensional. Estos dispositivos proporcionan nube de puntos como resultado inicial de la reconstrucción, lo que simplifica el proceso en comparación con medir sobre una malla 3D reconstruida. Esta aproximación elimina un paso, lo que no solo ahorra tiempo y reduce los costos de procesamiento, sino que también previene alteraciones en la superficie del objeto generadas por el algoritmo de reconstrucción de la malla 3D. Esto a su vez se traduce en una mejora sustancial en la exactitud y precisión que será posible obtener con las mediciones.

Por otro lado, es crucial resaltar la relevancia de este trabajo en el contexto de la arqueología. A pesar de los considerables esfuerzos realizados por los arqueólogos en la recopilación de datos a través de técnicas como la fotogrametría, el LiDAR y el escáner 3D, la capacidad de llevar a cabo análisis detallados de estos datos es un paso esencial que por ahora está subdesarrollado. Afortunadamente existen trabajos recientes en arqueología digital que ya resuelven algunas de estas problemáticas, sin embargo, hasta ahora la mayoría de ellos se limitan a tareas muy específicas y no existen muchas opciones en técnicas o herramientas públicas para que estos especialistas puedan medir directamente sobre nubes de puntos.

Trabajos como este cobran importancia ya que proporcionan a los arqueólogos la capacidad de obtener medidas reales de las nubes de puntos que generan. Esto, a su vez, brinda información valiosa sobre los objetos de estudio en arqueología, contribuyendo al avance y enriquecimiento de esta disciplina.

8.1. Limitaciones del método

Con lo presentado hasta ahora se tiene las siguientes limitaciones:

- El resultado numérico de distancia es dependiente de la discretización hecha por los nodos del octree. Como se demostró en [47], el resultado obtenido aproximando la suma de distancias euclidianas converge a la distancia geodésica real de la superficie. Sin embargo, existirá un error asociado a las dimensiones de los nodos que son sobre los que se esparce el frente de onda. Por la misma razón, la capacidad del algoritmo de detectar una superficie continua depende de que no existan nodos vacíos, o en el caso de detectar agujeros, de que existan nodos vacíos en la zona. Este comportamiento también puede ocasionar errores en determinadas circunstancias.
- El camino geodésico con la curva de Bézier no es la mejor aproximación al camino geodésico real, además la curva de 3er orden puede dar resultados incorrectos cuando hay una baja cantidad de nodos en el camino. Lo mejor sería utilizar una técnica de optimización como descenso del gradiente para obtener una curva geodésica más exacta.
- Una tercera limitante tiene que ver con la aplicación del método. Dependiendo del método de captura 3D, el objeto estará o no escalado a unidades reales. Por lo tanto, si la escala del objeto no tiene las dimensiones correctas el método no entrega medidas reales. Para lograrlo hay que referenciar el objeto recuperado con una escala u objeto con medidas conocidas. Por lo tanto aún requeriría un trabajo de ingeniería extra para que el método entregue datos útiles de la vida real.

9. Conclusiones

En esta tesis se siguió el desarrollo de un algoritmo para selección de puntos y medición de distancia geodésica sobre nube de puntos. La selección de puntos se resolvió con una técnica de selección *raycast*, lanzando un rayo desde la cámara a la escena para intersectar el punto seleccionado. La solución del problema de medición se basó en la resolución de la ecuación eikonal y aprovecha la computación paralela y una división espacial tipo Octree para ofrecer una solución eficaz. El algoritmo está pensado para procesar nubes de puntos obtenidas con tecnologías como fotogrametría o escaneo 3D, sobre piezas reales, por lo que se espera que sea capaz de trabajar con diversas geometrías de superficie y densidades de nube de puntos. Como caso de estudio el trabajo de tesis utiliza la arqueología, por ello, se presenta el estado actual en el uso de herramientas de arqueología, enfocado en la medición de restos óseos, y el panorama en el uso de tecnologías de captura 3D en arqueología digital.

En primer lugar, el objetivo de crear una herramienta que permita interactuar con un objeto 3D se logró con la implementación del método de *raypicking*. Este método al implementarse con cómputo paralelo (CUDA) resultó en una medida eficaz para la tarea común de seleccionar puntos de interés en un objeto tridimensional, en este caso, una nube de puntos.

En cuanto a las capacidades del algoritmo de distancia, con el uso del octree y CUDA se obtiene un algoritmo que procesa rápidamente nubes de puntos de considerable tamaño (probado con distancia máxima de aprox. 315 unidades de vértice) y densidad de puntos (probado con aprox. 8 millones de puntos).

Se realizó comparaciones de tiempo con softwares de procesamiento 3D populares como Meshlab, obteniendo en todos los casos un resultado más eficiente con el algoritmo propuesto. Las mediciones presentaron un bajo error relativo, respecto a la distancia máxima medida, y un error total creciente, al aumentar esta misma distancia. Se considera que la exactitud del algoritmo mejoraría enormemente teniendo un mayor control en las dimensiones de los nodos del octree de lo que se ha probado hasta ahora.

Por otro lado, y de acuerdo al tercer objetivo específico, gracias a una colaboración con el INAH, el algoritmo se pudo probar con restos arqueológicos encontrados en sitio. Específicamente con un cráneo con características de superficie interesantes para expertos en arqueología forense. Los resultados con el cráneo son positivos en cuanto a tiempo de procesamiento y capacidad de manipulación. Es posible, de manera intuitiva, dibujar caminos geodésicos sobre la superficie de objetos, siendo incluso capaz de encontrar circunferencias de agujeros.

Hablando de las aplicaciones del trabajo realizado, con el desarrollo de la tesis se obtiene una herramienta que acerca a no expertos en computación a tener la capacidad de realizar mediciones sobre nubes 3D que sigan la curvatura del objeto, y además hacerlo de forma eficiente. En un futuro se espera que este tipo de herramientas resuelvan la problemática existente de poder obtener más datos de interés de objetos capturados con tecnologías 3D.

Referencias

- [1] Guadalupe Lugo. “Estudian patrimonio con tecnología de punta e intervención multidisciplinaria”. En: *Gaceta UNAM* (2 de ene. de 2022). URL: <https://www.gaceta.unam.mx/estudian-patrimonio-con-tecnologia-de-punta-e-intervencion-multidisciplinaria/> (visitado 02-01-2023).
- [2] David Arturo Soriano Valdez et al. “CUDA Implementation of a Point Cloud Shape Descriptor Method for Archaeological Studies”. En: *Advanced Concepts for Intelligent Vision Systems*. Ed. por Jacques Blanc-Talon et al. Vol. 12002. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, págs. 457-466. ISBN: 978-3-030-40604-2 978-3-030-40605-9. DOI: 10.1007/978-3-030-40605-9_39. URL: http://link.springer.com/10.1007/978-3-030-40605-9_39 (visitado 17-01-2023).
- [3] Alexandros Iosifidis y Anastasios Tefas, eds. *Deep learning for robot perception and cognition*. London San Diego, CA\$Cambridge, MA\$Kidlington, Oxford: Academic Press, an imprint of Elsevier, 2022. 611 págs. ISBN: 978-0-323-85787-1.
- [4] Richard Hartley y Andrew Zisserman. *Multiple view geometry in computer vision*. Second edition. OCLC: 171123855. Cambridge, UK: Cambridge University Press, 2004. ISBN: 978-0-511-18711-7.
- [5] 3D FLOW. *Software 3DF Zephyr*. URL: www.3dflow.net/3df-zephyr-photogrammetry-software/ (visitado 01-12-2023).
- [6] Qlone. *Qlone App*. URL: <https://www.qlone.pro/> (visitado 01-12-2023).
- [7] Tomas Möller, Eric Haines y Naty Hoffman. *Real-time rendering*. Fourth edition. CRC Press, Taylor y Francis Group, 2018. 1 pág. ISBN: 978-1-351-81615-1.
- [8] John Nickolls et al. “Scalable Parallel Programming with CUDA: Is CUDA the Parallel Programming Model That Application Developers Have Been Waiting For?” En: *Queue* 6.2 (2008), 40–53. ISSN: 1542-7730. DOI: 10.1145/1365490.1365500. URL: <https://doi.org/10.1145/1365490.1365500>.
- [9] Leore Grosman. “Reaching the Point of No Return: The Computational Revolution in Archaeology”. En: *Annual Review of Anthropology* 45.1 (21 de oct. de 2016). Number: 1, págs. 129-145. ISSN: 0084-6570, 1545-4290. DOI: 10.1146/annurev-anthro-102215-095946. URL: <https://www.annualreviews.org/doi/10.1146/annurev-anthro-102215-095946> (visitado 24-05-2022).
- [10] Leore Grosman, Oded Smikt y Uzy Smilansky. “On the application of 3-D scanning technology for the documentation and typology of lithic artifacts”. En: *Journal of Archaeological Science* 35.12 (dic. de 2008), págs. 3101-3110. ISSN: 03054403. DOI: 10.1016/j.jas.2008.06.011. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0305440308001398> (visitado 15-01-2023).
- [11] Simone Parizzi y Carlo Beltrame. “Calculating the tonnage and the dimension of the cargoes of marble of Roman period”. En: *Digital Applications in Archaeology and Cultural Heritage* 18 (sep. de 2020), e00153. ISSN: 22120548. DOI: 10.1016/j.daach.2020.e00153. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2212054820300527> (visitado 10-02-2023).
- [12] T. White y P Folkens. *The Human Bone Manual*. Elsevier, 2005. ISBN: 0-12-088467-4.

- [13] Juan Comas. *Manual de antropología física*. 2nd ed. Mexico: Instituto de Investigaciones históricas, UNAM, 1966.
- [14] William M. Bass. *Human Osteology: A Laboratory and Field Manual*. 5th ed. Missouri Archaeological Society, 1971.
- [15] N. Langley y et al. *Data Collection Procedures for Forensic Skeletal Material 2.0*. 1st ed. Forensic Anthropology Center, The University of Tennessee, 2016.
- [16] Natalie Langley. *Evaluation of Osteometric Measurements in Forensic Anthropology*. U.S. Department of Justice, 2013.
- [17] Moore-Jansen et al. *Data Collection Procedures for Forensic Skeletal Material*. The University of Tennessee, 1994.
- [18] Heather M. Garvin y Michala K. Stock. “The Utility of Advanced Imaging in Forensic Anthropology”. En: *Academic Forensic Pathology* 6.3 (sep. de 2016), págs. 499-516. ISSN: 1925-3621, 1925-3621. DOI: 10.23907/2016.050. URL: <http://journals.sagepub.com/doi/10.23907/2016.050> (visitado 27-03-2023).
- [19] B. Mamabolo, A. Alblas y D. Brits. “Modern imaging modalities in forensic anthropology and the potential of low-dose X-rays”. En: *Forensic Imaging* 23 (dic. de 2020), pág. 200406. ISSN: 26662256. DOI: 10.1016/j.fri.2020.200406. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2666225620300555> (visitado 27-03-2023).
- [20] Kanchana Rathnayaka et al. “Quantification of the accuracy of MRI generated 3D models of long bones compared to CT generated 3D models”. En: *Medical Engineering Physics* 4.3 (2012), págs. 357-363. DOI: <https://doi.org/10.1016/j.medengphy.2011.07.027>.
- [21] Desiré M. Brits, Mubarak A. Bidmos y Paul R. Manger. “Stature estimation from the femur and tibia in Black South African sub-adults”. En: *Forensic Science International* 270 (2017), 277.e1-277.e10. ISSN: 0379-0738. DOI: <https://doi.org/10.1016/j.forsciint.2016.10.013>. URL: <https://www.sciencedirect.com/science/article/pii/S0379073816304509>.
- [22] Rhys Williams et al. “3D Imaging as a Public Engagement Tool: Investigating an Ox Cranium Used in Target Practice at Vindolanda”. En: *Theoretical Roman Archaeology Journal* 2.1 (21 de jun. de 2019). Number: 1, pág. 2. ISSN: 2515-2289. DOI: 10.16995/traj.364. URL: <http://traj.openlibhums.org/articles/10.16995/traj.364/> (visitado 24-05-2022).
- [23] José Luis Punzo Díaz. “New evidence for the occupation of tingambato (Western Mexico) during the classic and epiclassic periods”. En: *Arqueología Iberoamericana* 30 (2016), págs. 10-15.
- [24] José Luis Punzo Díaz. “Revisitando las exploraciones arqueológicas en el sitio de Tingambato, Michoacán: Nuevos datos, nuevas tecnologías”. En: *Latin American Antiquity* 33.1 (mar. de 2022), págs. 79-96. ISSN: 1045-6635, 2325-5080. DOI: 10.1017/laq.2021.65. URL: https://www.cambridge.org/core/product/identifier/S1045663521000651/type/journal_article (visitado 16-10-2023).
- [25] Piña Chan Román y Kuniaki OHI. “Exploraciones arqueológicas en Tingambato, Michoacán”. En: INAH, México, 1982.
- [26] José Luis Punzo Díaz y Alejandro Valdes Herrera. “La Tumba II de Tingambato: una morada para una joven mujer de élite en el Epiclásico mesoamericano”. En: *Americae. European Journal of Americanist Archaeology* 5 (2020). URL: <https://hal.science/hal-03197276>.

- [27] Revopoint. *Software Revo Scan 5*. URL: <https://global.revopoint3d.com/pages/revoscan5> (visitado 01-12-2023).
- [28] Paolo Cignoni et al. “MeshLab: an Open-Source Mesh Processing Tool”. En: *Eurographics Italian Chapter Conference*. Ed. por Vittorio Scarano, Rosario De Chiara y Ugo Erra. The Eurographics Association, 2008. ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.
- [29] David Arturo Soriano Valdez. “Simulación libre de mallas por partículas suavizadas optimizado mediante octree dinámico en CUDA”. Tesis doct. Universidad Nacional Autónoma de México, 2021.
- [30] J J Monaghan. “Smoothed particle hydrodynamics”. En: *Reports on Progress in Physics* 68.8 (1 de ago. de 2005). Number: 8, págs. 1703-1759. ISSN: 0034-4885, 1361-6633. DOI: 10.1088/0034-4885/68/8/R01. URL: <https://iopscience.iop.org/article/10.1088/0034-4885/68/8/R01> (visitado 27-05-2022).
- [31] M. S. Warren y J. K. Salmon. “A parallel hashed Oct-Tree N-body algorithm”. En: *Proceedings of the 1993 ACM/IEEE conference on Supercomputing - Supercomputing '93*. the 1993 ACM/IEEE conference. Portland, Oregon, United States: ACM Press, 1993, págs. 12-21. ISBN: 978-0-8186-4340-8. DOI: 10.1145/169627.169640. URL: <http://portal.acm.org/citation.cfm?doid=169627.169640> (visitado 19-04-2023).
- [32] Alfonso Gastelum Strozzi. “Smoothing Particle Hydrodynamics parallel implementation for numerical modelling of solid-fluid interactions”. Tesis doct. The University of Auckland, 2011.
- [33] Konrad Polthier y Markus Schmies. “Straightest Geodesics on Polyhedral Surfaces”. En: *Discrete Differential Geometry* (2006), pág. 9.
- [34] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. En: *Numerische mathematik* 1.1 (1959), págs. 269-271.
- [35] M Rodríguez et al. “Mapeo de Laberintos y Búsqueda de Rutas Cortas Mediante Tres Mini Robots Cooperativos”. En: 34.1 (2014).
- [36] Keenan Crane et al. *A Survey of Algorithms for Geodesic Paths and Distances*. Number: arXiv:2007.10430. 20 de jul. de 2020. arXiv: 2007.10430[cs]. URL: <http://arxiv.org/abs/2007.10430> (visitado 24-05-2022).
- [37] Jaume Masoliver y Ana Ros. “From Classical to Quantum Mechanics through Optics”. En: *European Journal of Physics* 31.1 (1 de ene. de 2010), págs. 171-192. ISSN: 0143-0807, 1361-6404. DOI: 10.1088/0143-0807/31/1/016. arXiv: 0909.3258[physics]. URL: <http://arxiv.org/abs/0909.3258> (visitado 09-08-2023).
- [38] Michael G. Crandall y Pierre-Louis Lions. “Viscosity Solutions of Hamilton-Jacobi Equations”. En: *Transactions of the American Mathematical Society* 277.1 (1983), págs. 1-42. ISSN: 00029947. URL: <http://www.jstor.org/stable/1999343> (visitado 23-11-2023).
- [39] Jinghong Miao. “Viscosity Solutions of the Eikonal Equations”. En: *The university of Chicago* (2020).
- [40] J. A. Sethian. “Fast Marching Methods”. En: *SIAM Review* 41.2 (1999), págs. 199-235. DOI: 10.1137/S0036144598347059. URL: <https://doi.org/10.1137/S0036144598347059>.
- [41] S.K. Godunov. “A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics”. En: *Mat. Sb. (N.S.)* 47.3 (1959), págs. 271-306.

- [42] J.A. Sethian. “Fast Marching Methods and Level Set Methods for Propagating Interfaces”. En: *Berkeley, University of California* (1998).
- [43] R. Kimmel y J. A. Sethian. “Computing geodesic paths on manifolds”. En: *Proceedings of the National Academy of Sciences* 95.15 (21 de jul. de 1998). Number: 15, págs. 8431-8435. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.95.15.8431. URL: <https://pnas.org/doi/full/10.1073/pnas.95.15.8431> (visitado 24-05-2022).
- [44] Maria Tugurlan. “Fast Marching Methods - parallel implementation and analysis”. Doctor of Philosophy. Louisiana State University, Agricultural y Mechanical College, 29 de ago. de 2008. DOI: 10.31390/gradschool_dissertations.236. URL: https://digitalcommons.lsu.edu/gradschool_dissertations/236 (visitado 02-12-2022).
- [45] Won-Ki Jeong y Ross T. Whitaker. “A Fast Iterative Method for Eikonal Equations”. En: *SIAM Journal on Scientific Computing* 30.5 (ene. de 2008), págs. 2512-2534. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/060670298. URL: <http://epubs.siam.org/doi/10.1137/060670298> (visitado 23-01-2023).
- [46] Sumin Hong, Ganghee Jang y Won-Ki Jeong. “MG-FIM: A Multi-GPU Fast Iterative Method Using Adaptive Domain Decomposition”. En: *SIAM Journal on Scientific Computing* 44.1 (feb. de 2022), págs. C54-C76. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/21M1414644. URL: <https://epubs.siam.org/doi/10.1137/21M1414644> (visitado 20-01-2023).
- [47] Facundo Memoli y Guillermo Sapiro. *Distance Functions and Geodesics on Points Clouds*: Fort Belvoir, VA: Defense Technical Information Center, 1 de ene. de 2005. DOI: 10.21236/ADA437158. URL: <http://www.dtic.mil/docs/citations/ADA437158> (visitado 02-12-2022).
- [48] Andrew Woo. “Fast Ray-box Intersection In Graphic Gems (pp 395-396)”. En: Elsevier, 1990.
- [49] Travian Barnes. *Fast, Branchless Ray/Bounding Box Intersections*. 2 de mayo de 2011. URL: https://tavianator.com/2011/ray_box.html (visitado 02-08-2023).
- [50] Hongkai Zhao. “A fast sweeping method for Eikonal equations”. En: *Mathematics of Computation* 74.250 (21 de mayo de 2004), págs. 603-627. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-04-01678-3. URL: <https://www.ams.org/mcom/2005-74-250/S0025-5718-04-01678-3/> (visitado 06-08-2023).
- [51] Blender Development Team. *Blender (Version 3.2.0)*. URL: <https://www.blender.org> (visitado 01-01-2022).
- [52] Hubert Mara et al. “GigaMesh and Gilgamesh 3D Multiscale Integral Invariant Cuneiform Character Extraction”. En: *VAST: International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*. Ed. por Alessandro Artusi et al. The Eurographics Association, 2010. ISBN: 978-3-905674-29-3. DOI: 10.2312/VAST/VAST10/131-138.